UNIVERSITY OF LJUBLJANA
Faculty of Computer and Information Science

Andrej Bratko

Text Mining Using Data Compression Models

DOCTORAL THESIS

Thesis supervisor: Prof. Dr. Blaž Zupan
Thesis co-supervisor: Prof. Dr. Bogdan Filipič

Ljubljana, 2012

UNIVERZA V LJUBLJANI
Fakulteta za računalništvo in informatiko

Andrej Bratko

Iskanje zakonitosti v besedilih s kompresijskimi modeli

DOKTORSKA DISERTACIJA

Mentor: Prof. Dr. Blaž Zupan
Somentor: Prof. Dr. Bogdan Filipič

Ljubljana, 2012

IZJAVA O AVTORSTVU

doktorske disertacije

Spodaj podpisani **Andrej Bratko**,
z vpisno številko **63970023**,

sem avtor/-ica doktorske disertacije z naslovom

Iskanje zakonitosti v besedilih s kompresijskimi modeli
(angl. Text Mining Using Data Compression Models)

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelal/-a samostojno pod vodstvom mentorja

  prof. dr. Blaža Zupana

  in somentorstvom

  prof. dr. Bogdana Filipiča

- so elektronska oblika doktorske disertacije, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko doktorske disertacije

- in soglašam z javno objavo doktorske disertacije v zbirki ≫Dela FRI≪.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Abstract

The idea of using data compression algorithms for machine learning has been reinvented many times. Intuitively, compact representations of data are possible only if statistical regularities exist in the data. Compression algorithms identify such patterns and build statistical models to describe them. This ability to learn patterns from data makes compression methods instantly attractive for machine learning purposes.

In this thesis, we propose several novel text mining applications of data compression algorithms. We introduce a compression-based method for instance selection, capable of extracting a diverse subset of documents that are representative of a larger document collection. The quality of the sample is measured by how well a compression model, trained from the subset, is able to predict held-out reference data. The method is useful for initializing k-means clustering, and as a pool-based active learning strategy for supervised training of text classifiers. When using compression models for classification, we propose that trained models should be sequentially adapted when evaluating the probability of the classified document. We justify this approach in terms of the minimum description length principle, and show that adaptation improves performance for online filtering of email spam.

Our research contributes to the state-of-the-art of applied machine learning in two significant application domains. We propose the use of compression models for spam filtering, and show that compression-based filters are superior to traditional tokenization-based filters and competitive with the best known methods for this task. We also consider the use of compression models for lexical stress assignment, a problem in Slovenian speech synthesis, and demonstrate that compression models perform well on this task,

while requiring fewer resources than competing methods. The topic of this thesis is text mining. However, most of the proposed methods are more general, and are designed for learning with arbitrary discrete sequences.

Keywords: text mining, compression, instance selection, active learning, k-means initialization, spam filtering, lexical stress assignment.

Povzetek

Kompakten zapis podatkov, ki omogoča pretvorbo nazaj v izvorno obliko brez izgube informacije, je možen le v primeru, da podatki vsebujejo določene ponavljajoče se vzorce ali statistične regularnosti. Algoritmi za kompresijo podatkov pri svojem delovanju takšne vzorce odkrijejo in gradijo statistične modele, s katerimi odkrite vzorce opisujejo. V disertaciji se posvečamo pogosto zastavljenemu vprašanju. Zanima nas, kolikšna je uporabna vrednost statističnih kompresijskih modelov za reševanje problemov v strojnem učenju.

V delu smo preučili možnosti uporabe kompresijskih modelov v različnih aplikacijah s področja odkrivanja zakonitosti v besedilih. Na osnovi kompresijskih algoritmov smo razvili novo metodo za izbor informativnih dokumentov. Metoda omogoča identifikacijo podmnožice oziroma vzorca dokumentov, ki so hkrati raznovrstni in vsebinsko reprezentativni za večji korpus besedil. Kakovost vzorca besedil ocenimo tako, da na podlagi vzorca zgradimo kompresijski model in preverimo, kolikšna je kompresivnost dobljenega modela na referenčnem korpusu, ki ga z vzorcem skušamo opisati. Uporabnost predlagane metode pokažemo na primeru inicializacije algoritma za razvrščanje v skupine k-povprečij in za izbor najinformativnejših učnih primerov za učenje klasifikatorjev.

Pri uporabi kompresijskih modelov za klasifikacijo predlagamo, da naj se naučeni kompresijski modeli pri oceni verjetnosti testnega besedila sproti prilagajajo vhodnemu besedilu. Postopek utemeljimo na podlagi načela najkrajšega opisa in pokažemo, da tovrstno prilagajanje modela izboljša rezultate na domeni filtriranja nezaželene elektronske pošte.

Disertacija prispeva k izboljšanju napovednih točnosti doslej znanih metod v dveh aplikativnih domenah. Predlagali smo uporabo kompresijskih modelov za filtriranje nezaželene elektonske pošte, kjer pokažemo, da so kompresijski

modeli primernejši od klasičnega pristopa, ki temelji na pretvorbi dokumentov v atributni opis na podlagi razčlembe po besedah. Z obsežnim primerjalnim ovrednotenjem pokažemo, da kompresijski modeli dosegajo primerljive ali boljše rezultate kot doslej najuspešnejše metode na tem področju. Kompresijske algoritme uspešno prilagodimo in uporabimo tudi za problem avtomatskega določanja naglasnih mest in tipa naglasa slovenskih besed. Problem je pomemben za sintezo slovenskega govora. S kompresijskimi metodami izboljšamo točnost obstoječih metod za avtomatsko naglaševanje v slovenščini brez uporabe dodatnih jezikovnih označb besedila, ki so potrebne v konkurenčnih metodah.

Čeprav se v delu osredotočamo na probleme s področja iskanja zakonitosti v besedilih, je večina obravnavanih metod primerna za učenje s poljubnimi podatki predstavljenimi v obliki diskretnih zaporedij.

Ključne besede: iskanje zakonitosti v besedilih, kompresija, aktivna izbira primerov, aktivno učenje, inicializacija algoritma k-povprečij, filtriranje nezaželene elektronske pošte, naglaševanje.

Acknowledgements

First and foremost, I wish to thank my supervisor Blaž Zupan, for motivating and guiding my research, and my co-supervisor Bogdan Filipič, for his advice and patient support throughout this journey. I am grateful to members of my reading committee Igor Kononenko and Gordon Cormack, for devoting their time to reviewing my work, and for their insightful comments and feedback.

Many thanks to Tea, Domen, Bernard, Mitja, Andraž, Sandi, and other friends and colleagues who helped make my time at the Jožef Stefan Institute a wonderful experience, and to my friends at Klika, for giving me the opportunity to complete this work.

Finally, I wish to thank my parents, and my dear Simona, for her unconditional support, her faith and understanding, and her love.

Andrej Bratko

Ljubljana, November 2012

To my daughter Živa and my son Gašper.

Contents

Chapter 1

Introduction

Text mining is concerned with extracting patterns, structured knowledge or descriptive models from text. The aim is to help users understand, organize and navigate text collections, or to facilitate machine processing of information contained in unstructured text. The field brings together methods of machine learning and statistics, data mining, information retrieval and computational linguistics.

Text can be mined in many ways. A classical text mining task is supervised classification of documents which, for example, helps keep mailboxes free from email spam. Another well-known task is clustering, which might be used by online news portals to estimate the importance of news stories. Other text mining tasks include named entity recognition and information extraction, extraction of document keywords, summarization, analysis of links between documents, and many more.

In its raw form, text is presented as a sequence of characters consisting of letters, digits and punctuation. This representation is difficult to model for standard machine learning methods, which typically accept data in the form of attribute vectors. Thus, the most common first step in text mining is to transform text into tokens which roughly correspond to words. For classification and clustering, this usually results in the "bag-of-words" vector representation, in which the order of words is lost. Applications for which the order of words is important typically use word pairs or longer sequences of consecutive words as attributes, or rely on various generative or discriminative Markov models over the sequence of word tokens.

In this thesis we take a different, less common approach to modeling text. We model text in its raw form, as a sequence of characters, and omit tokenization altogether. This approach has many potential advantages. It avoids ad hoc decisions regarding parsing and tokenization, which may have an effect on mining performance. These decisions include the handling of numbers, whitespace, punctuation, word inflection,

letter capitalization, etc. No information is lost in the representation, allowing the classifier to discover patterns that are otherwise lost due to pre-processing, as well as patterns that span across more than one word. Language-specific treatment of text is also avoided, such as stemming and stop-word removal. Note that for some languages, such as Chinese, segmentation of text into words is not trivial.

We model text using sequential prediction models originally developed for data compression. These models predict consecutive characters in the text, usually with the assumption that each character depends only on a limited number of characters that immediately precede it. As general probabilistic models over sequences, such models can be readily applied in many text mining tasks. Information theory tells us that the accuracy of sequential predictions directly affects the length of the compressed representation. Several learning theories further substantiate the notion that compression models are well-suited for machine learning tasks. In particular, the criteria for which adaptive compression models have been optimized for decades are exactly the criteria suggested for model selection according to the minimum description length theory of inference, and the testbed for evaluation and selection of compression models has traditionally been text. Furthermore, learning theory suggests that any "patterns" that cannot be usefully exploited for compression are most likely detected by chance due to over-fitting the data. The goals of compression and learning are therefore aligned.

This dissertation contributes to an increasing body of research in text mining, and machine learning in general, which draws on methods originally developed for data compression. Although text mining applications are at the core of this dissertation, most of the methods are suited for learning with arbitrary discrete sequences.

1.1 Research Topics

This thesis is concerned with novel practical applications of established compression algorithms in text mining, and the development of new text mining methods based on data compression models. We investigate many different text mining problems, namely instance selection (which is then further evaluated for active learning and text clustering), text classification, and sequence annotation.

Instance selection (representative sampling using compression models). Instance selection is the process of purposefully selecting a subset of examples according to some quality criterion. What exactly qualifies as a good sample often depends on the underlying application for which instance selection is performed. In this thesis, we consider the task of unsupervised instance selection, aimed at identifying a representa-

tive and diverse subset of documents from a large document collection. To this end, we experiment with training a compression model from a sample of texts, and measuring how well the model is able to compress the remaining, unsampled data. We investigate whether compression models can be used in this way to measure the representativeness of a data sample. Are the identified representative texts meaningful? Can this measure be computed efficiently? Is this approach useful for any practical text mining problems?

Classification (spam filtering using compression models). Given a set of training documents for which class labels are known, a text classifier learns to predict the labels of new, previously unseen texts. In this thesis, we explore the use of compression models for online spam filtering, an important application domain for online semi-structured document classification, which has seen a tremendous amount of research. When using compression models for filtering, the classification outcome is determined by the compression rates achieved by models trained from spam and legitimate email. We examine how compression models compare to traditional tokenization-based spam filters and state-of-the-art methods for this application domain. We also analyze the effect of online adaptation of sequential prediction models when such models are used for filtering.

Sequence annotation (lexical stress assignment using compression models). Sequence annotation is related to classification in that parts of the sequence must be assigned class labels. However, the label assignments are not necessarily independent in sequence annotation. In this thesis, we consider the use of statistical data compression methods to predict lexical stress for Slovenian word-forms, a specific sequence annotation problem with implications for Slovenian speech synthesis. We experiment with using compression models to determine the likelihood of a possible stress assignment solution. We investigate different techniques and modeling choices in this application domain, and compare compression models to methods considered in other studies.

1.2 Contributions

The reported work contributes new methods or insights (contributions A and B), and advances the state-of-the-art in two significant application domains (contributions C and D).

A. The cross-entropy reduction sampling (CERS) method. The CERS method introduces a compression-based approach for identifying a subset of representative and

diverse documents from a (typically large) document collection (representative sampling of documents).

- We propose a novel use of the cross-entropy measure to guide a search for representative examples, by measuring how well a generative model trained from the sample is able to predict held-out reference data.
- We develop an efficient algorithm to estimate cross-entropy reduction for sequential data, i.e. for representative sampling of arbitrary discrete sequences.
- A heuristic is developed which can be used to extract informative subsequences as a by-product of running the CERS algorithm.
- We empirically demonstrate the utility of the approach for two standard text mining tasks: initialization of k-means clustering and active learning for text classification.
- Finally, the method is used to extract representative stories and keywords from a stream of broadcast news.

B. Analysis of adaptive sequential prediction for classification. We expose the distinctions between applying trained probabilistic models that are "fixed" at classification time (the standard approach in machine learning) and adapting the models when classifying an example, a tactic inspired by adaptive data compression.

- We suggest the use of model adaptation when performing sequence classification, and justify this approach in terms of the minimum description length principle.
- An experimental evaluation for spam filtering shows that adaptation consistently improves filtering performance according to the AUC measure.
- We provide observational evidence indicating that adaptation penalizes redundant data pointing towards a particular classification outcome.

C. Advances in spam filtering.

- We propose the use of adaptive compression algorithms for spam filtering, which is found to outperform known prior methods considered for this task.
- We show that compression methods are less affected by obfuscation of text which is often used against tokenization-based filters in the adversarial spam filtering setting.

- Our work has contributed towards a broader understanding that character-based features are generally more suitable for learning-based spam filtering compared to traditional word-based tokenization.

D. Advances in lexical stress assignment for Slovenian.

- We propose a compression-based approach to predict lexical stress for Slovenian word-forms, which outperforms previously considered methods for this task, while requiring no additional linguistic features that are used in competing methods.

1.3 Organization of the Thesis

The remainder of the thesis is structured as follows.

Chapter 2 reviews the relationship between data compression and machine learning in theory and in practice. We also review prior work which considers the use of compression methods for text mining (although the work that is most closely related to the contributions of this thesis is also discussed in later chapters). This chapter places our work in a broader context, and covers relevant material for readers not familiar with the field.

Chapter 3 describes three established adaptive context-based data compression algorithms, which use different types of variable-order Markov models for sequential prediction. These sequential prediction models are the corner-stone of all methods described in this work.

Chapter 4 presents the CERS method for representative instance selection, and describes an algorithm for representative sampling of sequential data. Experimental results for initialization of k-means clustering and for active learning in binary text classification problems are reported. A study in which CERS is used for mining a stream of news is also described in this chapter.

Chapter 5 describes the use of compression models for spam filtering. The filtering performance of compression models is compared extensively with other spam filters in various settings. This chapter includes an analysis of the effect of adaptation of sequential prediction models when classifying a target document.

Chapter 6 reports on an application of compression methods for prediction of lexical stress for Slovenian words, with an analysis of certain modeling choices and techniques which critically affect the performance of compression methods in this application domain, and an experimental comparison against other methods proposed for this task.

Chapter 7 concludes the thesis with a brief summary and an outlook on future work.

Chapter 2

Background

The idea of using data compression algorithms in machine learning has been reinvented many times. The intuition behind this idea arises from the fact that compact representations of data are possible only if some kind of recurring patterns or statistical regularities are found in the data. Any general-purpose compression program *must* be able to detect such patterns in order to fulfill its purpose. Intuitively, it is this inherent *ability to detect patterns in data* that makes data compression algorithms instantly attractive for machine learning purposes.

This chapter covers the relevant theoretical and practical background which rationalizes the use of data compression methods in machine learning, with a particular focus on text mining applications. We first introduce some basic terms and notation that is used throughout the thesis (Section 2.1), followed by a short primer on coding theory, which establishes the duality between compression and prediction (Section 2.2). We review different types of data compression algorithms in Section 2.3. Theoretical machine learning paradigms which substantiate the relationship between compression and learning are discussed in Section 2.4. We conclude this section with a review of previous studies involving machine learning applications of algorithms originally developed for data compression, with a particular focus on text mining applications (Section 2.5).

2.1 Basic Terms and Notation

Sequence. We are generally concerned with learning from discrete *sequences*. For our purposes, sequences are independent units of data, manifested as finite-length sequences of *symbols* from a discrete *alphabet*. For example, a sequence can be a written text (a sequence of letters, digits and punctuation), or a musical composition (a sequence of notes and pauses). This thesis is focused on text data. However, the methods discussed are usually designed for modeling arbitrary discrete sequences. In the context of different

machine learning applications, we often prefer to use the term *example*, *instance*, or *data point* instead of the term *sequence*. In coding theory, a sequence is usually referred to as a *message*.

Symbol. Symbols are the basic units from which sequences are built. While sequences are independent units of data, it is typically assumed that there is some dependence among the symbols that comprise a sequence, and that some combinations of symbols are therefore more likely than others.

Alphabet. The set of available symbols is called an alphabet. It should be noted that there is much flexibility in how an alphabet is defined, and that the same data can be modeled with different alphabets. For example, a text data file can be modeled as a sequence of bits, a sequence of ASCII characters, a sequence of bytes, etc. The respective alphabets contain 2, 127, and 256 distinct symbols. One could use a different symbol to represent each possible word, or even each possible message, assuming that the set of possible messages is finite.

Model. We assume that the objective of learning is to infer a *model* or *hypothesis* from some observed data, and that the data (or *evidence*) on which our inferences are based is in the form of discrete sequences of symbols. A model is taken to represent a probability distribution over sequences. The inferred model can be used, for example, to provide an explanation of the data, or to predict future outputs of the data-generating process, or, as we shall see, to compress the data. We adopt this meaning of the word "model" from coding theory, although we note that this is in contrast to the meaning often found in statistical literature, where a "model" is typically a family of distributions taking the same parametric form, or the broader meaning of the term "model" in machine learning, where it includes inferences with non-probabilistic or structured outputs.

Model class. A *model class* is a set or continuum of models which are often related according to some property, depending on the context of the discussion, for example, models specified by the same parametric form (in some contexts, what we refer to as a *model class* might be considered a *model* by statisticians). In a typical setting, a model class is a family of possible hypotheses of similar "complexity".

Basic notation. We use strong lowercase letters to denote sequences in mathematical notation, for example $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, etc. The number of symbols, or *length*, of a sequence $\mathbf{x}$ is represented by $\overline{\mathbf{x}}$. We generally allow the length of a sequence to be arbitrary. The letter Σ is used to denote the symbol alphabet. The set of all possible sequences is

thus Σ^*. An *empty string* with a length of zero is denoted by ϵ. Lowercase letters are used for symbols $a \in \Sigma$ and subscripts x_i are used to denote a symbol at a particular position i in a sequence $\mathbf{x}$. The notation $\mathbf{x}_i^j$ is taken to represent a subsequence of $\mathbf{x}$ starting at position i and ending at j, so that $\mathbf{x}_1^n = \mathbf{x}$ if $\overline{\mathbf{x}} = n$. If $j < i$, we set $\mathbf{x}_i^j = \epsilon$ by definition. The number of occurrences of a symbol $a \in \Sigma$ in a sequence $\mathbf{x}$ is given by $f(a, \mathbf{x})$, while the set of distinct symbols that appear in a sequence $\mathbf{x}$ is denoted by $\Lambda(\mathbf{x}) = \{a \in \Sigma; f(a, \mathbf{x}) > 0\} \subseteq \Sigma$. We use $\mathbf{x} \| \mathbf{y}$ to denote the sequence of individual symbols in $\mathbf{x}$ which immediately follow occurrences of the subsequence $\mathbf{y}$. For example, $\texttt{abracadabra} \| \texttt{a} = \texttt{bcdb}$, and $\texttt{abracadabra} \| \texttt{ab} = \texttt{rr}$. The concatenation of two strings $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\mathbf{xy}$, while $\mathbf{x}a$ is the extension of $\mathbf{x}$ with the symbol a. Roman uppercase letters (e.g. X, Y, Z) are used to represent *sequence sets – classes* or *clusters* of examples in the context of machine learning applications. Each sequence $\mathbf{x} \in \mathrm{X}$ is a single data point with respect to the sequence set X. A model is usually denoted by θ, while $\theta(\mathbf{x})$ or $\theta(\mathrm{X})$ represents a model inferred, or trained, from a sequence $\mathbf{x}$ or a sequence set X. We use Θ to denote a model class.

2.2 An Elementary Introduction to Coding Theory

Information theory is a branch of applied mathematics historically concerned with fast and reliable communication of data. The fundamental concepts and laws of the field were laid out over half a century ago by C. E. Shannon with the publication of his seminal paper *A Mathematical Theory of Communication* (Shannon, 1948). The field has since found applications in many diverse areas of science, and has strongly influenced various approaches to statistical inference and machine learning. This section provides a brief, introductory treatment of the relevant aspects of *coding theory* – a branch of information theory concerned with *compression* of data.

2.2.1 Codes and Compression

A *code* is a mapping from input symbols to sequences of bits (*codewords*). The process of converting an input sequence (*message*) into a binary sequence by replacing each symbol in the message with its corresponding codeword is called *encoding*. A code is *uniquely decodable* or *decipherable* if two different messages are never encoded into the same bit string, which is required for the encoding to be reversible. When the purpose of encoding is to produce a compact representation of the original message, the process is called *compression*.

Codes in which codewords for different symbols differ in length are called *variable-length codes*. A *prefix-free code* is a variable-length code in which no codeword is a

prefix of another codeword. Prefix-free codes have the practical advantage that the boundary between codewords can be detected *instantaneously*, that is, it can be determined whether a certain position i in a binary sequence $\mathbf{x}$ is a codeword boundary by examining only the sequence of bits $\mathbf{x}_1^i$ leading up to the target position i. Such codes are also called *self-delimiting codes*. A prefix-free code can be represented as a binary tree with one leaf for each codeword, as depicted in Figure 2.1.

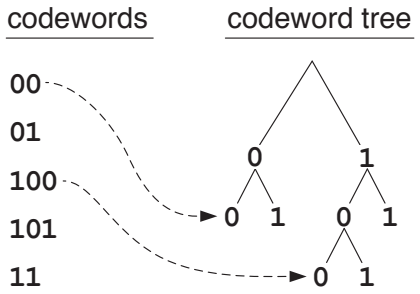


Figure 2.1: An example of a prefix-free code with the corresponding binary tree representation.

Let $\mathbf{c}_a$ denote the codeword used to encode the symbol a using a prefix-free code. Recall that each codeword $\mathbf{c}_a$ is represented by a leaf with depth $\overline{\mathbf{c}}_a$ in the corresponding binary tree representation of the code. If we imagine that the branches of each node in this binary tree divide a unit volume of space at the root of the tree exactly in half, we arrive at the following inequality (Kraft's inequality):

$$\sum_{a \in \Sigma} \left(\frac{1}{2}\right)^{\overline{\mathbf{c}}_a} \leq 1 \quad . \tag{2.1}$$

This inequality states that the sum of the volumes at the leaves cannot exceed the initial volume of 1 at the root of the tree.[1] The significance of Kraft's inequality is that it imposes a limited "budget" for codewords in a prefix-free code, with shorter codewords being "more expensive".

Let $\mathbf{c}_\mathbf{x}$ denote the encoded representation of a message $\mathbf{x}$, i.e. the concatenation of codewords which correspond to the sequence of symbols in $\mathbf{x}$. The length, in bits, of $\mathbf{c}_\mathbf{x}$ is then

$$\overline{\mathbf{c}}_\mathbf{x} = \sum_{a \in \Sigma} f(a, \mathbf{x})\, \overline{\mathbf{c}}_a \quad . \tag{2.2}$$

Clearly, using a shorter codeword $\mathbf{c}_a$ for a given symbol a reduces the length of the encoding $\overline{\mathbf{c}}_\mathbf{x}$ in proportion to the frequency $f(a, \mathbf{x})$ of the symbol a in the original message $\mathbf{x}$. It is therefore desirable to use short codewords for frequent symbols in order

[1] This inequality was published by Kraft (1949) for prefix-free codes and was later proven for the broader class of uniquely decodable codes by McMillan (1956).

to improve compression. However, Kraft's inequality implies that using short codes for certain symbols must be compensated by using longer codes for other symbols, if the code is to remain uniquely decodable. A recipe for optimal assignment of code lengths under these constraints is given in the following section.

2.2.2 Information Sources and Entropy

An *information source* is a stochastic process which emits messages. We assume that this process is stationary and ergodic.[2] A source can be treated as a random variable X taking values in Σ^*, with a corresponding probability mass function (pmf) over messages $p : \Sigma^* \to [0, 1]$.

Let $\mathbf{c_x}$ denote the encoded representation of a message $\mathbf{x}$ according to some uniquely-decodable code. Shannon's source coding theorem (Shannon, 1948) gives a lower bound on the *expected* code length when encoding messages produced by the source:

$$\begin{aligned} \sum_{\mathbf{x}\in\Sigma^*} p(\mathbf{x})\, \overline{\mathbf{c}}_\mathbf{x} &= -\sum_{\mathbf{x}} p(\mathbf{x})\ \log_2 2^{-\overline{\mathbf{c}}_\mathbf{x}} \\ &\geq -\sum_{\mathbf{x}} p(\mathbf{x})\ \log_2 \frac{2^{-\overline{\mathbf{c}}_\mathbf{x}}}{\mathrm{Z}} \\ &\geq -\sum_{\mathbf{x}} p(\mathbf{x}) \log_2 p(\mathbf{x}) \end{aligned} \tag{2.3}$$

where Z is a normalization coefficient

$$\mathrm{Z} = \sum_{\mathbf{x}} 2^{-\overline{\mathbf{c}}_\mathbf{x}} \tag{2.4}$$

so that $2^{-\overline{\mathbf{c}}_\mathbf{x}}/\mathrm{Z}$ is a probability mass on $\mathbf{x} \in \Sigma^*$. The binary strings $\mathbf{c_x}$ can be considered codewords of a decipherable code over an alphabet with one symbol for each different message $\mathbf{x} \in \Sigma^*$.[3] Therefore, since the codewords $\mathbf{c_x}$ comprise a uniquely-decodable code, $\mathrm{Z} \leq 1$ by Kraft's inequality (Eq. 2.1), leading to the first inequality in Shannon's theorem (Eq. 2.3). The second inequality in Shannon's theorem is known as Gibbs' inequality, stating that $-\sum_i p_i \log p_i \leq -\sum_i p_i \log q_i$ if p and q are both probability mass functions over the same set of events.

Shannon's lower bound on expected code length is called the *entropy* or *uncertainty*

[2] Stationarity and ergodicity are required for estimating statistical properties of a stochastic process from a (sufficiently large) sample of data. They are stated here for completeness and are usually taken for granted in a typical machine learning setting, where we expect to learn from past observations.

[3] It is convenient to assume that the number of messages $\mathbf{x}$ with non-zero probability $p(\mathbf{x}) > 0$ is finite, although the derivation holds even if this is not the case.

of the information source:

$$H(X) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log_2 p(\mathbf{x}) \tag{2.5}$$

The entropy is non-negative and reaches its minimal value of zero if $p(\mathbf{x}) = 1$ for some $\mathbf{x}$. The entropy is therefore lowest if the information source is completely deterministic, that is, if it always outputs the same message. The entropy reaches its maximum when all messages have equal probability $p(\mathbf{x}) = \frac{1}{n}$, where n is the number of admissible messages $n = |\{\mathbf{x}, p(\mathbf{x}) > 0\}|$. In this case, the entropy of the source is $\log_2 n$. To illustrate, the entropy of a Bernoulli variable is depicted in Figure 2.2.

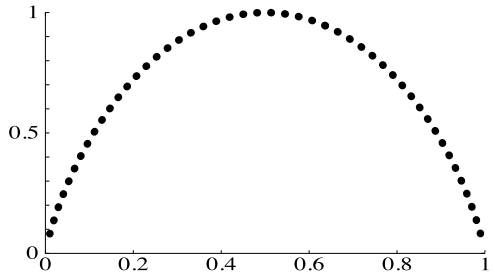


Figure 2.2: The entropy (in bits) of a Bernoulli variable (a biased coin flip), as a function of the variable's success probability.

Clearly, if the set of admissible messages is not bounded, as is generally the case, the entropy of the source can be infinite. The *entropy rate* $H'(X)$ of an information source X gives a lower bound on the average *per-symbol* code length required to encode a stream of data produced by the source:

$$\begin{aligned} H'(X) &= \lim_{i\to\infty} -\frac{1}{i} \sum_{\mathbf{x}\in\Sigma^i} p(\mathbf{x}) \log_2 p(\mathbf{x}) && (2.6) \\ &= \lim_{\overline{\mathbf{x}}\to\infty} E_{\mathbf{x}\sim P}\left[\frac{-\log_2 p(\mathbf{x})}{\overline{\mathbf{x}}}\right] && (2.7) \end{aligned}$$

For the class of stationary and ergodic processes, the following, equivalent formulation can be used:

$$\begin{aligned} H'(X) &= \lim_{i\to\infty} \sum_{\mathbf{x}\in\Sigma^i} p(\mathbf{x}) \sum_{a\in\Sigma} p(a|\mathbf{x}) \log_2 p(a|\mathbf{x}) && (2.8) \\ &= \lim_{\overline{\mathbf{x}}\to\infty} E_{\mathbf{x}\sim P}\left[-\sum_{a\in\Sigma} p(a|\mathbf{x}) \log_2 p(a|\mathbf{x})\right] && (2.9) \end{aligned}$$

The equivalence of both definitions follows from the chain rule of probability and the

stationarity of the information source (see Cover and Thomas, 1991, chap. 4). The entropy rate therefore equals the expected uncertainty of the next symbol in the data stream. Since the next symbol must take one of $|\Sigma|$ possible values, the entropy rate of the information source is between zero and $\log_2 |\Sigma|$ bits per symbol.

2.2.3 The Duality Between Compression and Prediction

Besides defining the limits of compressibility, Shannon's source coding theorem also gives a recipe by which optimal data compression can be achieved. This generally involves two steps:

- **Modeling:** To achieve optimal compression, the exact probability distribution P of the source must be known, i.e. the probability $p(\mathbf{x})$ of each possible message $\mathbf{x} \in \Sigma^*$.

- **Encoding:** Given a probability distribution over messages, a decipherable binary code should be constructed so that the length of the encoding $\mathbf{c_x}$ for each message $\mathbf{x} \in \Sigma^*$ is exactly $-\log_2 p(\mathbf{x})$ bits.

The second step is necessary to produce the final binary representation of the data, but is mostly irrelevant for our purposes. Suffice it to say that codes exist for which the length $\overline{\mathbf{c}}_\mathbf{x}$ of codewords $\mathbf{c_x}$ approaches the optimal value of $-\log_2 p(\mathbf{x})$, save for a small overhead due to the practical requirement that codewords have an integer length.[4]

The modeling problem is more difficult to tackle. The *true* source distribution P over all possible messages is typically unknown, so a compressor must learn an approximation of P in order to encode messages efficiently. A better approximation will, on average, lead to better compression, assuming that an "optimal" encoder is used. The performance of the compressor is therefore dominated by its modeling component – its ability to predict future outcomes given a sample of data produced by the information source. This implies that compression goes hand in hand with statistical modeling of certain types of processes – such that produce discrete sequences as output. Given that discrete sequences are the language of formal models of computation (Turing machines), this class of processes is indeed quite universal. This is perhaps the essence of philosophical arguments equating compression and learning.

[4] A classic "optimal" encoding is Huffman coding (Huffman, 1952). It turns out that Huffman coding is impractical to use if the number of codewords is very large, since all codewords are usually generated in advance. Arithmetic coding (Rissanen, 1976) overcomes this obstacle, in a sense producing a decipherable coding scheme over an alphabet containing all possible messages as symbols. Arithmetic coding has an overhead of at most 2 bits *per message* (see, for example, Witten et al., 1987). This overhead is negligible if messages are long.

Any probability distribution over messages can be used to construct a decipherable code (using, for example, arithmetic coding). The better this probability distribution reflects the true probabilities of messages, the more efficient the resulting code will be. It is important to note that the reverse is also true: Any decipherable code implies a probability distribution over messages with

$$p(\mathbf{x}) = \begin{cases} 2^{-\overline{\mathbf{c}}_{\mathbf{x}}} & \text{if } \mathbf{x} \text{ can be encoded by the code;} \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

The sum of $p(\mathbf{x})$ over all $\mathbf{x} \in \Sigma^*$ is less or equal to one due to Kraft's inequality (see Section 2.2.1). This distribution could therefore be used to create a code (e.g. using arithmetic coding), which would be (more or less) equivalent to the original code in terms of its compression performance. The implication is that any compression algorithm can be treated as a statistical model, even if the compressor is a black box which merely accepts messages $\mathbf{x}$ and outputs encodings $\mathbf{c}_{\mathbf{x}}$. Solving the compression problem implies solving the statistical modeling problem, *and* vice-versa.

2.2.4 Two-Part and One-Part Codes

Consider a general-purpose compressor with no a-priori assumptions about the data to be encoded. When a message $\mathbf{x}$ is to be compressed, the compressor infers a *statistical model* θ which captures the regularities found in the data. We assume that the model affords a probability mass function $p(\mathbf{y}|\theta)$ over messages $\mathbf{y} \in \Sigma^*$. A code can then be constructed by which the message $\mathbf{x}$ is encoded using no less than $-\log_2 p(\mathbf{x}|\theta)$ bits. We say that the model θ *compresses* $\mathbf{x}$ to $-\log_2 p(\mathbf{x}|\theta)$ bits. Ideally, $\mathbf{x}$ has a high probability according to θ, so that the number of bits needed to encode the message is minimized.

Two-part codes. In order to decode the message, the decoder must know the code by which the message $\mathbf{x}$ was encoded, or, equivalently, the probabilistic model θ from which the code can be (re)constructed. A *two-part code* works by first encoding the model θ and then encoding the message $\mathbf{x}$ using this model. The resulting code length of the two-part code is $\overline{\mathbf{c}}_\theta - \log_2 p(\mathbf{x}|\theta)$ bits, where $\mathbf{c}_\theta$ is a binary description of the model θ. The length of the two-part code could equivalently be written as

$$L(\mathbf{x}, \theta) = -\log_2 p(\theta) - \log_2 p(\mathbf{x}|\theta) \ , \tag{2.11}$$

where $p(\theta) = 2^{-\overline{\mathbf{c}}_\theta}$, i.e. $p(\theta)$ is the pmf over models which is implicitly defined by the model coding scheme (see Section 2.2.3). In other words, the encoding of the model in

two-part codes necessarily implies a prior over models $p(\theta)$.

Mixture-based codes. *One-part codes* do not explicitly include a model description in the compressed message. We describe two ways by which this can be achieved, beginning with codes based on *Bayesian mixtures*. Let Θ denote a model class – the set of all possible models θ which can potentially be inferred by some compressor. Now consider the discrete Bayesian mixture distribution over messages:

$$p(\mathbf{x}) = \sum_{\theta \in \Theta} p(\theta) p(\mathbf{x}|\theta) \quad . \tag{2.12}$$

If this pmf over messages is used to construct a code, there is no need to specify a model, assuming that the encoder and the decoder agree on the model class Θ and the model priors $p(\theta)$. This is because all models in the model class are essentially used at once. The mixture approach results in the following code length for a message $\mathbf{x}$:

$$L(\mathbf{x}, \Theta) = -\log_2 \left(\sum_{\theta \in \Theta} p(\theta) p(\mathbf{x}|\theta) \right) \quad . \tag{2.13}$$

This code is guaranteed to give better compression than a two-part code over the same model class, since

$$\sum_{\theta \in \Theta} p(\theta) p(\mathbf{x}|\theta) \geq \max_{\theta \in \Theta} \{p(\theta) p(\mathbf{x}|\theta)\} \quad , \tag{2.14}$$

and thus $L(\mathbf{x}, \Theta) \leq \min_{\theta \in \Theta} \{L(\mathbf{x}, \theta)\}$, provided that the prior $p(\theta)$ equals the pmf used for model encoding in the two-part code.

The problem with this approach is that the size of the model class Θ is typically unbounded, making it impossible to explicitly enumerate all possible models $\theta \in \Theta$. Note that the mixture equation (Eq. 2.12) should contain an integral instead of a sum in case that the model class is continuous. The mixture can be computed in closed form only for certain specific combinations of model classes and priors over models, but not in general.

Adaptive codes. The second class of one-part codes we consider are *adaptive codes*. An adaptive code uses a different model for encoding each consecutive symbol in a message. The encoding process starts with a "default" model, for example, a uniform distribution over all symbols. After encoding a symbol, the model is refined to better predict the symbols that follow, thus improving compression for the remainder of the

message. The encoded length of a message $\mathbf{x}$ using an adaptive code is

$$L^*(\mathbf{x}) = -\sum_{i=1}^{\overline{x}} \log_2 p(x_i|\mathbf{x}_1^{i-1}, \theta(\mathbf{x}_1^{i-1})) \tag{2.15}$$

bits, where $\theta(\mathbf{x}_1^{i-1})$ denotes the current model at time i, constructed from the input sequence $\mathbf{x}_1^{i-1}$. The length of the codeword used for encoding the symbol x_i is therefore $-\log_2 p(x_i|\mathbf{x}_1^{i-1}, \theta(\mathbf{x}_1^{i-1}))$ bits. Effectively, a different next-symbol codebook is used at each position i, corresponding to the "evolving" next-symbol probability distribution $\theta(\mathbf{x}_1^{i-1})$. A suitable encoder must be used for this scheme to be computationally efficient (e.g. arithmetic coding).

The decoder repeats the learning process, building its own version of the model as the message is decoded, thus reconstructing the code that was used for encoding at each symbol position. A major advantage compared to two-part codes is that adaptive codes require only a single pass over the data, making it possible to compress a stream of data in real time.

2.2.5 Universal Codes

Consider a situation in which a class of models Θ is available, and the task is to encode a sequence $\mathbf{x}$. Let $\hat{\theta}(\mathbf{x}) \in \Theta$ denote the model which compresses $\mathbf{x}$ the most, that is, the maximum likelihood model for which the probability $p(\mathbf{x}|\hat{\theta}(\mathbf{x}))$ of the sequence $\mathbf{x}$ is greatest. Compressing $\mathbf{x}$ using $\hat{\theta}(\mathbf{x})$ requires $-\log_2 p(\mathbf{x}|\hat{\theta}(\mathbf{x}))$ bits. This code length is optimal for the class of models Θ, but is in general too optimistic, since the identity of $\hat{\theta}(\mathbf{x})$ must also be communicated to the decoder. We now discuss this overhead.

A *universal code* relative to a model class Θ has the property that it compresses *any* sequence $\mathbf{x} \in \Sigma^*$ "almost" as well as the optimal model $\hat{\theta}(\mathbf{x})$ in the model class. More precisely, the worst-case difference in code length between a universal code and the best model in the model class increases sub-linearly with the length of the sequence $\mathbf{x}$. The relative performance of a universal code will be indistinguishable from the optimal code in the model class asymptotically, as the amount of data becomes large (provided that the data is not completely deterministic). Rissanen (1986) gives a precise non-asymptotic lower bound on this difference in the worst case, which turns out to be linearly related to the complexity of models available in the model class, measured in terms of the number of parameters. He also shows that codes exist that achieve this bound, and therefore guarantee a certain level of performance wrt the optimal model in the model class even for finite sequence lengths. The lower the complexity of the model class, the better this guarantee.

Universal codes play an important role in the minimum description length theory of inference (see Section 2.4.2). Simple two-part codes are universal, since only a finite code length is required to specify the model in a two-part code.[5] Codes based on mixtures over all models in the model class are clearly universal, since their performance is no worse than the shortest two-part code for the entire model class, up to a constant factor on account of possible differences in model priors. Certain adaptive codes are also universal (Rissanen, 1984). Moreover, adaptive compression algorithms exist that are proven to achieve Rissanens lower bound for certain conventional model classes (e.g. Willems et al., 1995, see also Section 3.4). The code length overhead incurred due to the fact that adaptive coding starts with a default, uninformed model, is comparable to the cost of separately encoding the model in two-part codes.

2.3 Overview of Lossless Compression Algorithms

Lossless data compression algorithms exploit regularities in the data to produce compressed representations from which the data can be reconstructed *exactly*. Lossy compression algorithms, on the other hand, may selectively discard some of the "detail" in the original information. Lossless compression methods have traditionally focused on text compression and on compression of text-encoded or binary data files, such as semi-structured documents, database files, computer programs, etc. Lossy compression is mainly used for compression of multimedia files, such as images, audio and video. In the following, we only discuss lossless data compression schemes.

Lossless data compression algorithms can generally be categorized into two groups of methods (cf. Witten et al., 1999b; Sayood, 2005), namely *dictionary techniques*, which capitalize on repetitions of subsequences, and *context-based methods*, which predict individual symbols based on the context of symbols around them. We note that most modern lossless compression algorithms are adaptive, capable of operating on a stream of data by incrementally training the compression models, and improving compression as the data is read and encoded.

2.3.1 Dictionary Methods

Dictionary methods aim to compress data by replacing recurrent subsequences (phrases) in the data with shorter descriptions, such as a pointer to a previous occurrence of the phrase. Repeated occurrences are identified in a single pass by building a dictionary of phrases found earlier in the input stream. The dictionary is built online, typically by

[5] We assume here that the particular model which results in the *shortest* two-part code is used in the compression scheme, which can in general be assured only if all models in the model class are tried.

adding phrases that extend the longest dictionary entry matching the current position in the input by one character. Many well-known lossless compressors and file formats are based on the "Lempel-Ziv" family of dictionary compression methods, which we briefly describe.

The LZ77 algorithm (Ziv and Lempel, 1977) works by replacing each repeated occurrence of a phrase with a reference to an earlier occurrence of the phrase within a fixed-length sliding window in the input stream (each reference is an offset and length pair). Characters that cannot be matched to any previous phrase are included as "literals" in the compressed stream. The standard DEFLATE algorithm, used in `gzip` and the `PNG` image format, is based on LZ77 and Huffman coding of the combined alphabet of back-references and literals.

The LZ78 algorithm (Ziv and Lempel, 1978) replaces repeated occurrences with references in the form of dictionary keys instead of pointers into a "sliding window" stream buffer. A variant of the LZ78 algorithm known as LZW (Welch, 1984) initializes the dictionary to the set of all input characters, which guarantees a match in the dictionary for any input and thus eliminates the need to include "literals" in the compressed stream. The LZW scheme is used in the `GIF` image format and the standard `compress` Unix utility.

2.3.2 Context-based Algorithms

In contrast to dictionary methods, context-based methods do not aggregate symbols into phrases. Instead, context-based methods use variable-length codes to encode individual symbols in the data stream. Codes are constructed in such a way that more probable symbols are represented with fewer bits, so that the expected code length is minimized (see also Section 2.2.3). Using entropy encoders, such as arithmetic coding (Rissanen, 1976; Witten et al., 1987), each individual symbol $a \in \Sigma$ can effectively be encoded with the "ideal" fractional number of bits given its probability, i.e. $-\log_2 p(a)$ bits.[6]

The crucial task in context-based methods is accurate estimation of probabilities of different symbols at each position in the input data stream. Symbol probabilities are assumed to depend on the *context* of symbols that precede the current position.

[6]The principle idea in arithmetic coding is to divide an interval into $|\Sigma|$ distinct sub-intervals which correspond to the possible symbols $a \in \Sigma$. The width of each sub-interval is proportional to the probability of the mapped symbol. To encode a symbol, the encoder simply transmits a number that falls within the correct sub-interval. The key observation is that this number can generally be transmitted with less precision for wider sub-intervals (i.e. for more probable symbols), while greater precision is required to distinguish shorter intervals corresponding to less probable symbols. This procedure is generalized to sequences using a recursion that starts with a unit interval and repeatedly sub-divides this interval with each consecutive symbol. The resulting code for the whole sequence is a number within the interval defined by the probability of the whole sequence of symbols. Various implementations of arithmetic coding allow sequential encoding and decoding of individual symbols in the sequence.

Algorithms generally differ in the way contexts are modeled. The leading context-based methods are prediction by partial matching (PPM; Cleary and Witten, 1984), dynamic Markov compression (DMC; Cormack and Horspool, 1987) and context tree weighting (CTW; Willems et al., 1995). All three algorithms fall into the category of variable-order Markov models. The PPM and CTW algorithms combine next-symbol predictions based on different contexts of $d = 0, 1, \ldots, n$ preceding symbols. These predictions are interpolated to strike a balance between model power (long contexts, i.e. large d) and stability of the gathered statistics (short contexts, i.e. small d). The DMC algorithm uses a different approach based on finite state machines (FSMs). The DMC algorithm incrementally constructs a FSM model of the information source with different next-symbol probabilities for each state in the FSM. Transitions between states are determined by the symbols leading up to the current position in the sequence. We describe these methods in greater detail in Chapter 3.

Although dictionary methods such as LZ77 and LZ78 are the "workhorse" algorithms for lossless data compression, their main advantages are speed and memory efficiency, while context-based methods generally exhibit better compression rates. The overall best compression results on large compression benchmarks are obtained by ensemble methods which blend many context-based sequential prediction models (see e.g. Hutter, 2006; Mahoney, 2009). The context-based models included in the mixture are usually optimized for different types of data.

2.3.3 Block Sorting

To conclude this section, we mention the Burrows Wheeler transform (BWT; Burrows and Wheeler, 1994), a reversible transformation used in data compression, which does not fit into any of the previous categories. Block sorting re-orders symbols in the original sequence in such a way that the transformed sequence tends to exhibit long runs of the same symbol, which are then easily compressed using simple techniques such as run length encoding. In the transformed sequence, symbols are effectively clustered together according to their context in the original sequence. BWT-based compression schemes such as `bzip2` generally achieve good compression rates. BWT has also been used extensively in bioinformatics problems such as sequence alignment (Li and Durbin, 2009; Langmead et al., 2009) and clustering of biological sequences (e.g. Yang et al., 2010).

2.4 Theoretical Relationships Between Compression and Learning

The information-theoretic foundations of data compression reveal a close relationship between compression and statistical inference. In this section, we illuminate the relationship from the opposite direction, with a review of theoretical schools of machine learning and statistical inference[7] which are grounded in information theory. The fundamental concepts and achievements of each of these theories are described. The section is concluded with a discussion and interpretation of these theoretical principles with respect to the contributions of this thesis, which characterizes the general rift between learning theory and practical applications involving actual compression algorithms for machine learning.

2.4.1 The Minimum Message Length Principle

The Minimum Message Length (MML) principle (Wallace and Boulton, 1968; Wallace, 2005) is an information-based measure by which models (i.e. hypotheses) can be compared given some observed data, that is, a sample of data $\mathbf{x}$, originating from some process which the competing hypotheses hope to explain. The principle states that the model θ^* producing the shortest two-part encoding of the model *and* the data is most likely to be correct:

$$\theta^* = \arg\min_{\theta \in \Theta} \{L(\mathbf{x}, \theta)\} \quad , \tag{2.16}$$

where $L(\mathbf{x}, \theta)$ is the length, in bits, of the two-part encoding of θ and $\mathbf{x}$:

$$L(\mathbf{x}, \theta) = -\log_2 p(\theta) - \log_2 p(\mathbf{x}|\theta) \quad . \tag{2.17}$$

The first term encodes the model θ, and the second term encodes the data $\mathbf{x}$ using a code that is optimal for the distribution specified by θ.

MML and Bayesian model selection. At first glance, the MML principle is just a restatement of the Bayesian model selection criterion. According to Bayes' theorem, the conditional probability of a model θ given evidence $\mathbf{x}$ is

$$p(\theta|\mathbf{x}) = \frac{p(\theta)p(\mathbf{x}|\theta)}{p(\mathbf{x})} \quad . \tag{2.18}$$

[7] Algorithmic information theory, which is also discussed in this section, is actually a branch of mathematics, with many diverse applications that transcend statistical inference. According to one of its founding fathers, Gregory Chaitin, it is "the result of putting Shannon's information theory and Turing's computability theory into a cocktail shaker and shaking vigorously." However, the basic ideas of the theory were proposed to tackle open problems in statistical inference (Solomonoff, 1960).

The posterior $p(\theta|\mathbf{x})$ is proportional to $p(\theta)p(\mathbf{x}|\theta)$, save for a term which is independent of θ, so the model minimizing the message length in Equation (2.17) is precisely the model with the highest posterior given the data $\mathbf{x}$. Despite this analogy, the information-theoretic perspective on model selection offered by the MML principle allows us to reason about the problem in ways that are not apparent in traditional Bayesian analysis.

Bias-variance tradeoff in MML. The MML criterion naturally balances the complexity of models considered and the degree to which we are able to fit the data using the models in the model class. Any one of the two terms in Equation (2.17), which determine the length of the two-part code $L(\mathbf{x}, \theta)$, can in general be made shorter at the expense of the other. For example, if we reduce the number of models in the model class Θ, the priors $p(\theta)$ of the remaining models will increase, leading to a reduction in the length of the first part of the code. By doing so, we may have removed models which yield the shortest encoding (i.e. the highest probability) for certain data points $\mathbf{x}$, thus possibly increasing the length of the shortest two-part code for such $\mathbf{x}$. The opposite effect results from extending the model class. In a sense, this property of two-part codes corresponds to the bias-variance tradeoff in statistics.

MML notions of randomness and success of learning. Another notion offered by the MML approach is that of randomness: If no two-part code can encode the data $\mathbf{x}$ with fewer bits than stating $\mathbf{x}$ explicitly, for example, in the way in which it was originally presented for analysis, then $\mathbf{x}$ is regarded as random, i.e. having no significant pattern which can be explained by any of the hypotheses in the class Θ. To put it another way: nothing can be reliably inferred from the available data. If the two-part message length $L(\mathbf{x}, \theta)$ in Equation (2.17) is to be less than $\overline{\mathbf{x}}$, then clearly, the model class Θ must be finite with priors $p(\theta)$ greater than zero. Furthermore, given some data $\mathbf{x}$, which we assume to be encoded in binary, the prior $p(\theta)$ of any acceptable model θ cannot possibly be less than $2^{-\overline{\mathbf{x}}}$. There are at most $2^{\overline{\mathbf{x}}}$ such models. When learning from data, the number of hypotheses that we are permitted to evaluate with any hope of success is thus dictated by the volume of data available. In information-theoretic terms, this makes perfect sense. The amount of information contained in the data $\mathbf{x}$ is at most equal to its length, in bits, and we cannot hope to make a reliable choice among competing hypotheses based on information that is less than the uncertainty of this choice.

The MML view on model parameters and priors. Much of the work in MML is concerned with the selection of an appropriate (sub)set of hypotheses Θ and devising

appropriate codes for models $\theta \in \Theta$. In principle, the selection of hypotheses is done in such a way that no two hypotheses in the subset are so similar that the available volume of data cannot be expected to distinguish reliably between them (Wallace and Dowe, 1999). One way to reduce or expand the set Θ of available hypotheses is simply to change the precision by which parameters of the models $\theta \in \Theta$ are stated, i.e. using a coarser or finer discretization of the parameter space. Indeed, the information-theoretic perspective offered by the MML principle allows us to usefully compare models with many parameters stated imprecisely against models with fewer parameters of greater precision. This is one of the most powerful features of MML.

The strict MML principle. Designing codes for hypotheses is equivalent to specifying a prior $p(\theta)$ over models $\theta \in \Theta$. Ideally, the encoding of hypotheses should minimize the expected message length $E_{\mathbf{x}}\left[\min_\theta L(\mathbf{x}, \theta)\right]$ (see Wallace, 2005, chap. 3). This requirement is known as the "strict" MML principle. The (strict) MML principle thus justifies certain choices of priors, which have been the Achilles' heel of Bayesian analysis. Unfortunately, devising strict MML codes is an NP-hard problem for all but the most trivial model classes (Farr and Wallace, 2002), forcing practitioners to resort to various approximations.

MML and Occam's Razor. The MML principle is often interpreted as a formalization of Occam's razor, that is, the notion that given two theories or explanations of the data, all other things being equal, the simpler one is to be preferred.[8] The MML principle quantizes this abstract notion of "simplicity" as the length of the resulting two-part code. A note of caution is required in connection with this interpretation (see, e.g., Domingos, 1999). As MML is not a method of inference, but rather a principle by which models can be compared, its choices are limited to hypotheses that are included in the model class Θ, and we can make the models in Θ as simple or as complex as we like. Moreover, any preference for less complex models within the class Θ should be stated explicitly in our choice of priors $p(\theta)$, which dictate the length required to encode θ. As it turns out, it is often both practical and intuitive to use priors that decrease with increasing complexity of hypotheses (cf. Needham and Dowe, 2001), thus actually producing a shorter code length for simpler hypotheses.

[8]The relationship to Occam's razor is also often found in references to the Minimum Description Length principle, a theory related to MML which we also review in the following.

2.4.2 The Minimum Description Length Principle

The Minimum Description Length (MDL) principle (Rissanen, 1978; Barron et al., 1998; Grünwald, 2005) adopts the same philosophy as Wallace's MML, that is, to evaluate models in terms of their ability to compress data. The traditional two-part MDL principle involves the use of two-part codes in a similar fashion as MML, which is perhaps why the two streams are often confused or considered equivalent. Baxter and Oliver (1994) give a detailed comparison of two-part codes used to evaluate hypotheses in MML and MDL, exposing certain differences between the two schools. While much of the work in the MML literature focuses on how to appropriately encode hypotheses using two-part codes, this problem is largely sidestepped with the use of one-part universal codes in the so-called "refined" MDL principle (Rissanen, 1996; Grünwald, 2005). It is this modern version of MDL that we describe in the following.

The refined MDL principle. Instead of selecting a particular model, the focus of MDL is to select the *model class* which is best suited for the given data $\mathbf{x}$. A model class would typically contain models of similar complexity, for example, models having the same parametric form. Each model class Θ is evaluated by compressing $\mathbf{x}$ using a code that is *universal* for Θ. Recall that a universal code for Θ compresses "almost as well" as the *best* model in Θ. In a sense, this means that the universal code "dominates" all models in Θ, since it is (asymptotically) as efficient as *any* model in Θ. On the other hand, the universal code inevitably has at least *some* overhead compared to the code length of $-\log_2 p(\mathbf{x}|\hat{\theta}(\mathbf{x}))$, achieved by the maximum likelihood model $\hat{\theta}(\mathbf{x}) \in \Theta$ for the particular data sequence $\mathbf{x}$, i.e. the model in the model class Θ which assigns to $\mathbf{x}$ the greatest probability $p(\mathbf{x}|\hat{\theta}(\mathbf{x}))$. The crucial observation is that this code length overhead is roughly proportional to the complexity of the model class (see Section 2.2.5). The code length overhead of a universal code compared to the best fitting model in Θ can therefore be interpreted as a complexity penalty on Θ.

The MDL principle selects the model class Θ^* exhibiting the best compression of the data using a universal code for the model class. If a specific model from the selected model class Θ^* is desired, e.g. for prediction of future data, MDL traditionally uses the maximum likelihood model $\hat{\theta}(\mathbf{x}) \in \Theta^*$ (Rissanen, 1989). Note that the maximum likelihood model $\hat{\theta}(\mathbf{x}) \in \Theta^*$ can safely be used, since it has already been established that the amount of data in $\mathbf{x}$ is sufficient for models in Θ^* not to overfit.

Minimax optimal universal codes. There are many ways to design a universal code for a given model class Θ. One option is to use two-part codes such as those used in MML, but this is neither the only, nor necessarily the best option in all circumstances.

It is often easier to devise a suitable one-part universal code. Among all universal codes, the MDL literature suggests the use of *minimax optimal* universal codes. If $L(\mathbf{x}, \Theta)$ is the length of $\mathbf{x}$ encoded using a universal code for Θ, the code is said to be minimax optimal if it minimizes:

$$\max_{\mathbf{x}\in\Sigma^*} \left\{ L(\mathbf{x}, \Theta) + \log_2 p(\mathbf{x}|\hat{\theta}(\mathbf{x})) \right\} \quad . \tag{2.19}$$

The objective is to minimize the largest (worst-case over all $\mathbf{x}$) code-length overhead compared to the best-fitting model $\hat{\theta}(\mathbf{x}) \in \Theta$. The minimax code-length overhead is known also as *minimax regret* under log loss. This is materially different from the "strict" MML principle, where the objective is to minimize the *expected* code length over all $\mathbf{x}$, which also requires a prior over $\mathbf{x}$ or an approximation thereof. In a sense, a minimax optimal universal code also ensures that all models, i.e. all possible data explanations contained in Θ, are treated on equal footing, specifically by requiring that the code-length overhead is not excessively large compared to any single model $\hat{\theta}(\cdot)$ in Θ.

Minimax optimality is uniquely achieved by a particular distribution known as the *normalized maximum likelihood* (NML) distribution (Shtarkov, 1987):

$$p_{\mathrm{nml}}(\mathbf{x}, \Theta) = \frac{p(\mathbf{x}|\hat{\theta}(\mathbf{x}))}{\sum_{\mathbf{y}\in\Sigma^*} p(\mathbf{y}|\hat{\theta}(\mathbf{y}))} \quad . \tag{2.20}$$

This distribution yields a universal code for Θ in which the code-length overhead compared to the best fitting model $\hat{\theta}(\mathbf{x})$ in Θ is exactly the same for all messages $\mathbf{x}$. This is shown by setting $L(\mathbf{x}, \Theta) = -\log_2 p_{nml}(\mathbf{x}, \Theta)$ in Equation (2.19), revealing that the regret equals the base-2 logarithm of the denominator in the NML distribution (Eq. 2.20), which is independent of $\mathbf{x}$. Roughly speaking, this overhead reflects the number of distinguishable probability distributions contained in Θ, and serves as a theoretically sound complexity penalty on Θ according to MDL.

Adaptive codes in MDL. While codes based on the NML distribution are preferred according to the MDL literature and play an important role in MDL theory, the NML distribution is impractical, or even impossible to compute for most model classes. Luckily, feasible universal codes exist with minimax performance guarantees that are only very slightly worse, typically within a constant of the minimax-optimal NML code. In particular, under certain regularity constraints on the model class and the data, this can be shown for adaptive one-part codes (Rissanen, 1984), for which the code length

is as follows:

$$L_{\mathrm{a}}(\mathbf{x}, \Theta) = \sum_{i=1}^{\overline{\mathbf{x}}} p(x_i | \mathbf{x}_1^{i-1}, \theta(\mathbf{x}_1^{i-1})) \ . \tag{2.21}$$

The model $\theta(\mathbf{x}_1^{i-1})$ from Θ which is used in the adaptive code should converge towards using the maximum likelihood model $\hat{\theta}(\mathbf{x}_1^{i-1})$ for the already-encoded part of the data, but it is not necessarily equal to $\hat{\theta}(\mathbf{x}_1^{i-1})$.

The use of adaptive codes to compare model classes is known as *prequential* or *predictive* MDL. Prequential MDL tells us to select the model class which minimizes the accumulated error, measured as logarithmic loss, resulting from sequentially predicting future outcomes after training on all the past outcomes. This embodiment of the MDL principle leads to a wrapper approach to model selection, which is directly comparable to popular cross-validation techniques. The procedure is similar to leave-one-out cross-validation, except that a model is trained only on data "preceding" the left-out datum. Prequential MDL is particularly suitable when the data is sequential in nature. If the data is not sequential, one may argue that the order in which data is processed will affect this model selection process. This may be the case, but the same objection could be made against the random choices involved in producing a k-fold cross-validation split.

MDL and Bayesianism. By using universal codes, MDL cleverly avoids the need to specify any subjective priors on models $\theta \in \Theta$, which are otherwise required in the development of two-part coding schemes and in Bayesian model selection. The universality of the code used to measure description length essentially guarantees consistence, that is, that given enough data, the MDL procedure will converge arbitrarily close to the "true" underlying model θ^* responsible for the data, provided that θ^* is within the space of hypotheses considered. Minimax optimal universal codes are maximally unbiased, in the sense that no hypothesis in the model class is *a priori* disadvantaged, as can be the case when priors on hypotheses are inappropriately chosen in Bayesian model selection. Grünwald (2005) points out that asymptotically (for large datasets), MDL based on the minimax optimal NML code becomes indistinguishable from Bayesian model selection using Jeffreys' prior – the prior distribution introduced by Jeffreys (1946) as the "least informative prior, to be used when no useful prior knowledge about the parameters is available".

2.4.3 Algorithmic Information Theory

Consider two long strings, the first beginning with `"000000000000000..."` and the second with `"010010110100101..."`. While the first string offers a reasonable short explanation – "a sequence of n zeros", there is no apparent short description for the second

string. Intuitively, we feel that the first string is somehow "less random". Around the same time, Solomonoff (1964), Kolmogorov (1965) and Chaitin (1966) proposed similar formalisms for measuring and reasoning about complexity, randomness, or "universal (im)probability", of a particular binary string. Although this quantification of randomness is deeply theoretical, we shall see that it is also fundamentally related to data compression and learning.

Algorithmic complexity. The Kolmogorov (or algorithmic) complexity of a binary string $\mathbf{x}$ is defined as the length of the *shortest* binary program $\mathbf{p}$ that produces the target string $\mathbf{x}$ as its output and terminates. This definition inevitably depends on the programming language in which the program $\mathbf{p}$ is written. Let T denote a machine which interprets the chosen language and $T(\mathbf{p})$ the output of T given the binary sequence (program) $\mathbf{p}$ as input. The Kolmogorov complexity of a string $\mathbf{x}$ under the language defined by T is

$$K_T(\mathbf{x}) = \min_{\mathbf{p}\in\{0,1\}^*} \{\overline{\mathbf{p}} \,:\, T(\mathbf{p}) = \mathbf{x}\} \quad . \tag{2.22}$$

It turns out that the definition of the Kolmogorov complexity $K_T(\mathbf{x})$ of a string $\mathbf{x}$ is not as sensitive to the choice of language (defined by T) as one might expect, provided that T is a universal Turing machine, so that any computable algorithm can be expressed in the language defined by T. When measuring the Kolmogorov complexity of a string $\mathbf{x}$ using two different *universal* reference machines T_1 and T_2, the difference is at most a constant (independent of $\mathbf{x}$), specifically the length of the program which makes T_1 emulate T_2, or vice-versa. This "compiler constant" becomes unimportant for sequences $\mathbf{x}$ which are sufficiently complex, that is, sequences with no trivially short description on either T_1 or T_2. This result, known as Salomonoff's invariance theorem, suffices to prove many important theoretical properties of Kolmogorov complexity without specifying a particular reference machine or description language. Consequently, the Kolmogorov complexity of a string $\mathbf{x}$ can be considered an inherent property of the string $\mathbf{x}$, independent of any description formalism (c.f. Grünwald and Vitányi, 2003). In the following, we sometimes write $K(\mathbf{x})$ if the choice of T is not important for the discussion, i.e. without providing any reference to a particular machine.

One important result of algorithmic information theory is that the algorithmic complexity $K(\mathbf{x})$ of a string $\mathbf{x}$ is in general *incomputable*. An upper bound can be determined by devising short "self-extracting" representations of the data in some chosen reference language, for example, by compressing the data using some general-purpose data compression program. However, there is in general no guarantee that all regularities that could be exploited to produce an even more compact representation of $\mathbf{x}$ have

been discovered. In terms of learning from data, this means that it is impossible to know whether we have successfully discovered all that is to be learned from a particular dataset. This is known as Chaitin's incompleteness theorem.

A string $\mathbf{x}$ is *incompressible* or *algorithmically random* if it has no representation, in the language defined by T, which would be shorter than stating $\mathbf{x}$ itself, therefore $K_T(\mathbf{x}) \geq \overline{\mathbf{x}}$. Such a string lacks any identifiable structure which could be usefully exploited to compress $\mathbf{x}$. It can be shown that most strings are incompressible (see, e.g. Li and Vitányi, 2008, section 2.2), and therefore, if we embrace the analogy between compression and learning, nothing can be reliably learned from most strings.

Algorithmic probability. The algorithmic probability $m(\mathbf{x})$ of a string $\mathbf{x}$ is the probability that a random binary program $\mathbf{p}$, when provided as input to a reference universal Turing machine T, produces $\mathbf{x}$ as its output and terminates:

$$m(\mathbf{x}) = \sum_{\mathbf{p}:T(\mathbf{p})=\mathbf{x}} \left(\frac{1}{2}\right)^{\overline{\mathbf{p}}} \quad . \tag{2.23}$$

The probability of each program $\mathbf{p}$ equals the probability of a particular outcome of $\overline{\mathbf{p}}$ sequential coin flips. It is required that the reference Turing machine T is *prefix-free*, that is, having a single binary input tape which it may only read from left to right. Consequently, the set of valid programs for T is prefix-free, and by Kraft's inequality (Eq. 2.1), $\sum m(\mathbf{x}) \leq 1$ over all sequences $\mathbf{x} \in \Sigma^*$. This makes $m(\mathbf{x})$ a valid probability measure. Due to the halting problem – since it is in general impossible to know whether a certain program $\mathbf{p}$ will halt when run on a given machine T – the set of all programs that output a particular sequence $\mathbf{x}$ cannot be enumerated. As a result, $m(\mathbf{x})$ is *incomputable* and can in general only be approximated. Searching for programs that generate $\mathbf{x}$ yields a lower bound on $m(\mathbf{x})$.

Let $M(\mathbf{x})$ denote the algorithmic probability of a sequence *beginning* with $\mathbf{x}$:

$$M(\mathbf{x}) = \sum_{\mathbf{y}\in\Sigma^*} m(\mathbf{xy}) \quad . \tag{2.24}$$

The ensuing conditional probability $M(a|\mathbf{x})$, that a particular symbol a follows $\mathbf{x}$, is then given by

$$M(a|\mathbf{x}) = \frac{M(\mathbf{x}a)}{M(\mathbf{x})} \quad . \tag{2.25}$$

Solomonoff (1978) showed that $M(a|\mathbf{x})$ has some remarkable properties for sequential prediction. With certainty for all sequences $\mathbf{x}$, $M(a|\mathbf{x}_1^i)$ approaches the *true* probability distribution $p(a|\mathbf{x}_1^i)$ as i increases. As a corollary, $M(a|\mathbf{x})$ is *universal* with respect to all

computable distributions. The total regret (accumulated error over the entire sequence $\mathbf{x}$) of sequential prediction based on $M(a|\mathbf{x})$, compared to the unknown *true* probability $p(a|\mathbf{x})$, or compared to the best possible computable approximation of $p(a|\mathbf{x})$, is bounded by $\mathcal{O}(K(p))$, a quantity which is proportional to the complexity $K(p)$ of the process generating the data. This agrees with the intuitive notion that $\mathcal{O}(K(p))$ "guesses" are required to identify a hypothesis with a complexity of $K(p)$. It can be shown that $\mathcal{O}(K(p))$ errors are in some sense unavoidable for most p, and consequently, no inference method can lead to significantly better results (Hutter, 2011). Hutter (2005, 2011) coins the term "universal artificial intelligence" to describe inductive inference based on algorithmic probability, and derives similar optimality and universality results for a range of learning problems including sequential decisions, classification, regression and reinforcement learning.

The definition of algorithmic probability and its use as a *universal prior* over strings embodies both Occam's razor and Epicurus' principle of multiple explanations. To illustrate this, we say that the program $\mathbf{p}$ generating a sequence $\mathbf{x}$ is an *explanation* of the sequence $\mathbf{x}$ or, alternatively, $\mathbf{p}$ is a *hypothesis* consistent with the sequence $\mathbf{x}$. In accordance with Occam's razor, sequences which can be explained by shorter programs have greater algorithmic probability. Similarly, the probability of a certain symbol a after the sequence $\mathbf{x}$ will be greater if this is consistent with simple hypotheses. On the other hand, the algorithmic probability $m(\mathbf{x})$ is non-zero for any computable string $\mathbf{x}$, as there exists by definition a program on T for computing any computable $\mathbf{x}$ (since T is universal). All consistent explanations of the string $\mathbf{x}$ contribute towards its probability $m(\mathbf{x})$, and no computable hypothesis is *a priori* rejected (Epicurus' principle).

The algorithmic probability $m(\mathbf{x})$ is the cumulative probability of randomly finding some program $\mathbf{p}$ which outputs the target string $\mathbf{x}$. It can be shown that this sum is dominated by the length of the *shortest* such program $\mathbf{p}$, for which $\overline{\mathbf{p}} = K(\mathbf{x})$. More precisely, for all $\mathbf{x}$, the algorithmic probability $m(\mathbf{x})$ is within a multiplicative constant of the probability $2^{-K(\mathbf{x})}$, obtained solely from the Kolmogorov complexity $K(\mathbf{x})$ of the sequence $\mathbf{x}$. This fundamental result in algorithmic information theory is known as Levin's coding theorem (Levin, 1974). Informally, this means that finding the one most simple explanation for $\mathbf{x}$ is "almost as good" as finding all possible explanations for $\mathbf{x}$. One interesting consequence is that if a string $\mathbf{x}$ is consistent with many complex hypotheses, then it must also have some short (simple) description (for details, see Li and Vitány, 1992).

Relation to coding theory. By definition, the Kolomogorov complexity $K(\mathbf{x})$ of a string $\mathbf{x}$ is a bound on how much the individual string $\mathbf{x}$ can be compressed in some

universal programming language. On the other hand, the entropy of an information source defines a lower bound on *average* compression, relative to an information source X, with an associated probability mass function over messages p. It can be shown that the two quantities are related (Grünwald and Vitányi, 2003). The *expected* (average) Kolmogorov complexity $K(\mathbf{x})$ of a message $\mathbf{x}$ is effectively no larger than the entropy of the information source $H(X)$, plus a term related to the "complexity" of the probability distribution over messages p associated with the source:

$$H(X) \leq \sum_{\mathbf{x} \in \Sigma^*} p(\mathbf{x}) K(\mathbf{x}) \leq H(X) + K(p) + O(1) \tag{2.26}$$

The last inequality holds up to a constant $\mathcal{O}(1)$ term. The complexity $K(p)$ of the distribution p is the length of a minimal program required to specify p. Intuitively, the $K(p)$ term can be considered as the cost of encoding p in addition to the particular message $\mathbf{x}$, and compensates for the assumption in the definition of Shannon's entropy, which is that both the sender and the receiver know the source distribution (given by p) in advance, before actually transmitting any data.

Grünwald and Vitányi (2003) provide a thorough review of various relationships and analogues between concepts and quantities in coding theory and their counterparts in algorithmic information theory.

2.4.4 Discussion

The theories reviewed in preceding sections depict an intricate relationship between data compression and machine learning. We conclude the review with a less formal discussion and some intuitive interpretations on how the different theories support the notion that data compression algorithms can, and should, be also regarded as capable methods of inductive inference.

Algorithmic information theory tells us that methods that are essentially optimal for prediction, classification, and other machine learning tasks would be possible if we could compute the algorithmic probability of a given string (see Solomonoff, 1964; Hutter, 2011). Results that are "almost as good" could be achieved by measuring the Kolmogorov complexity of strings (Levin, 1974; Li and Vitány, 1992). Much like algorithmic probability, Kolmogorov complexity is incomputable, so the ultimate compression algorithm – which would also be the ideal tool for learning from data – is unattainable. However, an upper bound on Kolmogorov complexity can be obtained by trying to find ways to compress the data. Developing algorithms that improve compression will lead to tighter approximations of Kolmogorov complexity, and it is reasonable to expect that using probability estimates derived from such methods should also improve results for

prediction, classification, and other machine learning applications.

The MML and MDL principles essentially advocate the same – that the success of learning from data can be measured by the ability to compress the data. This compressibility criterion also guards against overfitting. MML and MDL were developed for model selection, i.e. for choosing among competing hypotheses. In this thesis, we are not concerned with model selection. Instead, we trust that the extensive existing research in data compression has already selected appropriate models, or rather, classes of models, for our application domain. In fact, the criteria by which data compression algorithms have been optimized for decades are exactly the same criteria to be used for model selection in learning applications according to MML and MDL. The refined MDL principle makes this argument more precise by suggesting the use of universal codes for evaluating a model class. The compression algorithms used in most of our applications are in fact universal with respect to the relatively broad class of variable-order Markov models with limited memory. All of the compression algorithms that we consider are adaptive, i.e. based on sequential model updates and prediction. The branch of MDL theory known as prequential MDL deals specifically with the use of adaptive codes for measuring the description length relative to a model class, and generally implies that adaptive compression performance is a useful criterion for model selection.

The learning theories described in previous sections provide different recipes for "optimal" model selection and learning. These include the strict MML principle, minimax optimality in MDL, and the use of algorithmic probability for inductive inference. Interestingly, all of these recipes are impractical or even impossible to compute exactly in general applications, so practitioners are usually forced to resort to approximations. We do not claim that the work in this thesis falls under the MDL or MML principle, or theories based on Kolmogorov complexity, since none of the methods considered rigorously follow from any of the ideal measures proposed by these theories. As Grünwald (2005) points out, "*some authors use MDL as a broad umbrella term for all types of inductive inference based on data compression*". Indeed, the prevailing use of universal data compression models brings our work in-line with the definition of MDL given by Barron et al. (1998), as "*general inference based on universal models*". On the other hand, many practitioners ground their approach in algorithmic information theory (e.g. Li et al., 2004; Keogh et al., 2004, see also Section 2.5), by using compression as an approximation to the idealized Kolmogorov complexity. Such reasoning is appealing but unavoidably quite loose, since algorithmic information theory tells us that it is in general impossible to determine even the quality of approximation of Kolmogorov complexity.

The theoretical relationships between compression and learning have also spurred interest in the artificial intelligence community to advance research in data compres-

sion. Realizing that advances in text compression algorithms could foster progress in artificial intelligence, Hutter (2006) created the Human knowledge compression contest to "*encourage development of intelligent compressors/programs as a path to artificial general intelligence*". A similar argument is made by Mahoney (2009) in his Large text compression benchmark initiative.

To summarize, the learning theories discussed in this section uniformly support the idea of using data compression algorithms for inductive inference. Furthermore, data compression algorithms have traditionally been optimized for modeling certain types of sequential data, particularly natural-language text, which is otherwise non-trivial to represent using fixed-length feature vectors used in standard machine learning methods. This is the principal motivation for our work.

2.5 Text Mining Applications of Compression Methods

Many existing studies consider the use of data compression methods in machine learning. Compression algorithms are particularly well-suited for learning from sequential data, and have therefore most commonly been used for text mining (see, e.g. Witten, 2004) and for mining biological sequences (reviewed by Giancarlo et al., 2009). Other domains include mining music, as well as time series data after transformation into symbolic form (see, e.g. Keogh et al., 2007). In the following review we focus on text mining applications.

2.5.1 Compression-based Distance Measures

We first review schemes in which data compression algorithms are used as a subroutine to measure the similarity between two objects. The theoretical foundations underlying most of these schemes are laid out by Bennett et al. (1998), who propose a theoretical *information distance* between two strings $\mathbf{x}$ and $\mathbf{y}$. The distance is based on the notion of Kolmogorov complexity, and is defined as the length of the shortest program to compute $\mathbf{x}$ from $\mathbf{y}$ and vice-versa. This measure is incomputable, but is theoretically appealing, primarily for its universality – if two objects are similar by any reasonable measure, they are also close according to this ideal information distance. The theoretical properties of this ideal information distance have motivated several practical distance measures, in which the Kolmogorov complexity of a string $\mathbf{x}$ is approximated as the compressed size $L(\mathbf{x})$ of the string with the use of off-the-shelf compression programs (e.g. Li et al., 2004; Krasnogor and Pelta, 2004; Keogh et al., 2007). Unlike the theoretical information distance of Bennett et al. (1998), approximations using off-the-shelf compressors are *not* metrics, since the distance between a string and itself is typically not zero, and the

triangle inequality is typically also not satisfied. Such approximations of information distance are therefore more accurately described as measures of *dis-similarity*.

To compare two strings $\mathbf{x}$ and $\mathbf{y}$, compression-based dis-similarity measures essentially compare the compressed sizes $L(\mathbf{x})$ and $L(\mathbf{y})$, obtained by compressing each of the strings $\mathbf{x}$ and $\mathbf{y}$ in isolation, and the compressed size $L(\mathbf{xy})$ of the concatenation of the input strings $\mathbf{xy}$. Intuitively, compressing the concatenation of two similar strings will be more effective than compressing each string separately, that is, $L(\mathbf{xy}) < L(\mathbf{x}) + L(\mathbf{y})$, since the compressor will exploit patterns found in one string to improve compression of the other. We list various distance measures proposed by different authors in Table 2.1. As noted by Sculley and Brodley (2006), most of the measures in Table 2.1 are very similar, and can be written in a form that differs only in the normalizing factor[9]. When used for classification, the performance of these measures is quite similar (Sculley and Brodley, 2006; Ferragina et al., 2007). For textual data, the compression algorithm used to measure compression distance is most often LZ77 (`gzip`), and should be assumed in the following review unless stated otherwise.

Some authors argue that the use of off-the-shelf compression programs to measure compression distances implies "parameter free" data mining (Keogh et al., 2004) and "feature free" modeling of data (Li et al., 2004). However, this line of thought can be substantiated only as long as the compression programs which are used are treated as a "black box". Under the hood, however, compression programs construct models of data which are themselves parameter-laden. Furthermore, these models imply the use of certain feature spaces, which can also be made explicit (cf. Sculley and Brodley, 2006).

Compression-based distance measures using off-the-shelf compressors do not scale well to large problems. For example, in classification schemes relying on compression distance measures, the computational cost required to classify a single document is typically proportional to the size of the training data. For large-scale applications, the underlying models used in the compression scheme should be extracted and used explicitly, rather than implicitly through the use of an external compression program. Explicit use of compression models provides for greater flexibility, for example, a model can be trained, then stored, and applied at a later time, or manipulated to change its characteristics as required for specific applications. Moreover, compression models should not be treated as a "black box" if we wish to gain an understanding of the patterns that are detected by the scheme, and devise further improvements.

Nevertheless, methods which simply run an external compression program and read the length of the compressed files have merit for rapid prototyping of different com-

[9] An exception is the relative-entropy based measure by Benedetto et al. (2002). Note that this measure is also not posed as an approximation of Kolmogorov complexity.

Measure	Definition
Shared information distance (SID) (Chen et al., 1999; Li et al., 2001)	$d(\mathbf{x},\mathbf{y}) = 1 - \frac{L(\mathbf{x})-L(\mathbf{x}\|\mathbf{y})}{L(\mathbf{xy})}$
Normalized compression distance (NCD) (Li et al., 2004)	$d(\mathbf{x},\mathbf{y}) = \frac{L(\mathbf{xy})-\min\{L(\mathbf{x}),L(\mathbf{y})\}}{\max\{L(\mathbf{x}),L(\mathbf{y})\}}$
Compression-based dissimilarity (CD) (Keogh et al., 2004, 2007)	$d(\mathbf{x},\mathbf{y}) = \frac{L(\mathbf{xy})}{L(\mathbf{x})+L(\mathbf{y})}$
Universal similarity metric (USM)[a] (Li et al., 2003; Krasnogor and Pelta, 2004)	$d(\mathbf{x},\mathbf{y}) = \frac{\max\{L(\mathbf{xy})-L(\mathbf{x}),L(\mathbf{yx})-L(\mathbf{y})\}}{\max\{L(\mathbf{x}),L(\mathbf{y})\}}$
Compression-based cosine (CosS) (Sculley et al., 2006)	$d(\mathbf{x},\mathbf{y}) = \frac{L(\mathbf{x})+L(\mathbf{y})-L(\mathbf{xy})}{\sqrt{L(\mathbf{x})L(\mathbf{y})}}$
Benedetto's relative entropy measure (RE)[b] (Benedetto et al., 2002)	$d(\mathbf{x},\mathbf{y}) = \frac{L(\mathbf{x},\mathbf{y}_1^k)-L(\mathbf{y},\mathbf{y}_1^k)}{k}$

Table 2.1: Various distance or dis-similarity measures proposed in the literature, all of which are based on the use of off-the-shelf data compression programs. In the dis-similarity expressions, $L(\mathbf{x})$ denotes the copressed length of a sequence $\mathbf{x}$, and $L(\mathbf{y}|\mathbf{x}) = L(\mathbf{xy}) - L(\mathbf{x})$. The sequence $\mathbf{y}_1^k$ is a prefix of length k of the sequence $\mathbf{y}$.

a) Despite its name, the USM measure is a dis-similarity.

b) The parameter k in the relative-entropy based measure by Benedetto et al. (2002) defines the length of the prefix of $\mathbf{y}$ used when computing the measure, and typically ranges up to 16K.

pression schemes, and for capitalizing on data modeling techniques and optimizations implemented in sophisticated data compression programs. The success of such methods certainly provides cause for further investigation of the underlying compression models.

2.5.2 Clustering

The utility of compression-based distance measures has often been validated in clustering applications, using one of the many available clustering methods that rely on pairwise distances. A typical application is hierarchical clustering of documents translated into different languages, for which several authors report being able to extract language trees that agree with formal linguistic classification of languages into families (see, e.g. Benedetto et al., 2002; Li et al., 2004; Keogh et al., 2004; Cilibrasi and Vitanyi, 2005). Cilibrasi and Vitanyi (2005) experiment with clustering literary texts, where texts from the same authors are shown to cluster together. An innovative application is described by Bennett et al. (2003), who use hierarchical clustering based on a particular compression distance to reconstruct the evolution of 33 variations of a chain

letter circulating for over 15 years.

2.5.3 Classification

Supervised classification has traditionally been the hallmark application in text mining, and compression-based methods for classification have been studied extensively.

Classification Using Compression Distance

Nearest-neighbor classification based on the compression distance between documents is considered for authorship attribution by Benedetto et al. (2002). Lambers and Veenman (2009) also investigate authorship attribution, using the compression distance between a document and other training documents as features for linear discriminant analysis. Such features are found to be superior when compared to various other feature spaces. Chen et al. (2004) describe the use of compression distance for the one-class problem of software plagiarism detection. They use a special compressor designed to compress program source code for this task.

Sculley and Brodley (2006) study the problem of identifying users based on Unix shell session logs. They evaluate the classification performance of nearest-neighbor classifiers using several different definitions of compression distance in combination with various compression algorithms. The PPM compression scheme is found to outperform LZ77 and LZW compressors, and is comparable to the best of several different explicit feature space representations which they also considered, within the same instance-based classification framework.

Marton et al. (2005) compare two different compression-based classification schemes using a variant of Benedetto's relative entropy distance with $k = \overline{\mathbf{y}}$ (see Table 2.1). They consider a standard nearest-neighbor scheme, where the document being classified is compared to each individual document in the training data, as well as a scheme where the target document is compared to the concatenated sequence of all training documents of a class. The second scheme, which effectively builds a compression model of each class, outperforms the instance-based approach. They experiment with topic classification and authorship attribution. Among several compressors which are considered, the RAR program, which is based on a variant of PPM, outperforms dictionary-based and block sorting compression methods. It is also demonstrated that PPM successfully exploits patterns spanning across more than one word.

Model-based Approaches for Classification

In the model-based approach, one compression model is trained from the training data pertaining to each class. When classifying a document, the model that best compresses the target document determines the classification outcome. In this way, compression models are effectively used as Bayesian classifiers to estimate the class-conditional probability of the target document.

Model-based text classification using the PPM compression scheme was first investigated by Frank et al. (2000). They find that the compression-based method is inferior to support vector machines (SVM) and roughly on par with naive Bayes for categorization by topic. They conclude that compression models are not competitive for text categorization, since the correct category may be indicated by the presence of only a few relevant keywords, which are unlikely to be detected by compression methods. We note that the Reuters-21578 dataset which they use for their experiments is known to be specific in that it contains only a handful of highly relevant features (Bekkerman et al., 2003), which is not the case for most text classification tasks. Teahan and Harper (2003) enhance performance of PPM for assignment of Reuters topics to documents, primarily due to an improved PPM scheme, and report results competitive to SVM classifiers.

Teahan (2000) considers the use of PPM models for a number of different classification tasks. They find that compression models outperform other methods for dialect identification and authorship attribution, and also exhibit solid performance for genre classification. Khmelev and Teahan (2003) compare several compression schemes for authorship attribution. In the comparison, PPM-based compression models outperform other character-based methods, as well as word-based SVM classifiers. Pavelec et al. (2009) also find that PPM models perform well for authorship attribution when compared to SVM classifiers. Their SVM classifier uses specialized features intended to capture writing style.

Delany and Bridge (2006), and Zhou et al. (2011) investigate the use of PPM models for spam filtering, following our own work (Bratko and Filipič, 2005; Bratko et al., 2006a). Smets et al. (2008) report encouraging results when using PPM models for automated vandalism detection in Wikipedia – to determine whether an edit of an article is legitimate or malicious. DMC models are later considered for Wikipedia vandalism detection by Itakura and Clarke (2009). Nishida et al. (2011) use LZ77 models for model-based classification of short messages, or "tweets", with promising results.

Peng et al. (2004) propose augmenting a word-based naive Bayes classifier with statistical language models to account for word dependencies. They also consider training

the language models on characters instead of words, resulting in a method that is effectively very similar the use of PPM models for classification. In experiments on a number of categorization tasks, they find that the character-based approach often outperforms word-based models, and reaches or improves previously published results for the majority of the tasks considered. In a separate study, the same authors show that character-level language models perform very well for classification of Asian language documents (Peng et al., 2003).

Based on the reviewed body of work, we conjecture that compression-based classifiers are particularly effective for problems where subtle, stylistic features are relevant, as well as for multi-lingual and adversarial settings.

2.5.4 The Noisy Channel Problem

Compression models have also been applied to variants of the "noisy channel problem" (Brown et al., 1990) – the problem of restoring text which has been transformed by some noisy stochastic process. The use of compression models in this setting has been initially studied in the context of automatic text correction. Other versions of the noisy channel problem include restoration of information which has been "stripped" from the text, such as word delimiters, or tags that denote entities of a particular type of interest.

Teahan et al. (1998) describe the use of PPM compression models to correct optical character recognition (OCR) errors in English text. A set of transformations describing different types of errors and their probabilities is used to construct possible corrections, and to calculate the probability of the associated errors. The likelihood of a possible corrected source text is determined by a PPM model trained on a reference corpus of English text. Using the Viterbi algorithm (Viterbi, 1967), the space of possible source texts can be searched efficiently to find the solution maximizing the joint probability of the corrected text *and* the errors which transform the corrected text into the observed OCR text. The same method is successfully applied to segmentation of Chinese text into words (Teahan et al., 2000), where instead of OCR errors, the intent is to "correct" the omissions of word delimiters in written Chinese.

Witten et al. (1999a) describe a compression-based method for automatic extraction of named entities and other structured data from text (for example, geographic locations, dates, phone numbers, etc). In their approach, the idea is to replace entities we might wish to extract from the original text with special marker characters. The text is then compressed using a PPM model trained from pre-annotated texts, in which all relevant entities have also been stripped and replaced by markers. The entities extracted from the original text are compressed separately, using a PPM model which is trained on known examples of such entities. An entity is successfully extracted whenever this

scheme improves overall compression compared to compressing the unaltered original text. Using the Viterbi algorithm, the entities can be extracted efficiently in what is effectively a single pass of beam search over the unannotated input text.

2.5.5 Compressibility as a Measure of Complexity

In some cases, the compressibility of a document, or individual parts of a document, can also be a relevant feature of the text, by providing a measure of text complexity. For example, Dalkilic et al. (2006) differentiate between authentic and computer-generated texts using a set of features obtained by compressing documents with different compression programs at various parameter settings. This can be useful, for example, to identify computer-generated websites created for the sole purpose of hosting advertisements. Seaward and Matwin (2009) use `gzip` to measure the complexity of various "fingerprint" sequences extracted from document passages, for example, binary sequences signaling the occurrence of particular parts of speech. The complexity of these sequences is useful for identifying variations in style indicating, for example, plagiarized passages in the text.

Chapter 3

Adaptive Context-based Compression Models

This chapter describes the data compression algorithms that are used or adapted for machine learning purposes in later developments. We only consider compression algorithms in which the source modeling and encoding steps are clearly separated. Specifically, we are interested in adaptive context-based compression models, which explicitly compute sequential probabilities of symbols, and are capable of online adaptation of their prediction models (see Sections 2.2.4 and 2.3.2). Efficient incremental model updates are essential for some of the problems considered in this thesis, such as the cross-entropy reduction sampling method for instance selection (Chapter 4) and online spam filtering (Chapter 5).

The problem of encoding, that is, transforming symbols with assigned probabilities into binary codes, is irrelevant for our purposes. We therefore focus only on the source modeling aspect of context-based compression algorithms.

3.1 Zero-order Estimators

Consider a sequence $\mathbf{x} = \mathbf{x}_1^n = x_1 \dots x_n \in \Sigma^*$ of independent observations (symbols) over a finite alphabet Σ. A zero-order estimator sequentially determines the probability of the next symbol x_i after observing $\mathbf{x}_1^{i-1}$. The estimated probability of the whole sequence is then

$$p_e(\mathbf{x}) = \prod_{i=1}^{\overline{\mathbf{x}}} p_e(x_i; \mathbf{x}_1^{i-1}) \ , \tag{3.1}$$

where $p_e(x_i; \mathbf{x}_1^{i-1})$ is the estimated probability of x_i after observing $\mathbf{x}_1^{i-1}$. Note that the individual symbols are assumed to be independent by the zero-order estimator, i.e. the

notation $p_e(a; \mathbf{x})$ is used merely to suggest that statistics from $\mathbf{x}$ are used to infer the zero-order probabilities of each symbol $a \in \Sigma$, and not probabilistic dependence among consecutive symbols. When using an estimator for sequential prediction, or as a basis for arithmetic coding in data compression, care must be taken to ensure that $p_e(a; \mathbf{x})$ is non-zero for all $a \in \Sigma$ and $\mathbf{x} \in \Sigma^*$, even if a does not appear in $\mathbf{x}$.

Effect of symbol permutations. A zero-order estimator is used as a subroutine in later developments. It is sometimes desirable that the estimator is insensitive to permutations. For any such estimator, $p_e(\mathbf{x})$ only depends on the frequencies of symbols in $\mathbf{x}$, and not on the order in which they appear. When training is performed incrementally from multiple training sequences, the same estimator is obtained irrespective of the order in which training examples are presented.

3.1.1 Additive Smoothing

A classical approach is to artificially increment the number of occurrences of each symbol in the training sequence by some small constant β, yielding the following zero-order estimator:

$$p_e(a; \mathbf{x}) = \frac{f(a, \mathbf{x}) + \beta}{\overline{\mathbf{x}} + \beta|\Sigma|} \ . \tag{3.2}$$

This family of estimators includes the popular Laplace law of succession ($\beta = 1$), as well as the Krichevsky-Trofimov estimator ($\beta = \frac{1}{2}$), for which redundancy guarantees are known that are essentially optimal for binary alphabets (Catoni, 2004).

3.1.2 Escape Methods

The "escape method" family of estimators originates from the PPM compression scheme (Cleary and Witten, 1984). Escape methods explicitly predict the event that some previously unseen symbol will occur. This is called an *escape event* or an *escape symbol*. The escape symbol is used in place of all unseen symbols $a \notin \Lambda(\mathbf{x})$ and essentially accumulates their probability:

$$p_e(Esc; \mathbf{x}) = \sum_{a \notin \Lambda(\mathbf{x})} p_e(a; \mathbf{x}) \ . \tag{3.3}$$

It is important to note that the escape probability $p_e(Esc; \mathbf{x})$ is typically estimated based only on symbols that actually occur in $\mathbf{x}$, and does not directly depend on the size of the unused part of the alphabet $\Sigma \setminus \Lambda(\mathbf{x})$. For zero-order estimators, the escape

probability is distributed uniformly among all zero-frequency symbols:

$$p_e(a;\mathbf{x}) = p_e(Esc;\mathbf{x})\frac{1}{|\Sigma \setminus \Lambda(\mathbf{x})|} \quad \text{for } a \notin \Lambda(\mathbf{x}). \tag{3.4}$$

In the general case, however, the escape probability can be distributed among zero-frequency symbols non-uniformly, that is, according to some prior distribution.

Escape Method D

The escape method D estimator (Howard, 1993) is a permutation-insensitive member of the "escape method" family, which has been shown to achieve good performance in text compression (Teahan, 1995; Tjalkens et al., 1993).

This estimator discounts the frequency of all symbols that have occurred in the training data by $\frac{1}{2}$ occurrence and uses the gained probability mass for the escape probability:

$$p_e(Esc;\mathbf{x}) = \frac{\frac{1}{2}|\Lambda(\mathbf{x})|}{\overline{\mathbf{x}}} \ . \tag{3.5}$$

The next-symbol probability estimates (by Eq. 3.4 and 3.5) are:

$$p_e(a;\mathbf{x}) = \begin{cases} \frac{f(a,\mathbf{x})-\frac{1}{2}}{\overline{\mathbf{x}}} & \text{if } a \in \Lambda(\mathbf{x}); \\ \frac{\frac{1}{2}|\Lambda(\mathbf{x})|}{\overline{\mathbf{x}}}\ \frac{1}{|\Sigma\setminus\Lambda(\mathbf{x})|} & \text{if } a \notin \Lambda(\mathbf{x}) \text{ and } \overline{\mathbf{x}} > 0; \\ \frac{1}{|\Sigma|} & \text{if } \overline{\mathbf{x}} = 0. \end{cases} \tag{3.6}$$

Examples of a binary escape method D estimator applied to three specific strings are given in Figure 3.1. Note that the first two example sequences are both permutations of the same symbols and receive the same probability. The third string, which contains only one type of symbol, receives a higher probability – it is "easier to compress".

		example 1					$p(\mathbf{x}) \approx 0.002$	example 2					$p(\mathbf{x}) \approx 0.002$	example 3					$p(\mathbf{x}) \approx 0.123$
position in sequence	i	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
symbol in sequence	x_i	**a**	**a**	**b**	**b**	**b**	**b**	**b**	**a**	**b**	**b**	**a**	**b**	**a**	**a**	**a**	**a**	**a**	**a**
discounted symb. freq.	$f(x_i,\mathbf{x}_1^{i-1})$-½	/	1-½	/	1-½	2-½	3-½	/	/	1-½	2-½	1-½	3-½	/	1-½	2-½	3-½	4-½	5-½
escape probability	$\frac{½\ \Lambda(\mathbf{x}_1^{i-1})}{(i-1)}$	1	$\frac{½}{1}$	$\frac{½}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	1	$\frac{½}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	1	$\frac{½}{1}$	$\frac{½}{2}$	$\frac{½}{3}$	$\frac{½}{4}$	$\frac{½}{5}$
num. of zero-freq. symb.	$\lvert\Sigma\rvert$-$\Lambda(\mathbf{x}_1^{i-1})$	2	1	1	0	0	0	2	1	0	0	0	0	2	1	1	1	1	1
next-symbol probability	$p(x_i;\mathbf{x}_1^{i-1})$	$\frac{1}{2}$	$\frac{1-½}{1}$	$\frac{½}{2}$	$\frac{1-½}{3}$	$\frac{2-½}{4}$	$\frac{3-½}{5}$	$\frac{1}{2}$	$\frac{½}{1}$	$\frac{1-½}{2}$	$\frac{2-½}{3}$	$\frac{1-½}{4}$	$\frac{3-½}{5}$	$\frac{1}{2}$	$\frac{1-½}{1}$	$\frac{2-½}{2}$	$\frac{3-½}{3}$	$\frac{4-½}{4}$	$\frac{5-½}{5}$

Figure 3.1: Escape method D estimator applied to three different sequences using a binary alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

We may compute the full sequence probability by grouping positions in $\mathbf{x}$ that correspond to first occurrences of symbols and positions that correspond to subsequent occurrences of "*already seen*" symbols:

$$p_e(\mathbf{x}) = \frac{1}{|\Sigma|} \prod_{x_i \notin \Lambda(\mathbf{x}_1^{i-1})}^{i=2..\overline{\mathbf{x}}} \frac{\frac{1}{2}|\Lambda(\mathbf{x}_1^{i-1})|}{(i-1)|\Sigma \setminus \Lambda(\mathbf{x}_1^{i-1})|} \prod_{x_i \in \Lambda(\mathbf{x}_1^{i-1})}^{i=2..\overline{\mathbf{x}}} \frac{f(x_i, \mathbf{x}_1^{i-1}) - \frac{1}{2}}{(i-1)} \quad . \tag{3.7}$$

The first factor in the equation is the probability of the first symbol in the sequence, which is always $|\Sigma|^{-1}$. The sequence probability can now be rewritten so that $p_e(\mathbf{x})$ is expressed in terms of occurrence counts of symbols in $\mathbf{x}$:

$$p_e(\mathbf{x}) = \frac{2\,\Gamma(|\Lambda(\mathbf{x})|)}{\Gamma(\overline{\mathbf{x}})} \frac{\Gamma(|\Sigma \setminus \Lambda(\mathbf{x})| + 1)}{\Gamma(|\Sigma| + 1)} \prod_{a \in \Lambda(\mathbf{x})} \frac{\Gamma\big(f(a, \mathbf{x}) - \frac{1}{2}\big)}{2\sqrt{\pi}} \quad . \tag{3.8}$$

It is clear from Equation (3.8) that the estimator is insensitive to the order of symbols in the sequence, which is important in some of our later applications, particularly for the cross-entropy reduction sampling method described in Chapter 4.

3.2 Variable-order Markov Models

We now consider higher-order models in which the next-symbol probability distribution possibly depends on previous symbols in the sequence. When modeling dependencies between consecutive symbols in a sequence, the most common simplifying assumption is that the originating source is Markov with a limited memory of k preceding symbols. Each symbol x_i in a sequence $\mathbf{x}$ is therefore assumed to depend only on the preceding k symbols $\mathbf{x}_{i-k}^{i-1}$ (the *prediction context*):

$$p(\mathbf{x}) = \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_1^{i-1}) \approx \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_{i-k}^{i-1}) \quad . \tag{3.9}$$

The number of context symbols k is referred to as the *order* of the Markov model. If the order k is fixed for every context, the model is called a *Markov chain* of order k. Higher-order models have the potential to better approximate the characteristics of a complex source. However, since the number of possible contexts increases exponentially with context length, accurate parameter estimates are hard to obtain. Note that an order-k Markov chain generally requires $(|\Sigma| - 1)|\Sigma|^k$ parameters.

Context-based compression algorithms employ different strategies to tackle this parameter estimation problem. The common ground to all such algorithms is that the complexity of the model is adapted to the amount of training data. In some prediction

contexts the amount of training data might be sufficient to support a more complex model, and predictions are made based on a long context of preceding symbols (i.e. high order k). In other contexts, more reliable lower-order statistics may be used for the prediction. The class of algorithms capable of adapting the order of the prediction model from one position to the next are called *variable-order Markov models*. In the following, we describe three popular variable-order Markov models originally developed for data compression.

3.3 The Prediction by Partial Matching Algorithm

The prediction by partial matching (PPM) algorithm (Cleary and Witten, 1984) has set the standard for lossless text compression since its introduction almost three decades ago (cf. Cleary and Teahan, 1997; Sayood, 2005). The PPM algorithm is essentially a back-off smoothing technique for finite-order Markov models. The algorithm is similar to certain n-gram language models such as absolute discounting (Ney et al., 1994). While language models were originally developed for word-level modeling of natural language, compression models have traditionally been optimized for smaller alphabets that accommodate letters, byte-codes or binary symbols.

It is convenient to assume that an order-k PPM model stores a table of all contexts (up to length k) that occur anywhere in the training text. For each such context, the frequency counts of symbols that immediately follow is maintained. When predicting the next symbol x_i in a sequence $\mathbf{x}$, its context of preceding symbols $\mathbf{x}_{i-k}^{i-1}$ is matched against the stored statistics. If no match is found, indicating that the context $\mathbf{x}_{i-k}^{i-1}$ does not appear in the training text, the length of the context is reduced until a context $\mathbf{x}_{i-l}^{i-1}$ is found with non-zero statistics (note that $l \leq k$). If the target symbol x_i is found following the context $\mathbf{x}_{i-l}^{i-1}$ in the training text, its probability is estimated according to an escape method estimator, trained from the sequence of individual symbols that follow occurrences of the context $\mathbf{x}_{i-l}^{i-1}$ in the training text. If, however, x_i does not appear after $\mathbf{x}_{i-l}^{i-1}$ in the training data, an *escape symbol* is transmitted by the PPM compression scheme. Recall that the escape probability estimates the probability of a *zero-frequency* symbol, given the history of observations in the prediction context $\mathbf{x}_{i-l}^{i-1}$. The escape probability is distributed among symbols *not* seen in the current prediction context according to a lower-order model, that is, according to statistics for the shorter context $\mathbf{x}_{i-l+1}^{i-1}$. The procedure is applied recursively until all symbols receive a non-zero probability. If necessary, a default model of order -1 is used, which always predicts a uniform distribution among all possible symbols. This procedure is depicted in Figure 3.2.

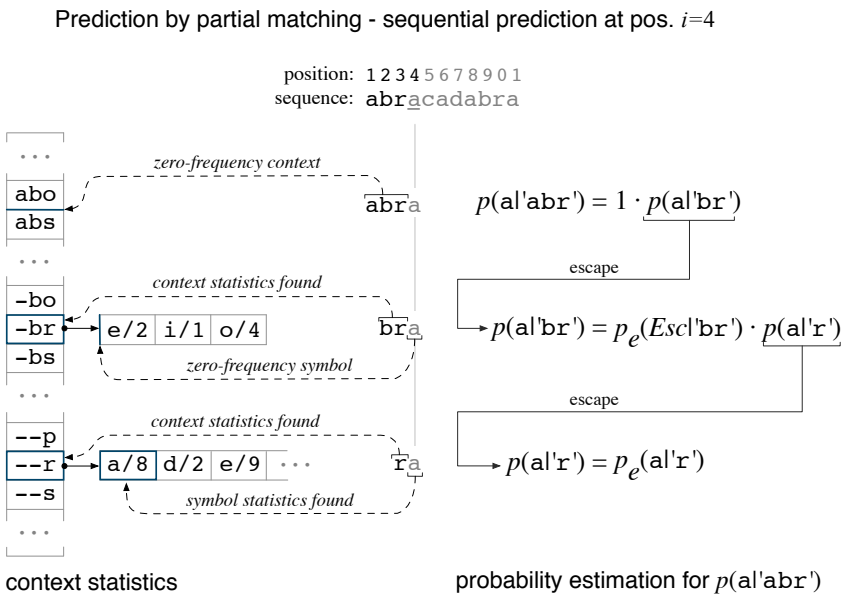


Figure 3.2: A schematic example of PPM probability estimation. The PPM context statistics data structure is depicted on the left. The probability estimation procedure is depicted on the right. The estimation procedure involves two escape events after which the algorithm backs-off to a shorter context.

Given the training text $\mathbf{s}$, the probability that the symbol $x_i = a$ follows the sequence $\mathbf{x}_1^{i-1}$ is:

$$p(a|\mathbf{x}_{i-k}^{i-1}) = \begin{cases} p_e(a;\mathbf{s}) & \text{if } k = 0 \text{ (zero-order);} \\ p_e(a;\mathbf{s}\|\mathbf{x}_{i-k}^{i-1}) & \text{if } \mathbf{x}_{i-k}^{i-1} \cdot a \text{ occurs in } \mathbf{s}; \\ p_e(Esc;\mathbf{s}\|\mathbf{x}_{i-k}^{i-1})\; p(a|\mathbf{x}_{i-k+1}^{i-1}) & \text{otherwise.} \end{cases} \tag{3.10}$$

Here, $\mathbf{s}||\mathbf{x}_{i-k}^{i-1}$ denotes the sequence of individual symbols from the training text $\mathbf{s}$ which are immediately preceded by the prediction context $\mathbf{x}_{i-k}^{i-1}$, while $\mathbf{x}_{i-k}^{i-1} \cdot a$ denotes the concatenation of $\mathbf{x}_{i-k}^{i-1}$ and a.

An adaptive PPM compressor starts with an empty model which defaults to the uniform distribution among all possible symbols. After each symbol is encoded, the algorithm updates its statistics for contexts of the current symbol. For an adaptive model, the training text equals the sequence of symbols leading up to the current position, which translates to $\mathbf{s} = \mathbf{x}_1^{i-1}$ in Equation (3.10).

3.3.1 The Exclusion Principle

When a zero-frequency symbol is encountered in some prediction context, the encoder emits an escape symbol and backs-off to a lower-order context. After receiving an escape

symbol, the decoder knows that the symbol that follows is *not* among the set of symbols with non-zero statistics in the higher-order context – hence the escape. Therefore, when predictions are made from a lower-order context following an escape, symbols with non-zero statistics in the higher order context (the context that generated the escape) can safely be excluded from the alphabet. The exclusion principle always increases the probability of the non-excluded symbols, and may increase or decrease the probability of an escape event for the lower-order prediction following an escape.

3.3.2 Implementations and Extensions

Many versions of the PPM algorithm exist, differing mainly in the way the escape probability is estimated. In later experiments, we use escape method D, described in the previous section, due to its invariance with respect to the order in which training data is presented and its known compression performance (Teahan, 1995; Teahan and Cleary, 1996).

The time and space complexity of PPM implementations is generally $\mathcal{O}(nd)$, where d is the order of the PPM model and n is the length of the training sequence and/or the sequence of symbols being predicted. The memory requirements can be reduced to $\mathcal{O}(n)$, irrespective of d, by using the suffix tree data structure (Weiner, 1973). A gain in speed can be achieved using update exclusion (Moffat, 1990), the practice of updating statistics of lower order contexts only if they are visited after an escape, or if no higher-order context matches the training text. This tactic brings the time complexity close to $\mathcal{O}(n)$ and has, perhaps surprisingly, also been shown to improve compression performance in many cases (Teahan, 1995).

Extensions of the PPM algorithm include an unbounded context-length version (Cleary and Teahan, 1997) and more sophisticated, context-based escape estimation (Shkarin, 2002).

3.4 The Context Tree Weighting Algorithm

The context tree weighting (CTW) algorithm (Willems et al., 1995) blends the predictions of an entire family of sequential prediction models known as tree source models, a class of variable-order Markov models which we describe in the following.

3.4.1 Tree Source Models

A context set $\mathcal{K} \subseteq \Sigma^*$ is a set of sequences which is suffix-free and complete, that is, no sequence in $\mathcal{K}$ is a suffix of another sequence in $\mathcal{K}$, and no sequence in Σ^* can be

added to $\mathcal{K}$ without breaking this property. A context set $\mathcal{K}$ is naturally represented with a tree structure so that each sequence in $\mathcal{K}$ maps to a leaf in the corresponding context tree, as depicted in Figure 3.3. Note that a context tree always has a constant branching factor of $|\Sigma|$. The order of a context set $\mathcal{K}$ is defined as

$$d(\mathcal{K}) = \max_{\mathbf{v} \in \mathcal{K}} \overline{\mathbf{v}} \ , \tag{3.11}$$

which equals the length of the longest path leading from the root node to a leaf in the corresponding context tree.

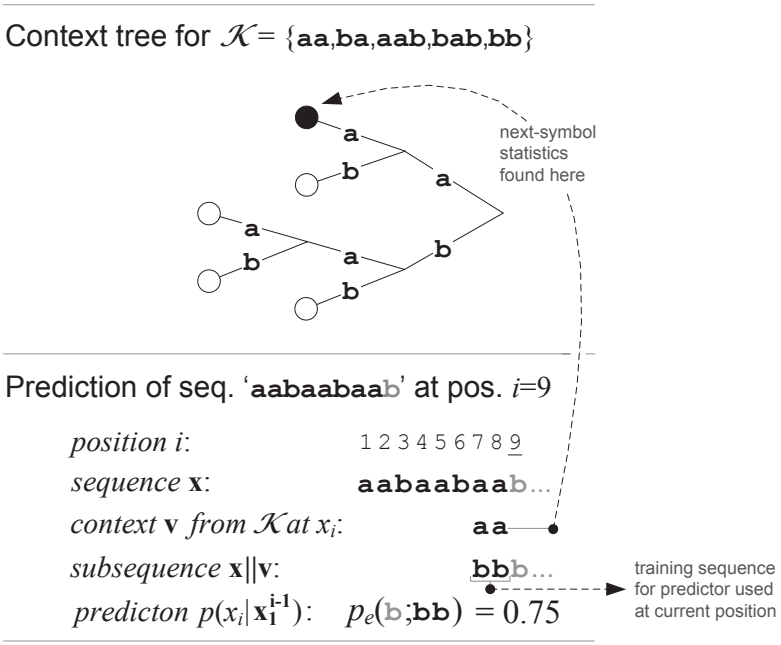


Figure 3.3: A binary alphabet context tree for the context set $\mathcal{K} = \{\mathtt{aa},\mathtt{ba},\mathtt{aab},\mathtt{bab},\mathtt{bb}\}$. The location of the predictor used at position 9 of the sequence aabaabaab... is also sketched. The sequence of individual symbols following the current prediction context aa in the preceding part of the sequence, i.e. aabaabaa‖aa = bb, serves as training data for the zero-order estimator $p_e(\cdot)$ used for the prediction context aa.

A tree source model based on $\mathcal{K}$ is a prediction model with distinct zero-order probability estimators for each leaf in the context tree specified by $\mathcal{K}$. When predicting the next symbol x_i in a sequence $\mathbf{x}$, the immediately preceding symbols $x_{i-1}, x_{i-2} \ldots$ are examined to determine a path in the context tree. This path is followed until a leaf is reached, and the corresponding estimator is used to output the prediction for $p(x_i; \mathbf{x}_1^{i-1})$ (see Figure 3.3). For example, a context tree with a single root node represents a zero-order estimator ($\mathcal{K} = \{\epsilon\}$), while a full order-$d$ context tree represents an order-d Markov chain ($\mathcal{K} = \Sigma^d$).

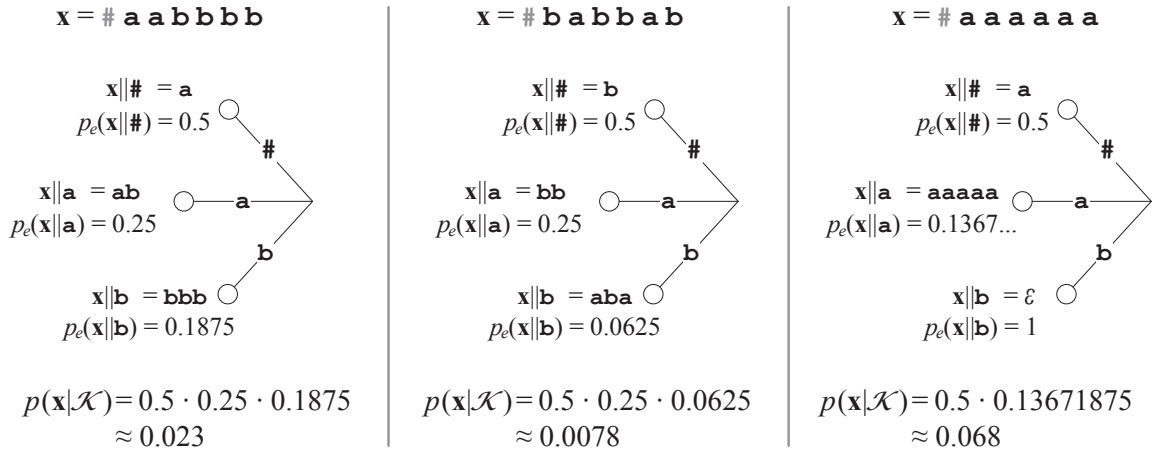


Figure 3.4: Examples of an order-1 tree source model corresponding to the context set $\mathcal{K} = \{\mathtt{a},\mathtt{b},\mathtt{\#}\}$, applied to three different sequences $\mathbf{x} \in \Sigma^*$ with $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

Each leaf in the context tree corresponds to a context $\mathbf{v} \in \mathcal{K}$, and the joint probability of $\mathbf{x}$ can be decomposed according to these contexts:

$$p(\mathbf{x}|\mathcal{K}) = \prod_{\mathbf{v}\in\mathcal{K}} p_e(\mathbf{x}\|\mathbf{v}) \quad . \tag{3.12}$$

We use $\mathbf{x}\|\mathbf{v}$ to denote the sequence of individual symbols which immediately follow occurrences of $\mathbf{v}$ in $\mathbf{x}$. An independent zero-order estimator is used for each prediction context $\mathbf{v} \in \mathcal{K}$.

If $\mathcal{K} \neq \{\epsilon\}$, handling the initial symbols in $\mathbf{x}$ poses a slight problem in that no suitable prediction context exists in $\mathcal{K}$. This is solved by prepending a special symbol '#' $\notin \Sigma$ at the beginning of a sequence and extending the context set $\mathcal{K}$ accordingly. This is depicted in the example in Figure 3.4, in which a simple tree source model is applied to three specific strings. The example sequences are the same as those used in Figure 3.1 for a zero-order escape method D estimator. Unlike the zero-order estimator, the tree source model assigns different probabilities to the first two sequences, which is due to its ability to model first-order dependencies.

3.4.2 The Context Tree Mixture

The CTW method is a sequential prediction algorithm for modeling tree sources with unknown structures. The probability assigned to a sequence $\mathbf{x}$ by an order-d CTW model is obtained by mixing the predictions of *all* tree source models with a maximum

order of d:[1]

$$p_w(\mathbf{x}) = \sum_{\mathcal{K}} \pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K}) \quad . \tag{3.13}$$

A careful choice of priors $\pi(\mathcal{K})$ over tree source structures $\mathcal{K}$ provides for an efficient recursive calculation of this mixture. The probability can be computed recursively along branches of the full order-d context tree, starting at the root ϵ

$$p_w(\mathbf{x}) = p_w(\mathbf{x}; \epsilon) \tag{3.14}$$

and using the following recursive definition of $p_w(\mathbf{x}; \mathbf{v})$:

$$p_w(\mathbf{x}; \mathbf{v}) = \begin{cases} p_e(\mathbf{x}\|\mathbf{v}) & \text{if } \overline{\mathbf{v}} = d; \\ \alpha\, p_e(\mathbf{x}\|\mathbf{v}) + (1-\alpha) \prod_{a\in\Sigma} p_w(\mathbf{x}; a\mathbf{v}) & \text{if } \overline{\mathbf{v}} < d. \end{cases} \tag{3.15}$$

Here, $\alpha \in (0..1)$ is a weighting parameter which ultimately defines the priors $\pi(\mathcal{K})$. The mixture at $\mathbf{v}$ is computed by weighting model structures which contain the context $\mathbf{v}$ and those that contain contexts of which $\mathbf{v}$ is a suffix (and therefore do not contain the context $\mathbf{v}$ itself). In terms of the context tree structure, the weighting is over trees that contain a leaf at the path defined by $\mathbf{v}$, and trees for which the path defined by $\mathbf{v}$ leads to an internal node. The $p_e(\mathbf{x}\|\mathbf{v})$ term in Equation (3.15) corresponds to the former case, and the $\prod_{a\in\Sigma} p_w(\mathbf{x}; a\mathbf{v})$ term corresponds to the latter.

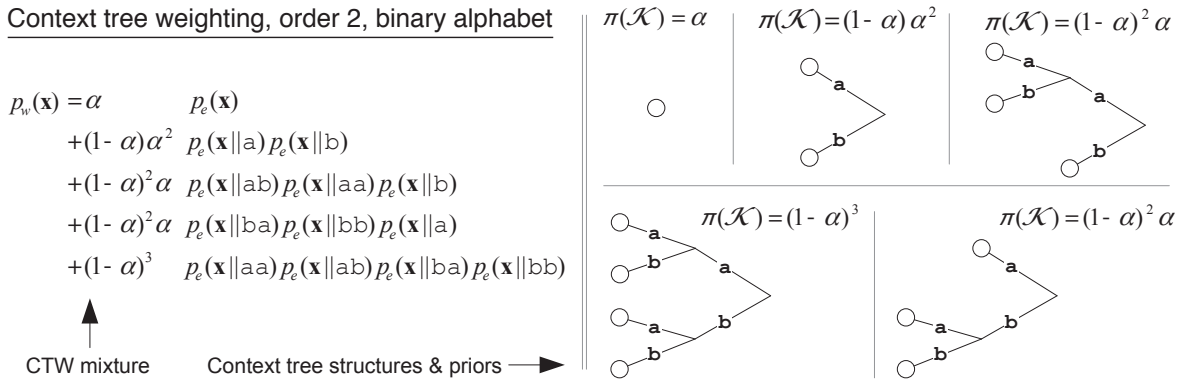


Figure 3.5: CTW priors over context tree structures: The mixture for predicting a binary sequence $\mathbf{x}$ using an order-2 CTW model is shown on the left. All binary context trees of depth ≤ 2 and their corresponding priors $\pi(\mathcal{K})$ in the CTW mixture are depicted on the right.

The recursive definition of $p_w(\mathbf{x})$ for an order-2 CTW model with a binary alphabet is unrolled in Figure 3.5. Each of the resulting terms corresponds to one of the context

[1] Although the CTW method can be made to model tree sources of unbounded depth efficiently (Willems, 1998), this case is not considered here.

tree structures $\mathcal{K}$ included in this particular mixture, and defines the prior $\pi(\mathcal{K})$, which is displayed in the figure next to the corresponding context tree.

Observe that $p_w(\mathbf{x}; \mathbf{v})$ is computed using only $\mathbf{x}\|\mathbf{v}$, that is, using only the sequence of symbols in $\mathbf{x}$ which are preceded by $\mathbf{v}$. When using zero-order estimators that are insensitive to permutations, this sequence can be evaluated in any order to produce $p_w(\mathbf{x}; \mathbf{v})$. Expanding the context tree at node $\mathbf{v}$ further partitions $\mathbf{x}\|\mathbf{v}$ into $\mathbf{x}\|a\mathbf{v}$, for $a \in \Sigma$. These subsequences are evaluated independently and then multiplied to produce the joint probability $\prod_{a\in\Sigma} p_w(\mathbf{x}; a\mathbf{v})$.

Figure 3.6 shows the computation of an order-1 CTW mixture for three specific strings for $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\alpha = 0.5$. The example sequences are the same as those used in Figure 3.1 (for a zero-order estimator) and Figure 3.4 (for an order-1 tree source model). In this particular case, the CTW mixture is the average of the two simpler estimators.

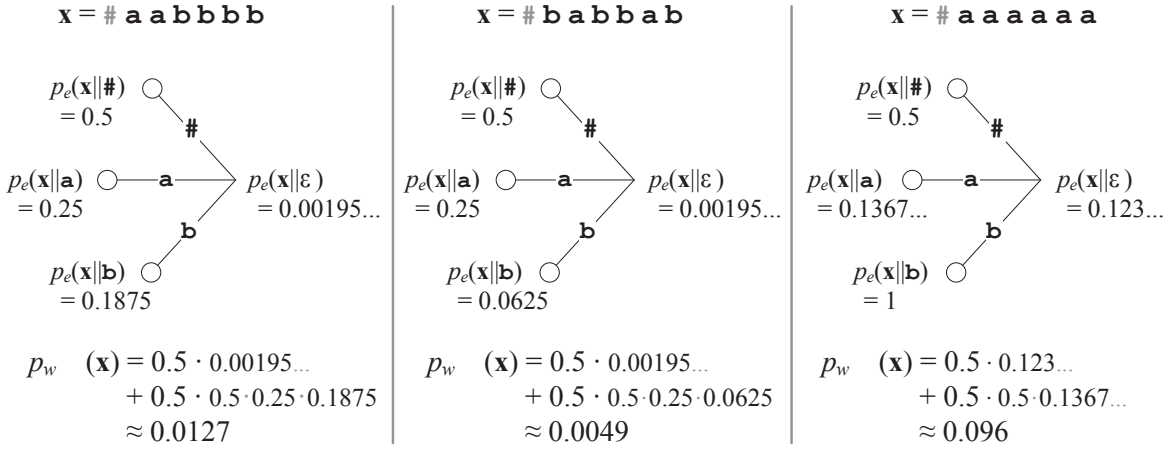


Figure 3.6: The computation of an order-1 CTW mixture for three different sequences $\mathbf{x} \in \Sigma^*$ with $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\alpha = 0.5$.

3.4.3 Priors over Context Tree Structures

The recursive CTW computation in Equation (3.15) produces a mixture over tree source models with priors decreasing exponentially with the number of nodes in the context tree. Let us assume that the structure of the source generating $\mathbf{x}$ corresponds to the context set $\ddot{\mathcal{K}}$, and that the length of any context in $\ddot{\mathcal{K}}$ is less than d. The prior $\pi(\ddot{\mathcal{K}})$ of the *true* model structure in the CTW mixture is then:

$$\pi(\ddot{\mathcal{K}}) = (1-\alpha)^{\frac{|\ddot{\mathcal{K}}|-1}{|\Sigma|-1}} \alpha^{|\ddot{\mathcal{K}}|} \quad . \tag{3.16}$$

We wish to maximize $\pi(\ddot{\mathcal{K}})$, that is, we wish to assign a large weight to the *true* model structure $\ddot{\mathcal{K}}$. Taking the logarithm of the expression in Eq. (3.16) and multiplying by $-\frac{|\Sigma|-1}{|\ddot{\mathcal{K}}||\Sigma|-1}$ gives

$$-\frac{|\ddot{\mathcal{K}}|-1}{|\ddot{\mathcal{K}}||\Sigma|-1}\log(1-\alpha) - \frac{|\ddot{\mathcal{K}}||\Sigma|-|\ddot{\mathcal{K}}|}{|\ddot{\mathcal{K}}||\Sigma|-1}\log(\alpha) \quad . \tag{3.17}$$

This expression should be minimized in order to maximize $\pi(\ddot{\mathcal{K}})$, since the log function is non-decreasing and $-\frac{|\Sigma|-1}{|\ddot{\mathcal{K}}||\Sigma|-1}$ is always negative. Furthermore, the expression is a cross-entropy, which is minimized by

$$\alpha = \frac{|\ddot{\mathcal{K}}||\Sigma|-|\ddot{\mathcal{K}}|}{|\ddot{\mathcal{K}}||\Sigma|-1} \qquad \text{and} \qquad 1-\alpha = \frac{|\ddot{\mathcal{K}}|-1}{|\ddot{\mathcal{K}}||\Sigma|-1} \quad . \tag{3.18}$$

This value for α assigns the largest weight (prior) to the *true* model $\ddot{\mathcal{K}}$. Since $\ddot{\mathcal{K}}$ is not known in advance, a good substitute is $\alpha = \frac{|\Sigma|-1}{|\Sigma|}$, which is due to the fact that $\ddot{\mathcal{K}} \gg 1$ for all but the most trivial target models. For details of this derivation, see (Tjalkens et al., 1993).

3.4.4 Efficiency and Other Properties

To compute $p_w(\mathbf{x})$, we only need to visit those nodes of the full order-d context tree which represent contexts that actually occur in $\mathbf{x}$. For any context $\mathbf{v}$ which does not occur in $\mathbf{x}$, $\mathbf{x}\|\mathbf{v} = \epsilon$ and $p_e(\mathbf{x}\|\mathbf{v}) = 1$ by definition. Since the number of different subsequences of length less or equal to d in $\mathbf{x}$ is limited by $d\overline{\mathbf{x}}$, a straight-forward implementation has time and space complexity of $\mathcal{O}(d\overline{\mathbf{x}})$. By using a suffix tree data structure, the time and space complexity of CTW can be further reduced to $\mathcal{O}(2\overline{\mathbf{x}})$ (Willems, 1998).

The CTW algorithm may also be applied efficiently for online prediction. This can be seen from Equation (3.15) by observing that $p_w(\mathbf{x};\mathbf{v}) = p_w(\mathbf{x}a;\mathbf{v})$ if $\mathbf{v}$ is not a suffix of $\mathbf{x}$, that is, if $\mathbf{v}$ is not a current prediction context for $p_w(a|\mathbf{x})$. In total, there are only d values of $p_w(\cdot)$ which need to be updated in order to compute $p_w(\mathbf{x}a)$ from $p_w(\mathbf{x})$. These updates require constant time and the time complexity for predicting $p_w(a|\mathbf{x})$ is thus bounded by $\mathcal{O}(d)$. It should be noted that $p_e(\mathbf{x}\|\mathbf{v})$ and $p_w(\mathbf{x};\mathbf{v})$ must be maintained for all prediction contexts $\mathbf{v}$ which occur in $\mathbf{x}$ to facilitate this computation.

The CTW algorithm lends itself well to theoretical analysis, with favorable performance guarantees both in the asymptote and for finite sequence lengths. For a thorough review of context tree weighting, including its theoretical properties and extensions, we refer the interested reader to (Willems et al., 1995; Willems, 1998; Tjalkens et al., 1993).

3.5 The Dynamic Markov Compression Algorithm

The dynamic Markov compression (DMC) algorithm (Cormack and Horspool, 1987) models an information source as a finite state machines (FSM). The algorithm is designed for modeling binary sequences. A probability distribution over symbols is associated with each state in the FSM and is used to predict the next binary digit emitted by the source. The algorithm begins in a predefined initial state and moves to the next state after each digit in the sequence. An example FSM structure, corresponding to an order-1 binary Markov model, is shown in the left side of Figure 3.7. Each state S in the FSM has two outbound transitions, one for each binary symbol. These transitions are equipped with frequency counts, from which next-symbol probabilities for the state S are calculated (as relative frequencies).

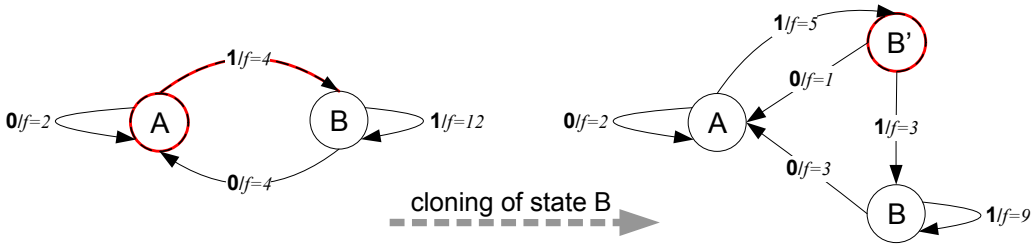


Figure 3.7: An example of DMC's state cloning operation. Transitions between states are annotated with the symbol that triggers the transition and a counter of the number of times the transition has been taken. The active state and transition are highlighted. The left hand side shows the model when state A is active and the observed symbol is '1'. This triggers the cloning of state B and a state transition to the new state B', as shown on the right hand side of the figure. The transition frequencies (visit counts) before and after the cloning operation are also shown.

The adaptive DMC algorithm increments the frequency counts of transitions whenever a transition fires (once after each symbol in the sequence). The structure of the state machine may also be built incrementally, by using a special state *cloning* operation. Specifically, as soon as the algorithm finds that a transition from some state A to some other state B in the FSM is used often, the target state of the transition may be cloned. Figure 3.7 depicts this cloning operation, in which a new state B' is spawned. The new state B' has a single inbound transition, which replaces the former transition from A to B. All outbound transitions of B are copied to B'.

After cloning, the statistics associated with the cloned state B are distributed among B' and B in proportion to the number of times state B was reached from state A, relative to the number of times state B was reached from other states (again, refer to Figure 3.7). Two parameters control the state cloning operation in DMC. These are the

minimal frequencies of B' and B after cloning. Both of the new states must exceed this minimal frequency that is required for stable probability estimates in order to trigger the cloning operation. At each position in the sequence only one state needs to be considered for cloning: The target state of the transition in the FSM that is triggered by the current symbol.

The cloning of state B allows the FSM to maintain separate statistics for situations when state B is reached from state A, and when it is reached from other states. Without loss of generality, suppose the former transition from A to B is associated with the symbol '1', as in the example. The new state B' then corresponds to the situation when state A is followed by '1'. As cloning continues, new states begin to express more and more specific situations, allowing the algorithm to incorporate richer context information when predicting the next symbol. The context used for prediction is implicitly determined by the longest string of symbols that matches all suffixes of paths leading to the current state of the FSM.

In the most basic version, the initial model contains a single state, corresponding to a memoryless source. When dealing with byte-aligned data, it is customary to start with a slightly more complex initial state machine which is capable of expressing within-byte dependencies. This initial FSM structure roughly corresponds to an order-7 binary Markov model. All transitions in the initial FSM are primed with a small initial visit count to avoid the zero-frequency problem, which is typically set to 1 initial visit. We note that although DMC restricts the source alphabet to binary symbols, it nevertheless exhibits solid performance on typical ASCII encoded text sequences (Cormack and Horspool, 1987).

Chapter 4

Representative Sampling Using Compression Models

In this chapter, we describe an information-based method for extracting a sample of representative and non-redundant documents from a large text corpus. The basic idea which we explore is to evaluate how well a statistical model, induced from a sample of selected documents, can *compress* the remaining (unselected) documents. The compression rate achieved in this way is used as a measure of the quality of the sample of documents. We develop an efficient algorithm to measure this objective function, based on the context tree weighting method. The utility of this approach is evaluated for different text mining tasks – initialization of k-means clustering, active learning for text classification, and detection of significant news events in a historical stream of broadcast news. Our experiments suggest that the approach compares favorably to specialized methods when applied to active learning and cluster initialization. In the news mining setting, we find that the method successfully identifies some of the major news stories to be found in the news data stream.

The methods described in this chapter are designed for modeling arbitrary sequential data, that is, data where each example is naturally represented as a sequence of symbols over a finite alphabet. Discrete sequences are typical of many important application domains such as text mining, mining biological sequences, music analysis, mining time series data, etc. In this thesis, we evaluate some of the possible text mining applications of the proposed approach, however, in the discussion we often refer to arbitrary discrete sequences, of which natural language text is just one example.

4.1 Problem Description

In recent years, the availability of data for research and analysis has increased dramatically. Large quantities of data can be harvested from the web or obtained from high-throughput experiments. The amount of data that is available to a researcher is usually far too large for exhaustive manual review, and may also exceed the available computational resources when using complex data mining methods. To tackle this problem, we aim to develop a method capable of extracting a small sample of documents which are particularly representative of a larger document collection.

Our problem statement is as follows. Given a large, unknown dataset, which data items should we inspect in order to gain as much knowledge about the data as possible, provided that only a small fraction of the data may be inspected? In case that part of the data is "known", for example, when a familiar document collection is augmented with some new texts, which data items convey the most information about material that is new (or different) in the "unknown" data with respect to the "known" data? The ability to extract such a representative sample can be useful in many settings, for example, to obtain a maximally informative subset of data items for manual inspection, or to assist further automated processing of the data in various data mining settings.

We aim to measure how well a probabilistic model trained from a particular data sample is able to predict (or, equivalently, compress) the unsampled part of the data (see Figure 4.1). Our goal is to find samples which result in a high probability of the "rest of the data" according to this criterion. If we assume that the probabilistic model used for this measurement is generally well-suited for modeling the data at hand, then the trained model represents, in a sense, the "knowledge" that is learned from the sample. The probability of the remaining data under the trained model entails the remaining uncertainty in the data given this knowledge, and can be used to measure the degree to which the sample is a useful and sufficient surrogate for the whole dataset. For example, in the extreme case when the remaining uncertainty in the data is zero, i.e. when the probability of the remaining data equals one, all of the information which exists in the document collection is contained in the sample. We develop an algorithm which iteratively selects documents to be added to the representative sample based on this objective function. In order to compute this objective function efficiently, we borrow some principal ideas from the context tree weighting compression method (Willems et al., 1995).

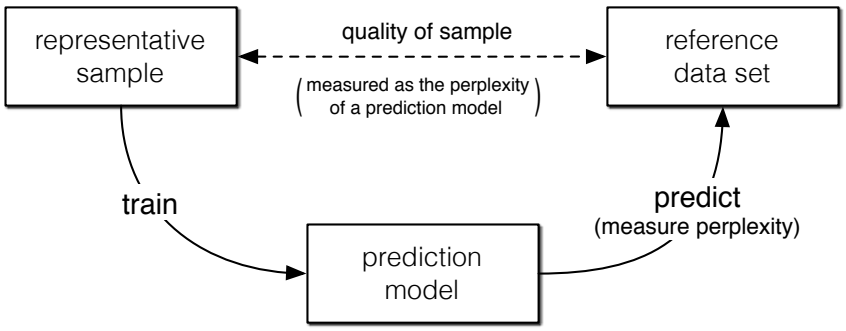


Figure 4.1: The quality of a data sample is assessed by training a probability model from the sample and measuring the perplexity of the sample-induced model on the whole data set.

4.2 Related Work

Instance selection describes a family of methods used in various data mining settings, aimed at reducing large datasets into manageable, or more specifically focused subsets. Instance-based learners, for which classification speed is proportional to the number of training examples, are particularly amenable to instance selection, and have traditionally been a focus of such methods. Survey papers in this field include (Liu and Motoda, 2001; Reinartz, 2002; Grochowski and Jankowski, 2004), and specifically for instance-based learning also (Wilson and Martinez, 2000). A typical setting for instance selection is to find instances which are most useful for a *particular* data mining tool, leading to various specialized instance selection methods geared towards specific data mining tasks and algorithms. We aim to extract instances which are generally representative of the data, in the sense that a minimal amount of information is lost due to data reduction, and without regard to any specific data mining tools that may be used on the resulting representative sample. Our model-based approach (as opposed to instance-based) and our focus on sequential data further distinguish our work from most instance selection methods found in the data mining literature.

The sampling method presented in this chapter is conceptually similar to static sampling described by John and Langley (1996). Static sampling also directly estimates how "representative" the sample is with respect to the whole. In static sampling, statistical tests are used to measure whether the distribution of attribute values in the sample differs from the distribution in the whole dataset. As in our case, the method is also not geared towards supervised learning, since there is no particular output attribute, and the method "*ignores the data-mining tool that will be used*" (John and Langley, 1996). Instead of the statistical tests used in static sampling, we propose an information-based measure to determine the quality of a sample.

The identification of exemplar data points is an integral part of certain clustering algorithms such as, for example, the k-medoids algorithm (Kaufman and Rousseeuw, 1990) and clustering by message-passing (Frey and Dueck, 2007). More generally, the result of any clustering method can be used to find a set of data points that are most representative of each identified cluster, and thus arguably representative of the dataset as a whole (cf. Reinartz, 2002, and references therein). However, while finding exemplar points may be a side-product of certain clustering algorithms, it is not their primary purpose, and while clustering may be used to the same effect, our method does not make use of clustering in any way. A major difference compared to using cluster representatives is that our method constructs the representative set of data points incrementally. It can thus also be used to find representative data points that best augment some previously selected data. An additional relationship also exists between our method and clustering using the expectation maximization algorithm (Dempster et al., 1977), in that maximizing data likelihood is the objective criterion in both cases.

Our representative sampling approach is based on the context tree weighting (CTW) data compression algorithm (Willems et al., 1995). The use of off-the-shelf data compression software to measure similarity between sequences has been proposed by many authors (cf. Chen et al., 1999; Li et al., 2001, 2004; Benedetto et al., 2002; Sculley and Brodley, 2006; Keogh et al., 2007), and is discussed also in Section 2.5 of this thesis. In this work, we also measure a type of compression-based similarity between a sample of representative data points and a reference dataset. However, we cannot resort to using standard data compression methods to achieve this task due to the inefficiency of such an approach. Although our approach is inspired by the CTW compression algorithm, we develop a novel algorithm designed specifically for our purposes.

The CTW algorithm itself has also received attention from the machine learning community. Dawy et al. (2004) consider its use for classification of text and DNA sequences while Begleiter et al. (2004) evaluate its prediction and classification performance in comparison to other variable-order Markov models. A mutual information kernel based on the CTW method was proposed by Cuturi and Vert (2005), who also evaluate its performance for protein classification in combination with an SVM classifier. Helmbold and Schapire (1997) adapt the CTW method for computing mixtures of decision tree classifiers as an alternative to decision tree pruning.

In this work, we adapt the CTW algorithm to efficiently estimate the cross-entropy of a prediction model, measured against a reference set of sequences. The prediction model is trained from a separate set of sequences and the cross-entropy estimate is used as a measure of how well the reference data is represented by the training data. Such cross-entropy estimates are widely used in corpus linguistics as a measure of similarity

between two corpora (Kilgarriff, 2001), except that word-level language models are used there, and that the measure is adapted so as to make it symmetric. We have no need for a symmetric measure since we are only interested in how well the reference dataset is represented by the training data, i.e. how much we can learn about the reference dataset from a particular sample of the data. Another difference is that we focus on optimizing the computation of the cross-entropy, since the measure is used to guide a search algorithm and must be recomputed frequently.

The application domains that are considered in our experimental evaluation have been studied extensively in the literature. In Sections 4.5.1, 4.6.1 and 4.7.1, we provide a brief description of these fields and the most relevant references before presenting our results in each of these domains.

4.3 The Cross-entropy Reduction Sampling Method

Given two sequences $\mathbf{x}$ and $\mathbf{y}$, we wish to measure how well $\mathbf{y}$ is represented by $\mathbf{x}$. More precisely, we aim to measure how well $\mathbf{y}$ can be *compressed* after observing $\mathbf{x}$. This is done by training a statistical model $\theta_{\mathbf{x}}$ from $\mathbf{x}$, and estimating the cross-entropy of this model against $\mathbf{y}$. The cross-entropy is estimated as the average code length achieved when the model $\theta_{\mathbf{x}}$ is used to compress $\mathbf{y}$ under optimal encoding, that is, under the assumption that no overhead is incurred in the conversion from probabilities to bit representations during compression.

Let $p(\mathbf{y}|\mathbf{x})$ denote the probability of $\mathbf{y}$ according to the model $\theta_{\mathbf{x}}$ induced from the sequence $\mathbf{x}$. An estimate of the cross-entropy between $\theta_{\mathbf{x}}$ and the information source of $\mathbf{y}$ is then

$$H(\mathbf{y}, \mathbf{x}) = -\frac{1}{\overline{\mathbf{y}}} \log_2 p(\mathbf{y}|\mathbf{x}) \ . \tag{4.1}$$

If $\mathbf{y}$ is long with respect to the memory of the source, this estimate will approach the actual cross-entropy almost surely if the source is ergodic (Algoet and Cover, 1988). In the following, we refer to $H(\mathbf{y}, \mathbf{x})$ as the cross-entropy between two sequences, bearing in mind that $H(\mathbf{y}, \mathbf{x})$ in truth approximates the cross-entropy between a prediction model $\theta_{\mathbf{x}}$ inferred from $\mathbf{x}$, and the probability distribution from which $\mathbf{y}$ was sampled.

Note that the cross-entropy estimate $H(\mathbf{y}, \mathbf{x})$ will be small if the probability $p(\mathbf{y}|\mathbf{x})$ is high. This is the case when most of the statistical properties of $\mathbf{y}$ are captured by the model $\theta_{\mathbf{x}}$, which also implies that $\mathbf{x}$ exhibits the same statistical properties as $\mathbf{y}$. Our goal is to find a good representative for $\mathbf{y}$ from a set of sequences X. To this end, we

select the sequence $\mathbf{s} \in \mathrm{X}$ which minimizes the cross entropy estimate:

$$\mathbf{s} = \arg\min_{\mathbf{x} \in \mathrm{X}} H(\mathbf{y}, \mathbf{x}) \quad , \tag{4.2}$$

or, equivalently, maximizes a "length-normalized" probability of $\mathbf{y}$:

$$\mathbf{s} = \arg\max_{\mathbf{x} \in \mathrm{X}} p(\mathbf{y}|\mathbf{x})^{1/\overline{\mathbf{y}}} \ . \tag{4.3}$$

We use cross-entropy estimates $H(\mathbf{y}, \mathbf{x})$ to guide a search in the space of possible sequences $\mathbf{x} \in \mathrm{X}$ to find sequences which are representative of $\mathbf{y}$. Therefore, it is important that the computation of the cross-entropy is efficient. A straight-forward implementation requires $\mathcal{O}(\overline{\mathbf{x}} + \overline{\mathbf{y}})$ time to compute $H(\mathbf{y}, \mathbf{x})$ for each candidate $\mathbf{x} \in \mathrm{X}$. Note that $\mathbf{y}$ is large compared to $\mathbf{x}$ for interesting applications. In the following, we describe an incremental algorithm capable of computing $H(\mathbf{y}, \mathbf{x_2})$ from $H(\mathbf{y}, \mathbf{x_1})$ with complexity that only depends on the sequences $\mathbf{x_1}$ and $\mathbf{x_2}$, and not on $\mathbf{y}$, making the search among sequences $\mathbf{x} \in \mathrm{X}$ computationally feasible.

4.3.1 Computing Cross-entropy Estimates

We now describe an efficient algorithm to compute cross-entropy estimates for sequential data. The algorithm is inspired by the CTW method, and is similarly based on mixtures of tree source models. A detailed description of tree source models and the context tree weighting algorithm is given in Section 3.4, and the following discussion is based on the material presented in that section. We also use the escape method D zero-order estimator described in Section 3.1.

Recall that the probability $p_w(\mathbf{x})$ assigned to a sequence $\mathbf{x}$ by the CTW method is a Bayesian average of predictions based on all possible tree source models $\mathcal{K}$, up to some maximum order d:

$$p_w(\mathbf{x}) = \sum_{\mathcal{K}} \pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K}) \ . \tag{4.4}$$

We first observe that the CTW algorithm implicitly produces a posterior distribution over tree source structures after seeing the sequence $\mathbf{x}$, and assuming the prior over tree source structures $\pi(\mathcal{K})$:

$$\pi(\mathcal{K}|\mathbf{x}) = \frac{\pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K})}{p_w(\mathbf{x})} \ . \tag{4.5}$$

Using this posterior over model structures, we compute $p(\mathbf{y}|\mathbf{x})$ as

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathcal{K}} \pi(\mathcal{K}|\mathbf{x}) p(\mathbf{y}|\mathcal{K}, \mathbf{x}) \tag{4.6}$$

where $p(\mathbf{y}|\mathcal{K}, \mathbf{x})$ is the probability assigned to $\mathbf{y}$ by the tree source model corresponding to $\mathcal{K}$, in which the zero-order estimators for each context $\mathbf{v} \in \mathcal{K}$ are trained only from statistics in $\mathbf{x}$:

$$p(\mathbf{y}|\mathcal{K}, \mathbf{x}) = \prod_{\mathbf{v} \in \mathcal{K}} \prod_{a \in \Sigma} p_e(a; \mathbf{x}\|\mathbf{v})^{f(a, \mathbf{y}\|\mathbf{v})} \ . \tag{4.7}$$

Here, $p_e(a; \mathbf{x}\|\mathbf{v})$ denotes the probability of the symbol a according to a zero-order estimator trained on the sequence $\mathbf{x}\|\mathbf{v}$ of individual symbols which immediately follow occurrences of $\mathbf{v}$ in $\mathbf{x}$, while $f(a, \mathbf{y}\|\mathbf{v})$ is the number of occurrences of $\mathbf{v}a$ in $\mathbf{y}$. Intuitively, $p(\mathbf{y}|\mathcal{K}, \mathbf{x})$ measures how well we are able to predict $\mathbf{y}$ if we only "know" $\mathbf{x}$, given a particular, fixed tree source structure which is specified by $\mathcal{K}$.

It turns out that $p(\mathbf{y}|\mathbf{x})$ can be computed efficiently using the CTW method. Plugging Equation (4.5) into Equation (4.6) gives

$$p(\mathbf{y}|\mathbf{x}) = \frac{\sum_{\mathcal{K}} \pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K}) p(\mathbf{y}|\mathcal{K}, \mathbf{x})}{p_w(\mathbf{x})} = \frac{p_w(\mathbf{x}, \mathbf{y})}{p_w(\mathbf{x})} \ , \tag{4.8}$$

where we have replaced the sum in the numerator with the shorthand notation $p_w(\mathbf{x}, \mathbf{y})$. The value of $p_w(\mathbf{x}, \mathbf{y})$ is computed recursively by traversing branches of a full order-d context tree, starting at the root

$$p_w(\mathbf{x}, \mathbf{y}) = p_w(\mathbf{x}, \mathbf{y}; \epsilon) \tag{4.9}$$

and using the following recursion:

$$p_w(\mathbf{x}, \mathbf{y}; \mathbf{v}) = \begin{cases} p_e(\mathbf{x}, \mathbf{y}; \mathbf{v}) & \text{if } \overline{\mathbf{v}} = d; \\ \alpha\, p_e(\mathbf{x}, \mathbf{y}; \mathbf{v}) + (1 - \alpha) \prod_{a \in \Sigma} p_w(\mathbf{x}, \mathbf{y}; a\mathbf{v}) & \text{if } \overline{\mathbf{v}} < d. \end{cases} \tag{4.10}$$

The recursion is analogous to the CTW recursion in Equation (3.15), except that $p_e(\mathbf{x}\|\mathbf{v})$ is replaced by $p_e(\mathbf{x}, \mathbf{y}; \mathbf{v})$. The value of $p_e(\mathbf{x}, \mathbf{y}; \mathbf{v})$ estimates the joint probability of $\mathbf{x}\|\mathbf{v}$ and $\mathbf{y}\|\mathbf{v}$, but keeps the next-symbol probabilities of the estimator frozen after $\mathbf{x}$ is observed:

$$p_e(\mathbf{x}, \mathbf{y}; \mathbf{v}) = p_e(\mathbf{x}\|\mathbf{v}) \prod_{a \in \Sigma} p_e(a; \mathbf{x}\|\mathbf{v})^{f(a, \mathbf{y}\|\mathbf{v})} \ . \tag{4.11}$$

Here, $p_e(\mathbf{x}\|\mathbf{v})$ is the probability assigned to the sequence $\mathbf{x}\|\mathbf{v}$ by an adaptive zero-order estimator, and $p_e(a; \mathbf{x}\|\mathbf{v})$ is the zero-order probability of the individual symbol a after observing $\mathbf{x}\|\mathbf{v}$.

As we shall see, the first factor in Equation (4.11), $p_e(\mathbf{x}\|\mathbf{v})$, measures the utility of the context $\mathbf{v}$ for predicting $\mathbf{x}$, and contributes towards the posterior $\pi(\mathcal{K}|\mathbf{x})$ for all $\mathcal{K}$ with $\mathbf{v} \in \mathcal{K}$. The product on the right-hand side of the equation contributes towards

$p(\mathbf{y}|\mathcal{K},\mathbf{x})$ for all $\mathcal{K}$ with $\mathbf{v} \in \mathcal{K}$. This becomes apparent when the recursive definition for $p_w(\mathbf{x},\mathbf{y})$ is fully expanded, resulting in a (very large) sum containing one term for each context set $\mathcal{K}$ with an order less or equal to d:

$$\begin{aligned} p_w(\mathbf{x},\mathbf{y}) &= \sum_{\mathcal{K}} \left[(1-\alpha)^{\frac{|\mathcal{K}|-1}{|\Sigma|-1}} \alpha^{|\mathcal{K}|} \prod_{\mathbf{v}\in\mathcal{K}} \left(p_e(\mathbf{x}\|\mathbf{v}) \prod_{a\in\Sigma} p_e(a;\mathbf{x}\|\mathbf{v})^{f(a,\mathbf{y}\|\mathbf{v})} \right) \right] \\ &= \sum_{\mathcal{K}} \left[(1-\alpha)^{\frac{|\mathcal{K}|-1}{|\Sigma|-1}} \alpha^{|\mathcal{K}|} \left(\prod_{\mathbf{v}\in\mathcal{K}} p_e(\mathbf{x}\|\mathbf{v}) \right) \left(\prod_{\mathbf{v}\in\mathcal{K}} \prod_{a\in\Sigma} p_e(a;\mathbf{x}\|\mathbf{v})^{f(a,\mathbf{y}\|\mathbf{v})} \right) \right] \\ &= \sum_{\mathcal{K}} \pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K}) p(\mathbf{y}|\mathcal{K},\mathbf{x}) \ . \end{aligned} \tag{4.12}$$

The last equalities stem from the definition of $p(\mathbf{y}|\mathcal{K},\mathbf{x})$ in Equation (4.7), and the original definition of how the probability $p(\mathbf{x}|\mathcal{K})$, of a sequence $\mathbf{x}$ given a particular tree source structure $\mathcal{K}$, is computed by the CTW algorithm (see Section 3.4.1, Eq. 3.12). The derivation confirms that the recursive method for computing $p_w(\mathbf{x},\mathbf{y})$ is equivalent to the original definition given in Equation (4.8).

4.3.2 Efficiency and Incremental Updates

The computation of $p(\mathbf{y}|\mathbf{x})$ preserves both the efficiency and the incremental nature of the CTW method. This can be seen from Equation (4.8), by observing that $p(\mathbf{y}|\mathbf{x})$ is simply the ratio between CTW mixtures over $p_e(\mathbf{x},\mathbf{y};\mathbf{v})$ and $p_e(\mathbf{x}\|\mathbf{v})$. The mixtures can be computed in time proportional to $\mathcal{O}(dn)$, where $n = \overline{\mathbf{x}} + \overline{\mathbf{y}}$ and d is the model order.

The cross-entropy can be computed incrementally as symbols are appended to either $\mathbf{x}$ or $\mathbf{y}$, since, as in the original CTW mixture, we do not need to update $p_w(\mathbf{x},\mathbf{y};\mathbf{v})$ for any $\mathbf{v}$ that is not a suffix of $\mathbf{x}$ or $\mathbf{y}$. For example, consider the situation where a single symbol s is appended to $\mathbf{y}$, and let $i = \overline{\mathbf{y}} + 1$. The required updates to $p_w(\mathbf{x},\mathbf{y};\mathbf{v})$ are computed recursively, starting with the longest suffix of $\mathbf{y}$ in the context tree, $\mathbf{y}_{i-d}^{i-1}$:

$$p_w(\mathbf{x},\mathbf{y}s;\mathbf{y}_{i-d}^{i-1}) = p_e(\mathbf{x},\mathbf{y}s;\mathbf{y}_{i-d}^{i-1}) \ . \tag{4.13}$$

The remaining updates are computed in decreasing context length k:

$$\begin{aligned} p_w(\mathbf{x},\mathbf{y}s;\mathbf{y}_{i-k}^{i-1}) &= \alpha\, p_e(\mathbf{x},\mathbf{y}s;\mathbf{y}_{i-k}^{i-1}) \\ &\quad + (1-\alpha) \prod_{a\in\Sigma} p_w(\mathbf{x},\mathbf{y};a\mathbf{y}_{i-k}^{i-1}) \frac{p_w(\mathbf{x},\mathbf{y}s;\mathbf{y}_{i-k-1}^{i-1})}{p_w(\mathbf{x},\mathbf{y};\mathbf{y}_{i-k-1}^{i-1})} \end{aligned} \tag{4.14}$$

for $k = d-1, k = d-2, \ldots$, until $p_w(\mathbf{x},\mathbf{y}s;\epsilon)$ is updated to finish the recursion.

The incremental update requires $\mathcal{O}(d)$ time, provided that $p_w(\mathbf{x}, \mathbf{y}; \mathbf{v})$ and $p_e(\mathbf{x}, \mathbf{y}; \mathbf{v})$ are maintained in memory for all $\mathbf{v}$ that appear in $\mathbf{x}$ or $\mathbf{y}$, and that $p_e(\mathbf{x}, \mathbf{y}s; \mathbf{v})$ can be computed from $p_e(\mathbf{x}, \mathbf{y}; \mathbf{v})$ in constant time, which holds for most popular zero-order estimators. Similar incremental updates are used to recompute the cross-entropy when symbols are appended to $\mathbf{x}$ instead of $\mathbf{y}$, as well as when removing symbols from either $\mathbf{x}$ or $\mathbf{y}$.

4.3.3 Cross-entropy Between Sequence Sets

Our main goal is the selection of a small sample of sequences S from a large pool of strings D, so that the sample conveys as much information as possible about the original dataset D. The objective function used to measure the quality of the sample is the sample cross-entropy $H(\mathrm{D}, \mathrm{S})$. The sample cross-entropy $H(\mathrm{D}, \mathrm{S})$, which is defined for a pair of sequence *sets*, is a generalization of the cross-entropy between two individual sequences, and is similar to the cross-entropy between the two concatenations of strings in S and in D.

The cross-entropy $H(\mathbf{y}, \mathbf{x})$ between a pair of sequences $\mathbf{x}$ and $\mathbf{y}$ is computed recursively from zero-order probabilities of subsequences $\mathbf{x}\|\mathbf{v}$ and $\mathbf{y}\|\mathbf{v}$ for all $\mathbf{v} \in \Sigma^d$. The result is insensitive to permutations of symbols within these subsequences, provided that the zero-order estimators used are also insensitive to permutations. Similarly, the sample cross-entropy $H(\mathrm{D}, \mathrm{S})$ is computed from $\mathrm{D}\|\mathbf{v}$ and $\mathrm{S}\|\mathbf{v}$, where $\mathrm{D}\|\mathbf{v}$ is the concatenated sequence of symbols which immediately follow occurrences of $\mathbf{v}$ in *all* sequences in D, and $\mathrm{S}\|\mathbf{v}$ is defined analogously. Note that the order in which symbols are arranged in $\mathrm{D}\|\mathbf{v}$ and $\mathrm{S}\|\mathbf{v}$ does not affect the value of $H(\mathrm{D}, \mathrm{S})$. The sample cross-entropy $H(\mathrm{D}, \mathrm{S})$ can be updated incrementally as additional data points are added to, or removed from, S or D. We set $H(\emptyset, \mathrm{S}) = 0$ and $H(\mathrm{D}, \emptyset) = \log_2 |\Sigma|$ by definition.

4.3.4 Generating Representative Samples

We consider a greedy algorithm which constructs S, the representative sample of D, incrementally, by successively adding sequences which result in the largest reduction of the sample cross-entropy:

$$\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S}) = H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S}) - H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}\}) \ . \tag{4.15}$$

The procedure is detailed in Algorithm 1 (the cross-entropy reduction sampling algorithm – CERS). As input parameters, the algorithm accepts the sequence dataset D, the number m of representative sequences which should be extracted from D, as well as parameters required to compute the cross-entropy reduction $\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})$. The latter

include the maximum order d of tree source models included in the mixture computation, the zero-order estimator function $p_e(\cdot)$, and the CTW α parameter, which controls the prior over context tree structures. The result of the algorithm is a sequence set $\mathrm{S}, \mathrm{S} \subset \mathrm{D}, |\mathrm{S}| = m$, containing representative sequences from the input dataset D.

Algorithm 1: The CERS algorithm.

Input : Sequence dataset $\mathrm{D} = \{\mathbf{x}; \mathbf{x} \in \Sigma^*\}$.
Input : Desired sample size $m > 0$.
Input : Parameters required to compute ΔH: the model order d, zero-order estimator routine $p_e(\cdot)$, and the CTW weighting parameter α.
Output : Representative set S of size m.

$\mathrm{S} \leftarrow \emptyset$; `/* Initialization */`
while $|\mathrm{S}| < m$ **do**
  $\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \mathrm{D}} [\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})]$; `/* Select representative point s */`
  $\mathrm{S} \leftarrow \mathrm{S} \cup \{\mathbf{s}\}$; `/* Add s to S */`
  $\mathrm{D} \leftarrow \mathrm{D} \setminus \{\mathbf{s}\}$; `/* Remove s from D */`
end

Note that the objective function for selecting representative examples could also be defined as $\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S}) = H(\mathrm{D}, \mathrm{S}) - H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}\})$, i.e. the reduction in the average code length for D after $\mathbf{x}$ is *moved* to S. However, this formulation is prone to selecting outliers, since moving outliers from D may improve compression of D only due to denoising of D, and not due to improvements in the modeling of other data points in D. Training on outliers will not improve modeling of "regular" examples in D, and is not useful for generalization, however, it does not necessarily decrease the probability of "regular" examples in D. This is due to context modeling (modeling conditional probabilities) and the resulting large parameter space of compression models: It is possible that training a compression model on a sequence $\mathbf{x} \in \mathrm{D}$ which is unrelated to other sequences in D only affects a space of model parameters that is not used for prediction for other data in D. A third variant of the CERS data selection criterion would be $\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S}) = H(\mathrm{D}, \mathrm{S}) - H(\mathrm{D}, \mathrm{S} \cup \{\mathbf{x}\})$. Although we did not experiment with this version, we believe it would also be susceptible to sampling outliers. This is because adding sequences $\mathbf{x}$ that are outliers in D to the training data S would lead to a major reduction in the compressed length of such $\mathbf{x}$ in D, which has a similar net effect as removing $\mathbf{x}$ from D completely.[1]

Unlike other variants, the formulation in Equation (4.15) only measures the cross-

[1] This effect is exacerbated when using CERS parameters that encourage models which easily overfit the training data, for example, by using a small α parameter and "aggressive" zero-order estimators p_e with minimal smoothing.

entropy reduction that results from improvements in statistical modeling of *other* examples in D due to training a model on $\mathbf{x}$, and thus rewards only examples that are useful for generalization. The basic idea of CERS is however flexible, and D could also be a held-out reference dataset, used only to measure the quality of a sample drawn from a third, independent set. Note that as the size of the dataset D increases, the three variants of computing the cross-entropy reduction converge. This is because the difference between removing or not removing a single sequence $\mathbf{x}$ from D has a diminishing effect on the overall statistics of D, if the size of D grows, while the average length of individual sequences $\mathbf{x} \in \mathrm{D}$ remains constant.

The time required to compute $\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})$ for a single data point is proportional to $O(d\overline{\mathbf{x}})$ (see Section 4.3.2), where d is the maximum order of tree source models considered in the computation. The sampling algorithm therefore requires $\mathcal{O}(dmn)$ time to select a sample of m data points from a dataset of size n (n is the sum of the lengths of all sequences in the dataset). Since d is a constant parameter, the time required to add each additional data point to the representative sample is linear in the size of the dataset.

Let us conclude this subsection with an illustrative example. Consider the set of strings $\mathrm{D} = \{$`aaabbb`, `aaaaaa`, `bbbbbb`$\}$. In this case, the first string selected by cross-entropy reduction sampling, as the most representative of D, is `aaabbb`. From this string it is possible to infer that both symbols `a` and `b` are equally common, and tend to occur in long sequences of the same symbol. The statistical properties inferred from this sample can indeed help to compress the remaining two strings. In order to keep reducing the cross-entropy (i.e. improve compression of D), we expect the algorithm to select successive data points which contain:

- patterns that are *common* in D (in order to compress D);
- *diverse* patterns (adding patterns that are already well-represented in S has diminishing returns).

If the dataset contains high-density clusters, we would intuitively expect that the representative sample generated by CERS contains data points that are representative of large clusters. There are many applications in which such samples are desirable, and we consider some of the possible applications of CERS sampling in our experiments.

4.3.5 Scoring Subsequences

The objective function used to select each additional sequence $\mathbf{x}$ to be added to the representative sample of D is the cross-entropy reduction ΔH over the remaining sequences

in $\mathrm{D} \setminus \{\mathbf{x}\}$. We might wish to determine which *part* of the selected sequence contributes most to the cross-entropy reduction. This gives insight into why a particular sequence is selected by CERS. It can also be useful in practical applications where the goal is to identify informative *parts* of a sequence. For example, in subsequent experiments on text data, we use this information to extract keywords from news documents.

The cross-entropy reduction that results from adding a sequence $\mathbf{x}$ to the representative sample S is computed online, by sequentially processing each position in $\mathbf{x}$. For each position $i = 1 \ldots \overline{\mathbf{x}}$, the cross-entropy reduction equals

$$\Delta H_i(\mathbf{x}, \mathrm{D}, \mathrm{S}) = H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}_1^{i-1}\}) - H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}_1^{i}\}) \ . \tag{4.16}$$

The combination of x_i and its context $\mathbf{x}_{i-d}^{i-1}$ contributes towards $\Delta H_i(\mathbf{x}, \mathrm{D}, \mathrm{S})$, so symbols between x_{i-d} and x_i are to be "credited" for the cross-entropy reduction at x_i. Instead of distributing the cross-entropy reduction uniformly among positions x_{i-d}, ... x_i, we wish to consider the "importance" of each of the contexts ϵ, $\mathbf{x}_{i-1}^{i-1}$, ... $\mathbf{x}_{i-d}^{i-1}$. To this end, we distribute $\Delta H_i(\mathbf{x}, \mathrm{D}, \mathrm{S})$ among subsequences $\mathbf{x}_i^i$, $\mathbf{x}_{i-1}^i$, ... $\mathbf{x}_{i-d}^i$ according to weights $w(\mathbf{x}_{i-k}^i)$, with $k = 0 \ldots d$. We also require that

$$\sum_{k=0}^{d} w(\mathbf{x}_{i-k}^i) = 1 \ . \tag{4.17}$$

Each weight $w(\mathbf{x}_{i-k}^i)$ reflects how much the corresponding context $\mathbf{x}_{i-k}^{i-1}$ contributes towards a low cross-entropy after x_i is appended to S. The heuristic used to compute $w(\mathbf{x}_{i-k}^i)$ is detailed in the following subsection.

After the weights $w(\mathbf{x}_{i-k}^i)$ are determined, symbols in $\mathbf{x}$ at positions $j = i, i-1, \ldots i-d$, are "credited" with

$$h_{ij} = \Delta H_i(\mathbf{x}, \mathrm{D}, \mathrm{S}) \sum_{k=i-j}^{d} \frac{w(\mathbf{x}_{i-k}^i)}{k+1} \ . \tag{4.18}$$

In this manner, the cross-entropy reduction $\Delta H_i(\mathbf{x}, \mathrm{D}, \mathrm{S})$ at each position i in $\mathbf{x}$ is distributed among x_i and its d preceding symbols. The final score h_j for each symbol x_j in $\mathbf{x}$ is

$$h_j = \sum_{i=j}^{j+d} h_{ij} \ , \tag{4.19}$$

that is, the sum of the "credits" it receives due to the cross-entropy reduction at positions between j and $j+d$ in $\mathbf{x}$.

Subsequence Weighting Heuristic

We now detail how the weights that are used in scoring subsequences are calculated. As each sequence $\mathbf{x}$ is added to the representative sample S, the cross-entropy $H_i = H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}_1^i\})$ is updated sequentially, once for each symbol x_i in $\mathbf{x}$. The "credit" for the cross-entropy reduction $\Delta H_i = H_{i-1} - H_i$ at position i is distributed among the subsequences $\mathbf{x}_{i-k}^i$ according to $w(\mathbf{x}_{i-k}^i)$, for $k = 0 \ldots d$.

To arrive at a reasonable weighting, we analyze how the cross-entropy, after x_i is appended to S, is computed:

$$\begin{aligned}
H_i &= H(\mathrm{D} \setminus \{\mathbf{x}\}, \mathrm{S} \cup \{\mathbf{x}_1^i\}) \\
&= H(\mathbf{d}', \mathbf{s}^i) && \textit{shorthand notation} \\
&= \frac{-1}{\overline{\mathbf{d}}} \log_2 p(\mathbf{d}|\mathbf{s}^i) && \textit{by Eq. (4.1)} \\
&= \frac{-1}{\overline{\mathbf{d}}} \log_2 \frac{p_w(\mathbf{s}^i, \mathbf{d})}{p_w(\mathbf{s}^i)} && \textit{by Eq. (4.8)} \\
&= \frac{-1}{\overline{\mathbf{d}}} \left(\log_2 p_w(\mathbf{s}^i, \mathbf{d}) - \log_2 \frac{p_w(\mathbf{s}) p_w(\mathbf{s}^i)}{p_w(\mathbf{s})} \right) \\
&= \frac{\log_2 p_w(\mathbf{s})}{\overline{\mathbf{d}}} + \frac{\log_2 p_w(\mathbf{x}_1^i|\mathbf{s})}{\overline{\mathbf{d}}} + \frac{-\log_2 p_w(\mathbf{s}^i, \mathbf{d})}{\overline{\mathbf{d}}} \quad . && (4.20)
\end{aligned}$$

We use $\mathbf{s}$, $\mathbf{s}^i$ and $\mathbf{d}$ as shorthand notation for what is essentially the concatenation of all sequences in S, $\mathrm{S} \cup \{\mathbf{x}_1^i\}$ and $\mathrm{D} \setminus \{\mathbf{x}\}$, respectively. Note that this is not a strict concatenation, since no dependencies that span across two sequences are permitted.

We now turn our attention to the sum in Equation (4.20) and analyze its behavior in the typical usage scenario for CERS sampling, that is, sampling from a *large* dataset D. If D is large and $\mathbf{x}$ is comparatively small, the $1/\overline{\mathbf{d}}$ factor will be similar for all $\mathbf{x}$. Moreover, the $1/\overline{\mathbf{d}}$ factor depends only on the length of $\mathbf{x}$, but not on its contents, so it has no bearing on the relative importance of subsequences of $\mathbf{x}$. If we eliminate $1/\overline{\mathbf{d}}$, the first term is completely independent of $\mathbf{x}$, and is thus unrelated to the reduction in cross-entropy as $\mathbf{x}$ is added to the sample S. The second term, containing the CTW conditional probability $p_w(\mathbf{x}_1^i|\mathbf{s})$, approaches zero as $\overline{\mathbf{d}}$ becomes large, since $\mathbf{x}_1^i$ is short in comparison. We are left with the last term, which is also responsible for most of the cross-entropy reduction resulting from adding $\mathbf{x}$ to S, since both the numerator and the denominator in the last term can grow in proportion to $\overline{\mathbf{d}}$.

Fortunately, it is relatively straight-forward to estimate the impact that various contexts of x_i have on $p_w(\mathbf{s}^i, \mathbf{d})$. Recall that $p_w(\mathbf{s}^i, \mathbf{d})$ is computed from $p_w(\mathbf{s}^i, \mathbf{d}; \mathbf{v})$ for different contexts $\mathbf{v}$, according to Equations (4.9) and (4.10). The mixture $p_w(\mathbf{s}^i, \mathbf{d}; \mathbf{v})$ defines the "contribution" to $p_w(\mathbf{s}^i, \mathbf{d})$ of all context tree models that contain the context

$\mathbf{v}$, and all context tree models that contain contexts of which $\mathbf{v}$ is a suffix (higher-order models). The ratio between these two terms determines the relative importance of the corresponding contexts, which we use to determine the weights $w(\mathbf{x}_{i-k}^{i})$, as follows:

$$\frac{w(\mathbf{x}_{i-k}^{i})}{\sum_{k'=k+1}^{d} w(\mathbf{x}_{i-k'}^{i})} = \frac{\alpha\, p_e(\mathbf{s}^i, \mathbf{d}; \mathbf{x}_{i-k}^{i-1})}{(1-\alpha) \prod_{a\in\Sigma} p_w(\mathbf{s}^i, \mathbf{d}; a\mathbf{x}_{i-k}^{i-1})} \quad . \tag{4.21}$$

Equations (4.17) and (4.21) uniquely define $w(\mathbf{x}_{i-k}^{i})$ for all $k \in [0 \ldots d]$. Since $p_w(\mathbf{s}^i, \mathbf{d}; \mathbf{x}_{i-k}^{i-1})$ needs to be computed in order to evaluate the cross-entropy reduction due to sampling $\mathbf{x}$, determining $w(\mathbf{x}_{i-k}^{i})$ incurs no additional computational cost.

4.4 Experimental Evaluation

We evaluate the utility of CERS for a number of text mining tasks, all of which may benefit from the selection of texts that are representative of a large document collection. The applications considered are: initialization of k-means clustering, active learning for text categorization, and identification of major stories in news streams.

4.4.1 Datasets

Our experiments are based on three publicly available datasets, all of which are frequently used in text mining research. The basic statistics on these three datasets are given in Table 4.1.

Dataset	*Examples*	*Classes*	*Largest class*	*Smallest class*	*Label entropy*
20 Newsgroups	18828	20	999	628	4.31
Reuters 21578	8085	5	3931	539	1.86
RCV1	12670	4	3857	2342	1.97

Table 4.1: Basic statistics for the evaluation datasets. The maximum and minimum number of examples in a class are listed. The "label entropy" column represents the uncertainty of the distribution of class labels of examples in the dataset, measured as the entropy of this distribution (in bits).

The 20 Newsgroups dataset[2] consists of 18828 usenet posts distributed roughly evenly across 20 newsgroups. We used the "18828" version of the Newsgroups dataset, which contains the bodies, `Subject:` and `From:` headers of posts, and does not contain duplicates otherwise found in the original dataset. The raw example files were used as documents.

The Reuters 21578 dataset[3] contains news articles in 135 categories. Articles may

[2] Available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`

[3] Available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

belong to zero or several categories. The distribution among categories is heavily skewed, with most categories containing less than 10 documents. For our evaluation dataset, we selected all documents in the categories `earn`, `acq`, `money-fx`, `grain` and `crude` – the five largest categories according the standard ModApte split.[4] After removing all documents belonging to more than one of these five categories, our dataset contained a total of 8085 documents. We used the title and body elements of documents in the original SGML files.

The RCV1 corpus[5] is a large dataset containing the entire English language news output of Reuters in a one year period from August 1996 to August 1997 – over 800,000 news stories covering a broad range of topics. The predominant amount of articles covers financial news. We selected stories from four general news categories of comparable size and discarded documents belonging to more than one of these four categories. The categories selected were `science` (2342 articles), `religion` (2775 articles), `entertainment` (3696 articles) and `weather` (3857 articles)[6]. Documents were represented by the headline and paragraph elements of the original XML-encoded data.

4.4.2 Data Representation

As a preprocessing step, all documents were converted to lower case, sequences of non-alphanumeric characters were replaced with a single blank space, and all digit characters were replaced with a special symbol '`#`'. Sequences of more than 30 consecutive characters *not* delimited by white-space were removed, which eliminates most ASCII-encoded binary attachments in Usenet posts. The resulting alphabet contains 28 symbols – 26 letters, '`#`' and '`space`'.

Our sampling algorithm models text as a sequence of symbols. However, the clustering and classification methods that facilitate our evaluation are based on attribute-vector representations. The standard "*bag-of-words*" document representation was used for these methods.[7] The vocabulary was constructed from substrings delimited by white-space in the running text. We did not remove stopwords and no stemming was applied. Features occurring in only one training document and those occurring in all training

[4]The top five categories were also chosen since there is relatively little overlap between these categories, which is due to the pseudo-hierarchical labeling scheme used in this dataset. In this way, relatively few documents needed to be removed to produce a unilabeled dataset for the clustering evaluation.

[5]Available from NIST, see `http://trec.nist.gov/data/reuters/reuters.html`

[6]Note that the `weather` category mostly contains news concerning the impact of weather-related events (e.g. flooding, hurricane reports), rather than weather forecasts, as one might assume from the label.

[7]During our initial experiments in cluster initialization, we also considered using character n-grams for the attribute-vector document representation. However, the absolute clustering performance was lower when using character n-grams instead of "bag of words" vectors, while the ranking of different cluster initialization methods was consistent.

documents were discarded.

Features were weighted using a variant of the term frequency–inverse document frequency scheme (TF-IDF; Salton and Buckley, 1988). The weight of feature i in document $\mathbf{d}$ is given by

$$w_i(\mathbf{d}) = \log(tf_i(\mathbf{d}) + 1)\ \log \frac{|\mathrm{D}|}{df_i}\ ,$$

where D is the training dataset, $tf_i(\mathbf{d})$ denotes the frequency of feature i in document $\mathbf{d}$, and df_i is the number of documents in D for which $tf_i(\mathbf{d}) > 0$. All bag-of-words vectors were normalized by their Euclidean norm. In randomized trials, the vocabulary and document frequency components df_i were rebuilt according to the document subset "visible" to the clustering or classification algorithm.

4.4.3 CERS Parameters

An escape-method D zero-order estimator (Howard, 1993) was used for the CERS algorithm (see Section 3.1.2). The α parameter was set to its default value of $\frac{|\Sigma|-1}{|\Sigma|}$ for $|\Sigma| = 28$ (see Section 3.4.3). We set the order parameter to $d = 5$ in all experiments, since order-5 character-level models are generally thought to be sufficient for modeling English text (Teahan and Cleary, 1996). We stress that the CERS algorithm, like the CTW method, is not particularly sensitive to the choice of d, in the sense that a higher-order model is unlikely to degrade performance. This is because predictions based on higher-order contexts receive extremely low weights in the CTW mixture, unless there is sufficient training data to support them. The reason for limiting d in practice is to reduce computation time and the amount of memory required to store the model.

4.4.4 Computational Efficiency of CERS

The computational efficiency of our CERS implementation is assessed in Table 4.2, for the task of sampling 100 representative data points from the full-size datasets that form the basis of our experiments. The average processing time that is required to score individual documents while constructing the representative sample is also reported. The execution time was measured on an Intel Xeon E5430 processor with a clock speed of 2.66 GHz. The time required to select each successive document in the representative sample (the iteration time) and the document scoring throughput (in kilobytes per second) were consistently stable across all 100 sampling iterations in each of the experiments, and did not deviate from the reported average for more than 5% in any iteration of the sampling procedures.

	Dataset size		*Running time*		*Processing speed*	
Dataset	documents	kilobytes	total (hr)	iteration (min)	docs/sec	kB/sec
20 Newsgroups	18828	33,172	143.94	86.37	7.26	6.03
Reuters 21578	8085	5,524	16.18	9.71	27.67	9.14
RCV1	12670	24,700	82.64	49.58	8.50	8.21

Table 4.2: CERS running time and processing throughput of the cross-entropy reduction computation that is used to score documents.

4.5 Applications: Cluster Initialization

The k-means method (Forgy, 1965; Lloyd, 1982) is an iterative clustering technique widely used in scientific and industrial applications (Berkhin, 2006). Clusters are represented by centroids (mean points) and each data point is assigned to the cluster corresponding to its nearest centroid. Starting from an arbitrary initial solution, the algorithm iteratively recomputes the mean points of each cluster and updates cluster assignments of data points according to the newly obtained centroids. It is known that the k-means algorithm is sensitive to the initial choice of centroids.

The CERS method was designed to extract a sample of representative and diverse data points from a large dataset. Ideally, such representative points should lie in dense areas which correspond to cluster centers. If the points are truly diverse, they are also likely to belong to different clusters. We therefore expect that data points selected by CERS constitute a good set of initial centroids for the k-means algorithm. In the following, we evaluate CERS as an initialization method for the k-means clustering algorithm.

4.5.1 Related Work

Many initialization techniques for k-means clustering have been proposed in the literature. We briefly describe some of these methods here. Several comparative evaluations of k-means initialization techniques are available for a more thorough review of this field (Steinley and Brusco, 2007; Peña et al., 1999; Meilă and Heckerman, 1998; He et al., 2004).

A very common practice is to initialize centroids by choosing k data points at random, also known as Forgy's initialization (Forgy, 1965; He et al., 2004). Alternative random initialization strategies include generating initial centroids by randomly perturbing the mean of the inputs, or using the mean points of random data partitions as the initial seeds.

Katsavounidis et al. (1994) propose an initialization method that selects k *diverse*

data points as cluster seeds. The point with the largest Euclidean norm is chosen as the initial seed. The i-th cluster seed $\mathbf{a}_i$ is then selected as follows:

$$\mathbf{a}_i = \arg\max_{\mathbf{x}} \left[\min_{j<i} dist(\mathbf{x}, \mathbf{a}_j) \right] \quad , \tag{4.22}$$

where $dist(\mathbf{x}, \mathbf{a}_j)$ is the distance between $\mathbf{x}$ and $\mathbf{a}_j$ according to the distance function used for clustering. The point which is most different from the existing seeds is preferred. Using this initialization method, the first k-means iteration is practically equivalent to MaxMin clustering (Gonzalez, 1985). This method performed well in the comparative study by He et al. (2004) and, for large datasets, also in (Steinley and Brusco, 2007). A similar method is described by Mirkin (2005, p. 88), except that they suggest using the most distant pair of points as the initial two centroids. However, this requires the computation of all pairwise distances which is prohibitively expensive for large datasets. Probabilistic variants of the Katsavounidis method were also proposed (Ostrovsky et al., 2006; Arthur and Vassilvitskii, 2007), in which the probability of selecting a data point as the next cluster seed is proportional to its distance to the closest existing seed.

Density-based initialization methods aim to select initial points from dense areas in the input space. For example, the method proposed by Kaufman and Rousseeuw (1990) selects points having many nearby examples, with the additional constraint that these nearby examples are closer to the candidate point than to any of the existing seeds. A similar approach can be found in earlier work by Astrahan (1970). Note that such density-based initialization methods are computationally much more expensive than the k-means algorithm itself since they require all pairwise distances between points.

The k-means algorithm can also be initialized using the mean points of some "best-guess" partition of the data. A deterministic clustering algorithm, such as hierarchical agglomerative clustering, can be used to generate this initial partition (Milligan, 1980), possibly using only a sample of the original data (e.g. Meilă and Heckerman, 1998). An alternative is to initially use a top-down divisive partitioning of the data, for example, according to values of the attribute with the greatest variance (Su and Dy, 2007), or by using hyperplanes perpendicular to the first principal component (Boley, 1998; Su and Dy, 2007).

4.5.2 Experimenal Setup

We evaluated the effect of initializing the k-means clustering method using the representative sample generated by CERS, and compared this approach to the Forgy (random data points) and Katsavounidis initialization methods. These two methods were chosen for comparison because they share a key feature with CERS: Initial cluster centroids

correspond to actual data points in the dataset. The computational cost of the Katsavounidis method is also comparable to CERS. Since we used normalized TF-IDF vectors in our experiments, the initial seed for the Katsavounidis method was selected randomly.[8]

We used dot products of TF-IDF vectors as the similarity measure for determining cluster membership, as is customary in text mining. Although document vectors were normalized, we did not renormalize centroids, which were simply averages of all document vectors in the cluster, following Steinbach et al. (2000). Clusters were iteratively refined as long as the cluster membership of more than one document was updated in each iteration.

We used the class labels of documents as a gold standard for cluster quality. The quality of clusters was measured using conditional entropy (Dom, 2001) – the remaining uncertainty in determining the class label of a document after the cluster assignment is revealed:

$$\sum_{i=1}^{k} \frac{|\mathrm{K}_i|}{|\mathrm{D}|} \sum_{y \in C} P(y|\mathrm{K}_i) \log_2 \frac{1}{P(y|\mathrm{K}_i)} \quad . \tag{4.23}$$

In this expression, D is the dataset, K_i are the clusters returned by k-means ($i = 1 \ldots k$), and C is the set of class labels in the gold standard. The probability $P(y|\mathrm{K}_i)$ of a class label y in each cluster K_i is measured as the relative frequency of data points in K_i labeled with y.

It is interesting to study the degree to which cluster centroids in the converged k-means result deviate from the initial seeds for various initialization methods. To this end, we generated k ranked lists of all data points, ordered by their similarity to each of the k final cluster centroids. Let $\mathrm{rank}_i(\mathbf{x})$ denote the rank of data point $\mathbf{x}$ in the similarity list corresponding to cluster i. In the following section, we report the mean rank of the initial cluster seeds $\mathbf{c}_i$ in the ranked lists of clusters they spawned:

$$\frac{1}{k} \sum_{i=1}^{k} \mathrm{rank}_i(\mathbf{c}_i) \quad . \tag{4.24}$$

We refer to this measure as *centroid drift* and use it to measure the distance between the initial centroids and the final solution found by k-means.

For each dataset, we report two sets of results based on a random stratified 10-fold split of the data:

[8] We note that Katsavounidis et al. (1994) suggest the use of the item with the largest Euclidean norm as the initial seed, however, our TF-IFD vectors were normalized before clustering. We also tried using the document with the largest norm of its unnormalized TF-IDF vector as the first seed, but this did not improve the results of the Katsavounidis method on any dataset, and degraded results in some cases.

– results obtained by clustering each of the 10 disjoint data folds separately;

– results obtained by clustering the entire dataset after selecting cluster seeds from only one of the 10 folds.

The second set of experiments facilitates comparison of clustering quality with other studies using the same data, but handicaps the initialization methods, since initial seeds can be chosen only from a subset of the dataset being clustered. K-means clustering using various initialization methods was tested for values of k ranging from $k = 2$ to $k = 20$. For each measurement, a t-test was conducted on the 10 paired results obtained using different k-means initialization methods.

4.5.3 Results and Discussion

Cluster quality for different initialization methods is depicted in Figure 4.2. To assess the "information gain" of the clustering solution, the conditional entropy in the plots can be compared to the label entropy of each dataset in Table 4.1.

The results indicate that data points selected by CERS are indeed useful for cluster initialization, particularly in the first set of experiments, where each data fold is clustered independently. The improvement over other methods is almost always significant in this case. When clustering whole datasets, cluster quality improves for all initialization methods. This is to be expected due to the increased data redundancy when clustering whole datasets: The goal of both experiments is to find clusters in the same underlying population, but the sample of this population is ten times smaller in the first experiment. Clustering whole datasets also appears to reduce the sensitivity of k-means to the choice of initial centroids, and differences between the various initialization methods are smaller in this experiment. To some extent, this can be attributed to the design of the experiment, in which the cluster initialization methods may only consider a random subset of the data when selecting seeds. While CERS still outperforms other methods in this experiment, the difference is significant in only 30% of the measurements.

We note that the Katsavounidis method is found to perform rather badly on the Newsgroups dataset while CERS performs comparatively well on the same data. We believe this is because the Newsgroups dataset contains many "pathological" outliers (e.g. Usenet posts containing only a few words and posts containing fragments of ASCII-encoded binary attachments). The Katsavounidis method is prone to selecting such outliers because seed diversity is its primary selection criterion. This notion is corroborated in the centroid drift measurements.

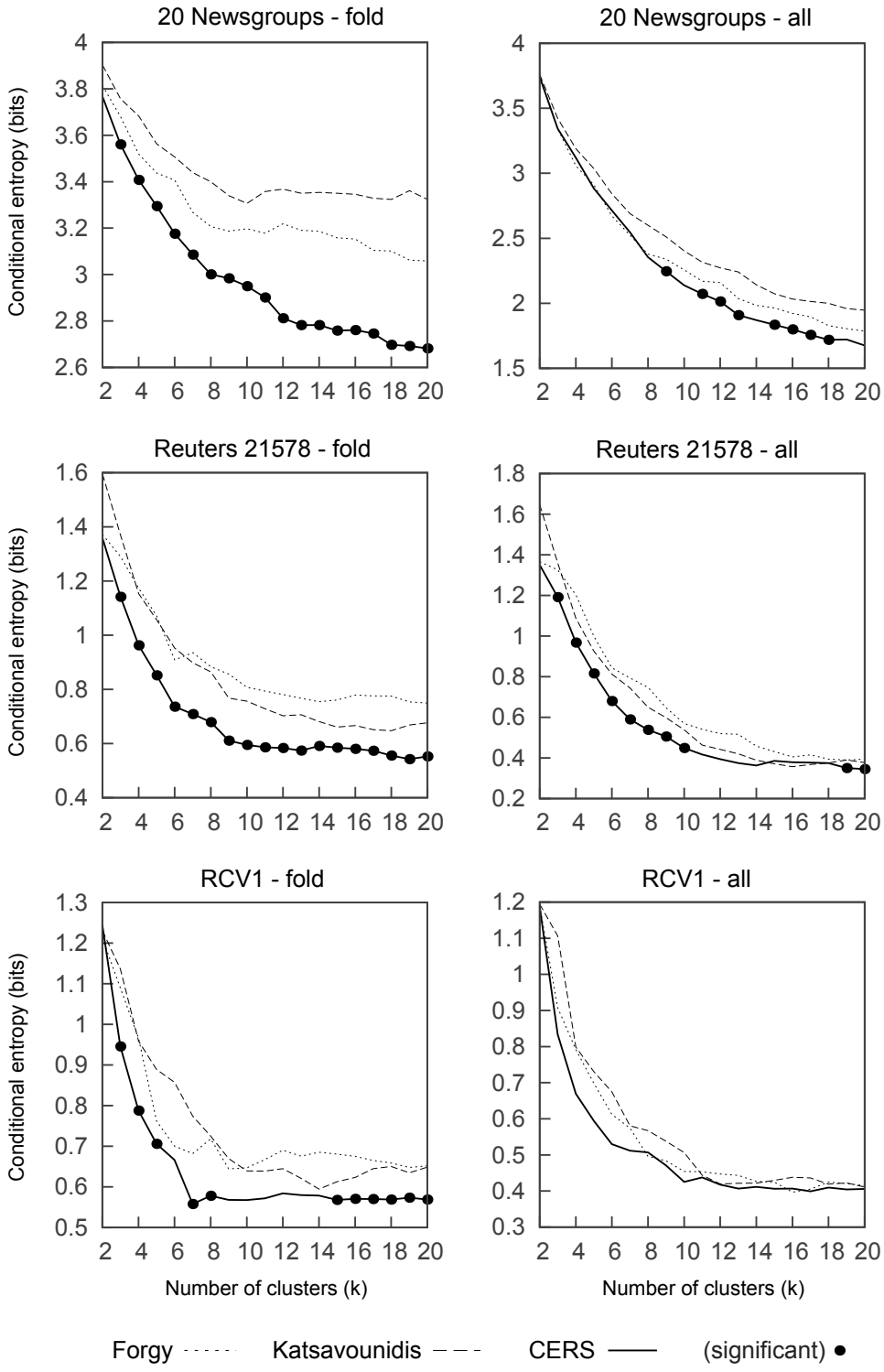


Figure 4.2: Clustering quality for $k = 2$ to $k = 20$, measured as the conditional entropy (in bits) of the gold standard classification given the cluster membership of each data point. Results for clustering each disjoint data fold separately are shown in the left column. Results for clustering the entire dataset after seeding from one of the data folds are shown in the right column. The '•' symbol marks runs for which the best-performing initialization method achieves significantly better results compared to both other methods ($p < 0.05$, one-tailed t-test).

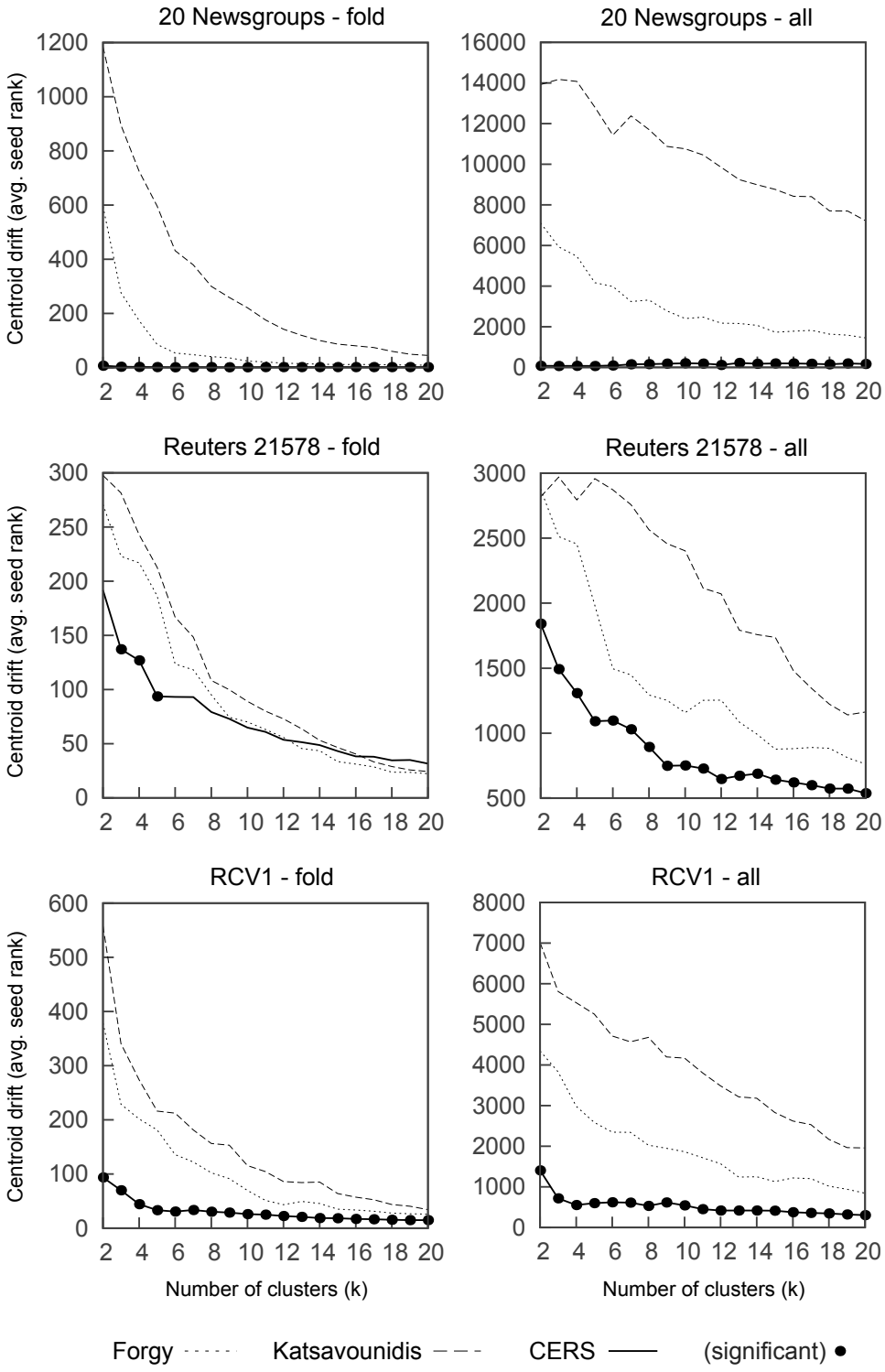


Figure 4.3: Centroid drift for values of k ranging from $k = 2$ to $k = 20$. Results for clustering each disjoint data fold separately are shown in the left column. Results for clustering the entire dataset after seeding from one of the data folds are shown on the right. The '•' symbol marks runs for which cluster seeds that are selected by one initialization method are by rank significantly "closer" to the respective centroids after k-means converges ($p < 0.05$, one-tailed t-test; compared to both other methods).

The graphs in Figure 4.3 show average centroid drift for various initialization methods. We find that the initial cluster centroids selected by CERS are closer to the converged k-means result, supporting our hypothesis that samples produced by CERS tend to contain cluster representatives. On the other hand, seeds produced by the Katsavounidis method are far from the final cluster centroids, which is to be expected due to the method's intrinsic preference for distant data points. Comparing the measurements for clustering each data fold independently and clustering whole datasets, we find that centroid drift increases roughly ten-fold in the second set of experiments. Since the number of data points that are clustered is also increased by a factor of ten in these experiments, an increase of this magnitude would also be expected if the rankings used to compute centroid drift according to Equation (4.24) were random.

In general, centroid drift is significantly smaller for CERS compared to other methods in almost all cases. The centroid drift measurements are particularly striking for the Newsgroups dataset. An exception is the Reuters 21578 dataset, but only when clustering each data fold independently at larger values of k. We are unable to explain this slight inconsistency.[9]

4.6 Applications: Active Learning

With large amounts of textual data easily accessible on the web, the cost of manually labeling texts to produce training examples for classification is typically much greater than the cost of the data collection effort. Pool-based active learning methods suggest which data points should be labeled in order to best exploit limited resources that are available for manual annotation. Training examples are usually selected incrementally, and it is assumed that labels for previously selected training examples are available to the active learner when selecting the next training example. For a recent review of this field, see (Settles, 2012).

We evaluated a straight-forward "unsupervised" application of CERS for active learning; an unconventional approach in which we do not consider document labels at all. The idea is to select a sample that is generally the most representative of the available data. We expect that such a sample will cover the most common patterns to be found in the data pool. We also expect that such a sample will not contain outliers or redundant data points, thereby maximizing the informativeness of each labeled training example.

This approach runs the risk of sampling from dense areas that are far from the

[9] To test whether the result was obtained by chance, we repeated the Reuters clustering experiments using a fresh 10-fold split. The experiment produced virtually the same results.

classification boundary. Labeling such examples may be unnecessary if, based on previous training data, we are already sufficiently confident of their classification. A more fruitful approach would be to select representative examples that lie near the classification boundary. To this end, we consider a simple modification of the unsupervised CERS algorithm. The "near-boundary" CERS variant selects the next training example from a limited number of most uncertain points according to the current classifier. The pseudo-code for this procedure is given in Algorithm 2.

Algorithm 2: Near-boundary version of the CERS algorithm for active learning.

Input : Unlabeled data pool D.
Input : Current labeled dataset S and classifier $\hat{\theta}_{\mathrm{S}}$ trained on S.
Input : Number of uncertain examples to consider n.
Input : Parameters required to compute ΔH: the model order d, zero-order estimator routine $p_e(\cdot)$, and the CTW weighting parameter α.
Output : Next example $\mathbf{q} \in \mathrm{D}$ for which to request a label.

$\mathrm{D}^* \leftarrow \mathrm{D}$;
if *at least one positive and one negative example in* S **then**
  /* Compute the uncertainty $u_{\mathbf{x}}$ of each example x */
  /* Maximum uncertainty is indicated by $y(\mathbf{x}|\hat{\theta}_{\mathrm{S}}) = 0$ */
  foreach $\mathbf{x} \in \mathrm{D}$ **do** $u_{\mathbf{x}} \leftarrow |y(\mathbf{x}|\hat{\theta}_{\mathrm{S}})|$;
  /* Find n most uncertain examples */
  $\mathrm{D}^* \leftarrow$ top n items in D, sorted by increasing $u_{\mathbf{x}}$;
end
$\mathbf{q} \leftarrow \arg\max_{\mathbf{x} \in \mathrm{D}^*} [\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})]$;

The near-boundary CERS variant selects the same data points as the unsupervised CERS method until at least one positive and one negative example are found. In each subsequent iteration after this point, the method trains a classifier $\hat{\theta}_{\mathrm{S}}$ based on the current set of labeled documents S, and uses this classifier to rank all documents in the current unlabeled pool D by the uncertainty of their class labels according to $\hat{\theta}_{\mathrm{S}}$. If $y(\mathbf{x}|\hat{\theta}_{\mathrm{S}})$ is a binary classification score with a 0 (zero) classification boundary for the classifier $\hat{\theta}_{\mathrm{S}}$ (e.g. the log odds of the positive class), then ranking by uncertainty is equivalent to sorting $\mathbf{x} \in \mathrm{D}$ by increasing absolute classification score $|y(\mathbf{x}|\hat{\theta}_{\mathrm{S}})|$. The near-boundary CERS method selects a subset of n most uncertain points $\mathrm{D}^* \subseteq \mathrm{D}, |\mathrm{D}^*| = n$, from the current pool of unlabeled documents D, so that: $\forall \mathbf{u}, \mathbf{v} \in \mathrm{D} : \mathbf{u} \in \mathrm{D}^* \wedge \mathbf{v} \notin \mathrm{D}^* \Rightarrow |y(\mathbf{u}|\hat{\theta}_{\mathrm{S}})| \leq |y(\mathbf{v}|\hat{\theta}_{\mathrm{S}})|$. The next example to be labeled is the document $\mathbf{x} \in \mathrm{D}^*$ from this subset which maximizes the reduction in cross-entropy $\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})$ when $\mathbf{x}$ is added to the training set S. The number of uncertain points n is a parameter of the method.

4.6.1 Related Work

Given a pool of unlabeled documents and a labeled training set from which a classifier can be trained, a popular approach is to select the data point for which the label is most uncertain according to the current classifier. For a binary classification problem, determining the most uncertain example is straight-forward (e.g. score closest to $\frac{1}{2}$, if the classifier returns the probability of the positive class). This method, named uncertainty sampling, was first proposed for text classification by Lewis and Gale (1994). For linear classifiers, uncertainty sampling also has a geometric interpretation – the selected data point lies closest to the current decision hyperplane (Schohn and Cohn, 2000; Tong and Koller, 2001).

It has been argued that uncertainty sampling is prone to selecting outliers, since the density of data in the proximity of an example is not considered by the method. Labeling outliers may lead to improvements of the classification hypothesis for "unimportant", scarcely populated parts of the input space. Several methods which explicitly address this problem can be found in the literature. For example, Xu et al. (2003) propose clustering examples that lie within the margin of an SVM classifier. Data points that are cluster representatives (medoids) are selected by the method. Shen and Zhai (2003) propose a similar approach in the relevance feedback framework: They cluster the top n most relevant initial results and seek user feedback for the cluster medoids. The density-weighted query-by-committee method of McCallum and Nigam (1998b) combines the disagreement of a classifier ensemble about the label of an example (a measure conceptually similar to uncertainty) and the average distance of the example to other points in the dataset (a measure of data density). Nguyen and Smeulders (2004) combine data density and label uncertainty in a soft clustering framework. Uncertainty is measured at the cluster level, and representative examples of dense clusters which lie near the classification boundary are selected for labeling. The motivation for our near-boundary CERS method is the same as in other studies mentioned in this paragraph – to avoid outliers when sampling uncertain data points.

Many other active learning strategies exist besides uncertainty sampling, although their application is often limited to certain types of data (e.g. noiseless data), or certain types of classifiers (e.g. probabilistic classifiers). For completeness, we briefly mention some of these approaches. A classical approach is to label the points where there is disagreement among classification hypotheses that are all "consistent" with the existing training data. If the data is noiseless, this approach guarantees reducing the version space of possible hypotheses with each training example (cf. Seung et al., 1992; Cohn et al., 1994; Freund et al., 1997). Information-based criteria for active data selection are

discussed by MacKay (1992), specifically for Gaussian regression models. Cohn et al. (1996) suggest labeling points that are expected to reduce the learner's variance, provided that this objective function can be expressed in closed form. Roy and McCallum (2001) describe an active learning method for probabilistic classifiers, which directly estimates the reduction in classification error over the unlabeled pool after a particular example is labeled. When classification error is measured using log loss, this amounts to minimizing the uncertainty (entropy) of the prediction model's outputs given the unlabeled data. Anderson and Moore (2005) theoretically develop information-based and misclassification-cost-based objective functions for various learning tasks related to active learning with hidden Markov models.

Many of the previously mentioned studies deal with information-based objective functions for active learning (e.g. MacKay, 1992; Anderson and Moore, 2005; Roy and McCallum, 2001). In these studies, information-based criteria are used to optimize certain properties of the prediction models being trained, e.g. to reduce the uncertainty of the model parameters or the uncertainty of the model's predictions over the test set. Our cross-entropy reduction method is also information-based, however, our active learning approach is fundamentally different. We aim to measure how well the input space (data pool) is represented by the training set, without regard for the outputs (class labels), and independently from the classification model being trained. The near-boundary CERS variant loosely couples our method to the classification model, by limiting the selection of training examples to data points which are uncertain according to the current model. This works as a simple pre-filter which helps our method "focus" on areas that are most relevant for the classification problem at hand. It does not change the objective function by which instances are selected from the (reduced) pool of candidate points.

To conclude our literature review in this field, we mention the farthest-first heuristic, which was studied by Baram et al. (2004) in the broader framework of combining different active learning strategies. After picking an initial data point at random, successive queries are chosen which maximize the shortest distance between the new data point and any of the previously selected points. This approach is also independent of the classifier being trained, and is similarly motivated to our use of CERS in the active learning setting – to produce a training set which "covers" the data pool well.

4.6.2 Experimenal Setup

It is common to evaluate active learning strategies on binary classification tasks.[10] To this end, we constructed two binary classification problems from each of the datasets described in Section 4.4.1:

- We used two category pairs from the Newsgroups dataset: `comp.sys.ibm.pc` vs. `comp.os.ms-windows.misc`, and `comp.graphics` vs. `comp.windows.x`. These two binary datasets contain 1935 and 1967 documents, respectively. The same pairs of newsgroups were also used in previous active learning studies (Schohn and Cohn, 2000; Roy and McCallum, 2001).

- The `earn` and `acq` categories account for most of the 8085 documents in our version of the Reuters 21578 dataset (3931 and 2327 documents, respectively). We constructed two "one-vs-rest" classification tasks, using documents belonging to one of these categories as positive examples and all other documents in the dataset as negative examples. Both binary datasets thus contain 8085 documents.

- From the RCV1 dataset, we selected the following two pairs of categories: `science` vs. `weather` (6471 documents), and `entertainment` vs. `religion` (6199 documents).

We compared active learning using CERS with uncertainty sampling, a well understood active learning method which is widely used in practice, and random sampling (i.e. not using any active learning strategy). For active learning with uncertainty sampling, examples were selected at random until at least one positive and one negative example was found.

Although the number of uncertain points that are considered by the near-boundary CERS variant is a parameter of the method, we did not thoroughly explore its effect on performance. The main goal of this experiment is to determine whether any potential risk of selecting outliers by uncertainty sampling could be lowered using CERS, and to evaluate this effect by comparing near-boundary CERS to the standard uncertainty sampling method. To this end, we did not wish to deviate too much from the original uncertainty sampling principle, and kept the parameter fixed at a very conservative value of 10 most uncertain points.

For each binary classification task, we first selected 1000 documents at random to create the pool from which examples could be selected for labeling. The remaining documents were withheld from the active learning method and served as an independent

[10] There are a number of reasons for this, the foremost perhaps being that binary classification tasks are "better behaved" at extremely small training set sizes. This also allows the use of binary classifiers (e.g. SVM) without further complicating the design of experiments.

test set. The classifier was retrained after each new example. Each such experiment was repeated 10 times using different random splits of the data and all active learning methods were tested on each of the 10 data splits. To determine statistical significance of each measurement, we conducted t-tests on the 10 paired results achieved by random sampling, uncertainty sampling, and each of the CERS variants.

Active learning methods were evaluated in combination with SVM (Vapnik, 1995) and naive Bayes (see e.g. Lewis, 1998) text classifiers. SVM classifiers are widely regarded as the state-of-the-art in terms of text categorization performance (Joachims, 1998b; Yang and Liu, 1999). The naive Bayes classifier is also commonly used for text categorization, mainly due to its simplicity, computational efficiency, and relatively solid performance, particularly when little data is available for training (Forman and Cohen, 2004). We used a linear kernel for the SVM, as is customary when dealing with textual data. As in previous experiments, documents were represented with normalized TF-IFD vectors. The regularization parameter was set to $C = 1$, the default setting for SVM$^{\text{light}}$ – the SVM solver we used (Joachims, 1998a). We did not use a bias term since this was found to improve performance for small training sets (this applies equally for all of the active learning methods). Our implementation of naive Bayes is based on the multinomial event model, as described by McCallum and Nigam (1998a). All features found in the training set were used for classification with naive Bayes. Word probabilities were smoothed using the Laplace estimator.

4.6.3 Results and Discussion

Learning curves for SVM and naive Bayes classifiers trained on examples selected by various active learning methods are given in Figures 4.4 and 4.5. The learning curves depict the accuracy of the classifier at particular training set sizes. The accuracy is averaged over 10 runs obtained from the 10 data splits. For datasets where the difference between the top-performing methods becomes small as the number of training examples increases, we also plot the odds of predicting the correct class label, i.e. $(\frac{x}{1-x})$, where x is the accuracy of the method ($0 \leq x \leq 1$). The average accuracy achieved after training on 100 examples selected by uncertainty sampling and near-boundary CERS is reported in Table 4.3. For comparison, we also report accuracy after training on the entire pool of examples which were available for active learning in Table 4.3.

We first remark on the overall performance of naive Bayes and SVM classifiers with and without an active learning strategy. The SVM classifier outperforms naive Bayes by a substantial margin when trained on the full data pool (see Table 4.3). However, the learning curves of naive Bayes are very competitive on the Reuters 21578 and RCV1 datasets (see Figures 4.4 and 4.5). The SVM classifier still dominates on

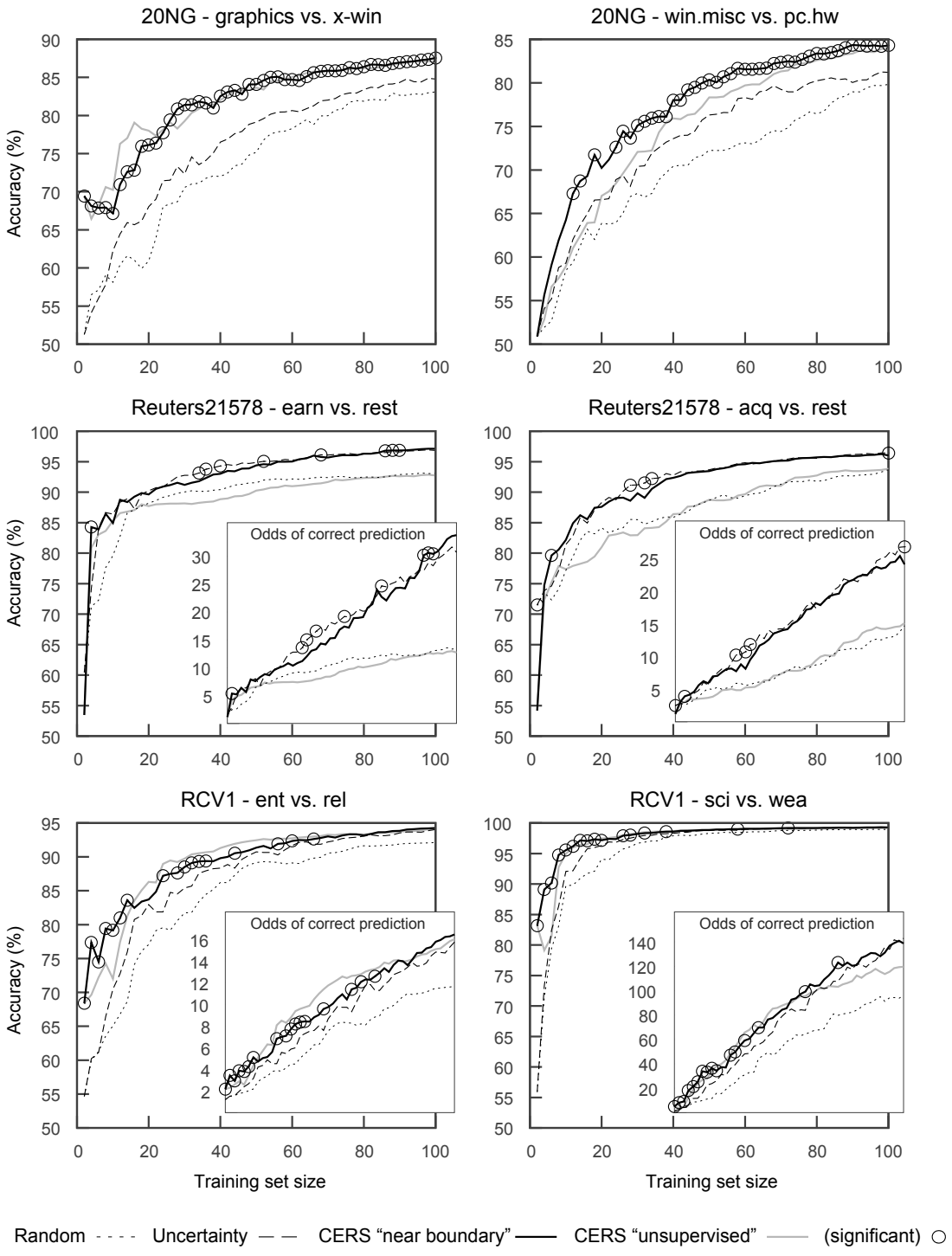


Figure 4.4: **SVM**; Average accuracy (%) at training set sizes ranging from 2 to 100 using different active learning methods. The '○' symbol marks significantly higher performance of one of the methods compared to both competing methods ($p < 0.05$, one-tailed t-test; "unsupervised" CERS not considered in the comparison).

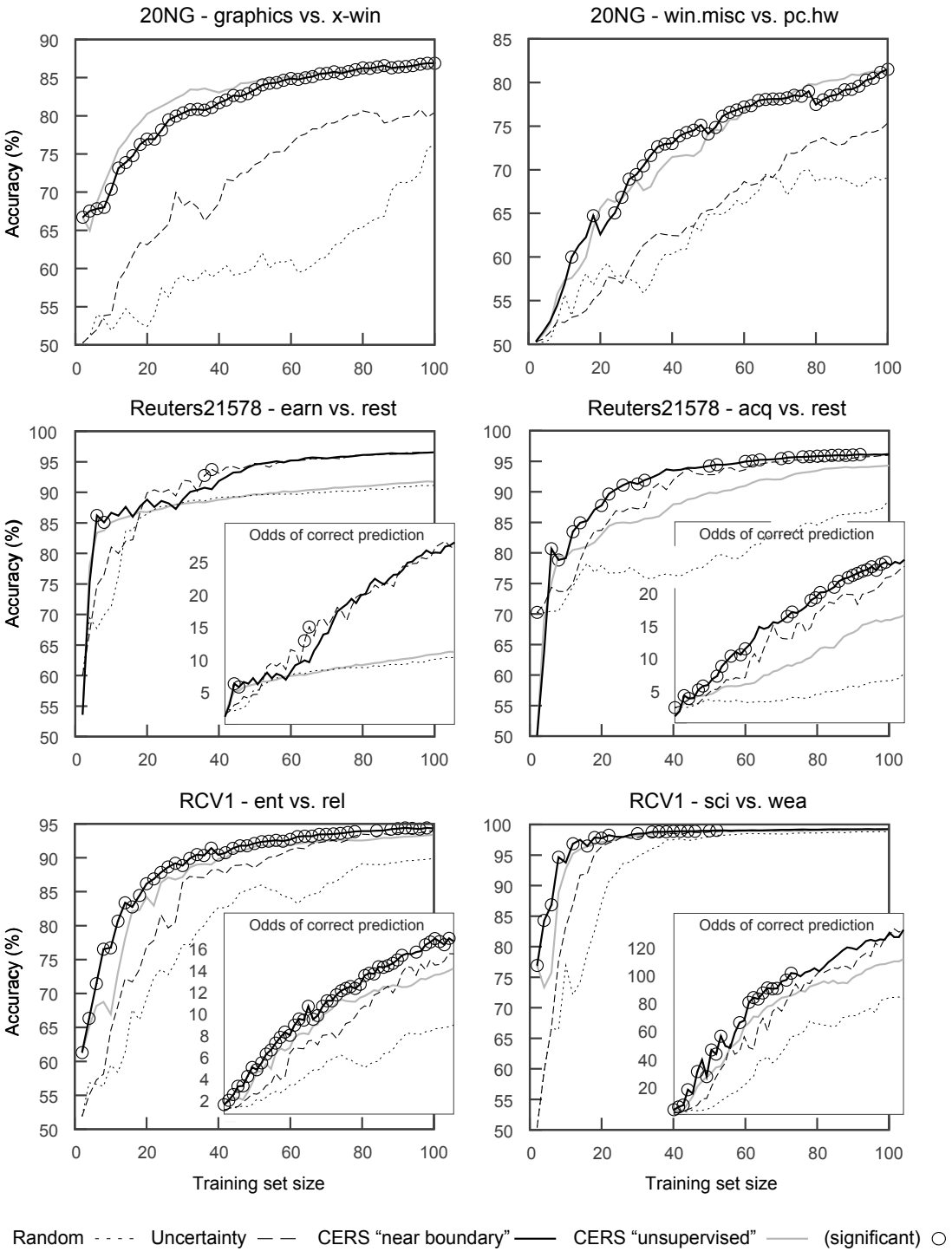


Figure 4.5: **Naive Bayes**; Average accuracy (%) at training set sizes ranging from 2 to 100 using different active learning methods. The '○' symbol marks significantly higher performance of one of the methods compared to both competing methods ($p < 0.05$, one-tailed t-test; "unsupervised" CERS not considered in the comparison).

SVM

Dataset and classification task	Majority classifier	Full training	Uncertainty sampling	CERS near-boundary
20NG: `graphics` vs. `windows.x`	50.18	93.34	84.78	**87.54**
20NG: `os.ms-win` vs. `sys.ibm.pc`	50.08	90.45	81.20	**84.29**
Reuters21578: `earn` vs. `rest`	51.38	97.27	96.85	**97.13**
Reuters21578: `acq` vs. `rest`	71.22	97.52	**96.42**	96.03
RCV1: `entertainment` vs. `religion`	57.12	96.71	94.03	**94.27**
RCV1: `science` vs. `weather`	62.22	99.49	**99.31**	99.28

Naive Bayes

Dataset and classification task	Majority classifier	Full training	Uncertainty sampling	CERS near-boundary
20NG: `graphics` vs. `windows.x`	50.18	90.75	80.44	**86.89**
20NG: `os.ms-win` vs. `sys.ibm.pc`	50.08	85.71	75.35	**81.51**
Reuters21578: `earn` vs. `rest`	51.38	94.23	**96.43**	96.55
Reuters21578: `acq` vs. `rest`	71.22	96.25	96.00	**96.16**
RCV1: `entertainment` vs. `religion`	57.12	95.25	93.93	**94.33**
RCV1: `science` vs. `weather`	62.22	99.24	**99.25**	**99.25**

Table 4.3: Average test set accuracy (%) after training on 100 examples selected by uncertainty sampling and near-boundary CERS, compared to the average test set accuracy obtained after training on the entire pool of 1000 examples which were available for active learning, and the majority class selection rule.

the Newsgroups data even when few training examples are available (particularly for the `comp.os.ms-windows.misc` vs. `comp.sys.ibm.pc` task). Our experiments also suggest that naive Bayes appears to benefit more from using an active learning strategy instead of choosing training examples at random.

Active learning using unsupervised CERS

Unsupervised CERS substantially outperforms the baseline approach of selecting training examples at random in all of our experiments on the 20 Newsgroups and RCV1 datasets. However, when using the SVM classifier on the Reuters 21578 dataset, the unsupervised CERS method fails to improve upon the random sampling baseline. Using naive Bayes, CERS performance is again comparable to that obtained by random sampling for the `earn` category, and better than random sampling for the `acq` category.

Compared to uncertainty sampling, the unsupervised CERS variant shows mixed results for different datasets. It excels in both classification tasks based on the 20 Newsgroups dataset, but is considerably worse than uncertainty sampling when applied to the Reuters 21578 data. In experiments based on the RCV1 dataset, unsupervised

CERS tends to perform better than uncertainty sampling in the initial parts of the learning curves and becomes worse as the number of training examples increases. These results are consistent for both naive Bayes and SVM classifiers.

Several authors report that the Reuters 21578 dataset can be classified extremely well using a relatively small set of carefully chosen keywords, to the extent that some categories can be determined almost perfectly based on the (non)occurrence of a single word (cf. Frank et al., 2000; Bekkerman et al., 2003). The extensive analysis by Bekkerman et al. (2003) shows that the Reuters 21578 and 20 Newsgroups datasets differ greatly in this respect, in the sense that a large vocabulary is required to classify the Newsgroups data well. We suspect this might contribute towards the sub-par performance of CERS on the Reuters 21578 data. Intuitively, the CERS method selects examples that are representative of the dataset in an "unsupervised" way, in that all potential features are treated on equal footing. If there are relatively few highly informative features, these may be overwhelmed by other patterns that exist in the data, but are less useful for the particular classification problem at hand. Uncertainty sampling selects documents which contain combinations of features that remain uncertain according to the current classification hypothesis – a strategy which may be more suitable in the scenario just described.

A major advantage of the unsupervised CERS method is that it does not require labels for previously selected training examples, and can therefore be used to produce a training set offline without human intervention. In contrast, uncertainty sampling can proceed only after each selected training example is labeled, a requirement which is typical of almost any active learning method. Such close interaction with a human annotator is cumbersome to implement in practice, and time lost due to the (in)efficiency of the active learning method should also be considered in this setting. Having multiple annotators working in parallel further complicates this human-in-the-loop scheme. Since our method is unsupervised and does not require labeling of examples, we argue that it is perhaps more fairly compared to random selection of training examples than to methods such as uncertainty sampling, which require feedback.

Active learning using near-boundary CERS

We find that the near-boundary CERS variant (a combination of CERS and uncertainty sampling) notably improves upon unsupervised CERS for datasets where uncertainty sampling performs well. This is most visible in experiments based on the Reuters 21578 data. Also surprising is the finding that near-boundary CERS performs as well or better than unsupervised CERS on *all* datasets, including those for which uncertainty sampling does not perform too well. A close inspection of the graphs in Figures 4.4

and 4.5 reveals limited learning curve intervals where this is not the case. Without exception, these areas correspond to measurements where unsupervised CERS performs *much* better than uncertainty sampling, resulting in a slight decrease in performance when the methods are combined.

Near-boundary CERS substantially outperforms uncertainty sampling in 9 of the 12 experiments that were conducted. For experiments using the SVM classifier, uncertainty sampling has a slight edge over near-boundary CERS on the Reuters 21578 data. However, the learning curves intersect many times and the absolute difference in performance between the two methods is minimal in these experiments. Also, no clear winner emerges when using naive Bayes on the Reuters 21578 `earn` category.

A summary of active learning results

A summary of the relative performance of methods that were considered in our active learning experiments is given in Table 4.4. When comparing the results of two methods for a particular experiment, one method was considered superior if the resulting mean classification accuracy was greater for 75% (or more) of the measurements in the learning curve. A method was considered "substantially" superior if it achieved greater accuracy in 95% of the learning curve, with the difference being significant in 50% of the measurements or more according to a one-tailed paired t-test ($p < 0.05$).

	CERS unsupervised *vs.*				CERS near-boundary *vs.*			
	Random		*Uncertainty*		*Random*		*Uncertainty*	
Dataset and classification task	SVM	NB	SVM	NB	SVM	NB	SVM	NB
20NG: `graphics` vs. `windows.x`	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
20NG: `os.ms-win` vs. `sys.ibm.pc`	$\gg$	$\gg$	$>$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
Reuters21578: `earn` vs. `rest`	$<$	$\sim$	$\ll$	$<$	$\gg$	$\gg$	$\sim$	$\sim$
Reuters21578: `acq` vs. `rest`	$\sim$	$\gg$	$\ll$	$<$	$\gg$	$\gg$	$\sim$	$\gg$
RCV1: `entertainment` vs. `religion`	$\gg$	$\gg$	$\gg$	$\sim$	$\gg$	$\gg$	$>$	$\gg$
RCV1: `science` vs. `weather`	$\gg$	$\gg$	$\sim$	$\sim$	$\gg$	$\gg$	$>$	$>$

Table 4.4: A qualitative summary of the active learning performance of both CERS variants compared to uncertainty sampling and random selection of training examples. The $\sim$ symbol indicates comparative performance. The symbols $>$ and $<$ respectively indicate better/worse performance of CERS relative to the competing method. The $\gg$ and $\ll$ symbols indicate that the difference in performance is "substantial".

The results presented in this section support the notion that CERS is a promising method for active learning, especially when the method is suitably adapted in order to sample points near the classifier's decision boundary. We have by no means explored the full range of possibilities in this application domain, and it is reasonable to expect that a more intricate combination of CERS and uncertainty sampling could yield further improvements. We believe this to be a promising path for future research.

4.7 Applications: News Mining

In this section, we present the results of a case study in which we used CERS to identify important articles in a historical stream of broadcast news. Given a chronological news stream segmented into consecutive time intervals, we wish to find stories that are "most informative" of events that were reported. The result is a timeline containing representative articles selected by CERS for each time interval, as well as the most significant keywords for each article, i.e. words that most contributed towards an article being selected. Although we do not formally evaluate the performance of CERS in this application domain, we do compare our results to an independent reference list of important news events from the same time period.

In the analysis, we mostly focus on the contents of articles that were identified by the method, in order to give some insight into how the method "actually works" by addressing the following pertinent questions:

- Why is a particular article selected by the CERS method; Which "features" contribute most towards a high score for a particular article?
- How do articles that were selected in previous iterations affect the selection of the next representative article? Do successive articles contain *different* information?

The experiment also serves to illustrate a property of CERS which was not yet discussed: Its ability to find texts which contain new information with respect to some background knowledge. This property arises naturally from the cross-entropy reduction criterion, that is, the preference for articles which *improve* our ability to compress the remainder of the data. This improvement is always relative to other articles in the representative set. If the representative set is primed is such a way that it includes some background data, patterns that are adequately represented in the background data will not contribute towards a reduction in cross-entropy. In the news mining application, we wish to find a small subset of articles that best describe *new* developments, i.e. events that actually occurred in the time period of interest, and have not been reported before. To this end, we prime the representative set to include all news stories published before the target time period, as if those stories had already been selected in previous iterations of the CERS algorithm.

Our CERS-based method for identifying representative news articles is formalized in Algorithm 3. As input, it accepts chronologically ordered sets of documents $\mathrm{D_i}$, $i = 1 \ldots n$, containing articles published in each of n successive time intervals. In our experiments, time intervals were defined as calendar months. When selecting articles for a particular month, the representative sample S in the CERS algorithm was primed

so that it contained all articles published in previous months. In this way, articles were scored by how much they reduce the remaining uncertainty of "current news" after the prediction model is trained on "old news". As discussed previously, this favors articles which report new information with respect to news published in previous months. When scoring articles, the cross-entropy reduction objective function was measured against the combined volume of news published in the same calendar month as the article, as well as news published in the following month (see Algorithm 3). This "lookahead" handles possible cases in which a story breaks at the end of a calendar month while the bulk of the related news coverage materializes in the following month. In such cases, the significance of the story could be underestimated if it were assessed only against other news published in the same time interval. Due to the use of past and future data when searching for representative articles in a particular month, results are given only for those time intervals for which such past and future data exists in the RCV1 corpus.

Algorithm 3: CERS-based news mining.

Input : Chronologically ordered doc-sets $\mathrm{D_i}$, $i = 1 \ldots n$ (one per month).
Input : Number of representative documents to select from each set $m > 0$.
Input : Parameters required to compute ΔH: the model order d, zero-order estimator routine $p_e(\cdot)$, and the CTW weighting parameter α.
Output : Representative documents $\mathrm{S_i}$ from each set $\mathrm{D_i}$, for $i = 2 \ldots n-1$.

for $i \leftarrow 2$ **to** $n-1$ **do**
  $\mathrm{S} \leftarrow \bigcup_{j<i} \mathrm{D_j}$; `/* Set S to be the union of` $\mathrm{D_j}$ `, for` $j < i$ `*/`
  $\mathrm{D} \leftarrow \mathrm{D_i} \cup \mathrm{D_{i+1}}$; `/* Set D to be the union of` $\mathrm{D_i}$ `and` $\mathrm{D_{i+1}}$ `*/`
  $\mathrm{S}_i \leftarrow \emptyset$; `/* Initialize` S_i `*/`
  while $|\mathrm{S}_i| < m$ **do**
    `/* Select next representative document s from` $\mathrm{D_i}$ `*/`
    $\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \mathrm{D}_i} [\Delta H(\mathbf{x}, \mathrm{D}, \mathrm{S})]$;
    `/* Add s to` $\mathrm{S_i}$ `, and move s from D to S */`
    $\mathrm{S_i} \leftarrow \mathrm{S_i} \cup \{\mathbf{s}\}$;
    $\mathrm{S} \leftarrow \mathrm{S} \cup \{\mathbf{s}\}$;
    $\mathrm{D} \leftarrow \mathrm{D} \setminus \{\mathbf{s}\}$;
  end
end

4.7.1 Related Work

The Topic Detection and Tracking (TDT) initiative (Allan et al., 1998; Allan, 2002) is perhaps the most systematic and organized research effort related to broadcast news mining. The TDT series of workshops, held annually from 1998 to 2004, has produced

systematic evaluations of methods for various news mining tasks. The TDT evaluations were mainly focused on clustering documents into groups of articles which discuss the same news event[11], either as an online task, in which case articles are assigned to event clusters (or spawn a new cluster) as they arrive, or as a retrospective task, in which case the entire corpus was available prior to clustering.

Although finding "important" stories was not a subject of the TDT evaluations, the size of an event cluster produced by a TDT system is a natural measure of the news event's significance (cf. Yang et al., 1998). Del Corso et al. (2005) propose a more refined approach, in which the reputation of the contributing news sources is considered in addition to the size of a cluster formed around an event. Our CERS-based method does not make use of clustering in any way. However, we suspect that the representative articles identified by CERS will lie close to cluster centers, as demonstrated in the results of our clustering experiments (Section 4.5). We also expect that the representative articles selected by CERS will lie in dense areas, and therefore belong to large clusters.

Our approach is document-centric, in the sense that our primary goal is to construct a chronological timeline by identifying representative documents in the corpus. An alternative, word-centric approach is to extract individual words by analyzing the temporal distribution of words or phrases in the data stream. One might wish to identify time intervals in which a word/phrase occurs more frequently than would be expected (Swan and Allan, 1999; Kleinberg, 2002), or intervals of greatest rising or falling trends in word usage (Liebscher and Belew, 2003). A survey of such methods is given by Kleinberg (2006). In our analysis, we extract the most significant keywords for each selected document. It would be relatively straight-forward to adapt our method in order to produce a word-centric timeline, consisting only of influential keywords detected in the news stream.

Sophisticated news filtering systems combine methods from different fields such as text mining, information retrieval and document summarization. One of the most comprehensive (academic) systems for news aggregation and filtering is Newsblaster[12], which was developed at Columbia University (McKeown et al., 2002). A similar system – NewsInEssence – was developed at the University of Michigan (Radev et al., 2005). The various components which are used in such systems are typically developed, evaluated and refined independently. The case study presented in this section can be considered a preliminary investigation into using CERS as a potential building block for sophisticated news filtering system such as those previously mentioned.

[11] Segmentation of running text into news stories was also considered in TDT.
[12] Available at `http://newsblaster.cs.columbia.edu/`

4.7.2 Experimenal Setup

Our news mining experiments are based on the RCV1 corpus. As in previous experiments based on the RCV1 data, headline and paragraph elements of the original XML files were extracted to create plain-text documents. However, we did not pre-process the text of articles in order to increase the readability of the content which is presented in our analysis, except for replacing occurrences of the escape sequence '`&quot;`' with the double quote character '`"`', again in order to increase readability. Accordingly, we used a 256-valued alphabet for the prediction model and set the CTW weighting parameter α to its default value for $|\Sigma| = 256$, that is, $\alpha = \frac{|\Sigma|-1}{|\Sigma|} = \frac{255}{256} \approx 0.996$ (see Section 3.4.3, Eq. 3.18).

The RCV1 corpus is chronologically annotated and contains all news stories produced by Reuters in a one-year period between August 20, 1996 and August 19, 1997, making it ideal for our case study. We split the corpus into 13 subsets, one for each calendar month in the time period spanned by the data. We then used CERS to identify representative articles for each month between September 1996 and July 1997 (inclusive) according to Algorithm 3. For each selected article we also extracted keywords which most significantly contributed towards the article being selected. Keywords were extracted using the subsequence scoring approach described in Section 4.3.5, summing the scores of all characters in whitespace-delimited subsequences to produce the total score of each word in the article.

We focus on news timelines generated from the `science` category. This category contains a total of 2410 news stories in science and technology, of which 84 articles that were found to be identical duplicates were not used. This yields approximately 200 articles per month from which the most representative articles were selected.

4.7.3 Results and Discussion

The timeline in Figure 4.6 shows the most informative news articles from the RCV1 `science` category as determined by CERS. The timeline contains headlines of top-scoring documents for each month, except for Feb '97 and Jul '97, where we have displayed more than one article in order to showcase the diversity of successive documents selected by CERS. Each article is accompanied by a numeric score which reflects the relative importance of the document in comparison to other documents which could be selected in the same iteration of the algorithm. The relative importance is defined simply as the cross-entropy reduction score for the selected document divided by the median score over all candidate documents in the same iteration. The relative importance scores are also depicted graphically as gray circles with an area proportional to the relative

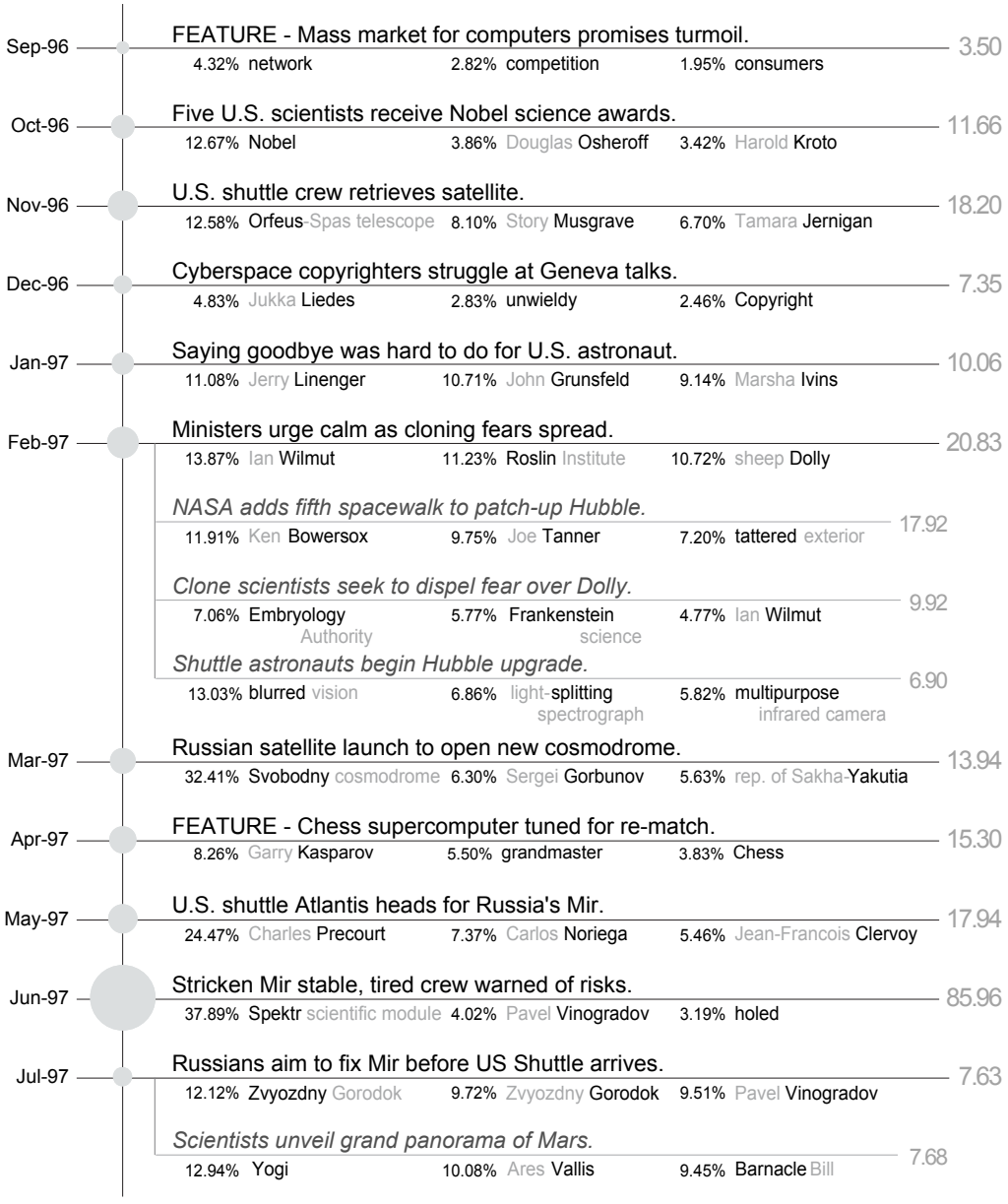


Figure 4.6: The top-scoring news stories extracted from the Science and Technology category of the RCV-1 corpus.

importance of the document. For each article, we show three most significant keywords, together with one or two context words from the text where suitable. The percentage points next to each significant keyword indicate the proportion of the document's cross-entropy reduction score occasioned by the first occurrence of the keyword.

We first try to assess whether the news articles identified by CERS are relevant for the time period in question. To this end, we compare the stories selected by CERS with CNN's editorial pick of ten most important science and technology stories in 1997[13] (we could not find a similar resource for 1996). The top six stories from CNN's list are presented in Table 4.5 – it so happens that the remaining four stories occurred after September 1997 and fall outside of the time period covered by the RCV1 data. Stories at positions 4 and 5 describe trends or topics that were popular during the entire year. They are not related to significant news events that would cause a burst of articles in a short timespan, and are therefore not likely to be picked up by our method (this is also not our intent). Of the remaining four stories which do correspond to breaking news events, we successfully identified three as the most important stories of the month. The Pathfinder landing on Mars – the first story in CNN's list – is the second story selected by CERS for Jul '97, slightly behind more news about the Mir drama. One of the reasons for this is that progress of the Pathfinder mission was covered throughout the news stream, which causes our method to underestimate the significance of the burst of Mars-related news occurring at the time of the July landing. This is manifested in the list of most significant keywords for the Pathfinder story – previously unreported names of features of the Martian landscape, rather than obvious keywords such as "`Pathfinder`" or "`Mars`".

Since important news events trigger large bursts of news reports, one could argue that reports about such events are most likely to be selected even if the selection process were random. To address this concern, we turn our attention to the keywords extracted for each article. Typically, a few keywords which are strongly related to a particular news story overwhelmingly contribute towards an article being selected. This indicates that articles are indeed selected because of the underlying story that is reported in the article. It is also interesting to note that the vast majority of the extracted keywords are named entities, although there is nothing in the design of our method that would favor named entities over other keywords. Using bursts of previously unseen or rarely mentioned named entities as an indicator of breaking news events has been suggested many times in the literature (Swan and Allan, 1999; Yang et al., 2002; Gabrilovich et al., 2004).

We find that for each particular time period, successive representative articles se-

[13] Available at `http://edition.cnn.com/SPECIALS/1997/year.ender/scitech/`

#	Date	Story headline and quotes
1.	Jul '97	***Earth Invades Mars!*** ...Pathfinder's arrival was heralded with more fanfare than any space event since, perhaps, Apollo 11 landed on the moon. Giddy scientists gave the rocks nicknames such as Barnacle Bill and Yogi...
2.	Feb '97	***Cloning Cloning*** The sheep hit the fan in February, when scientists at Scotland's Roslin Institute announced that they had cloned a ewe. Dolly, cloned from cells from another sheep's udder, became an overnight celebrity – and the center of a heated debate...
3.	Jun '97	***Misadventures of Mir*** ...Broken air conditioners, frequent computer crashes and a fender-bender... The most serious problem began June 25, when an unmanned cargo ship collided with Mir's Spektr module, cracking its hull...
4.	N/A	***Comet Hale-Bopp*** Stargazers of all levels of experience enjoyed the show this year... [The comet] was easily visible to the naked eye for much of the year...
5.	N/A	***Internet Growth*** As the Internet moved from buzzword to basic for many Americans, it remained in the news – but as the backdrop, not the focus...
6.	May '97	***Deep Blue beats Kasparov*** After decades of the mental struggle between man and machine, the machine finally won...

Table 4.5: CNN's editorial pick of most important science and technology stories in 1997 and the date on which the corresponding news event occurred (if applicable).

lected by CERS typically report different stories. Four articles are shown in the timeline for Feb '97. For this month, the top four articles are not as diverse, covering only two distinct stories. However, the extracted keywords for each article indicate that successive articles covering the same story are selected because they report different aspects of the story than previously selected articles. Except for the name "`Wilmut`", the keywords that most contribute towards the third and fourth article being selected by CERS do not occur in the first two articles for Feb '97. It can also be seen that the relative importance of the first article on a particular story is much higher compared to the next article reporting the same news event.

An excerpt from the top article for Feb '97 is shown in Figure 4.7. The visualization shows which parts of the article most significantly contributed to the article's cross-entropy reduction score using the subsequence scoring approach described in Section 4.3.5 (insignificant parts of the article are not included in the excerpt). For each letter, the increase above a minimal font size is proportional to the score of the corresponding position in the article. A word-based visualization is also shown, in which word tokens are scored by the sum of the scores of the comprising letters. We again

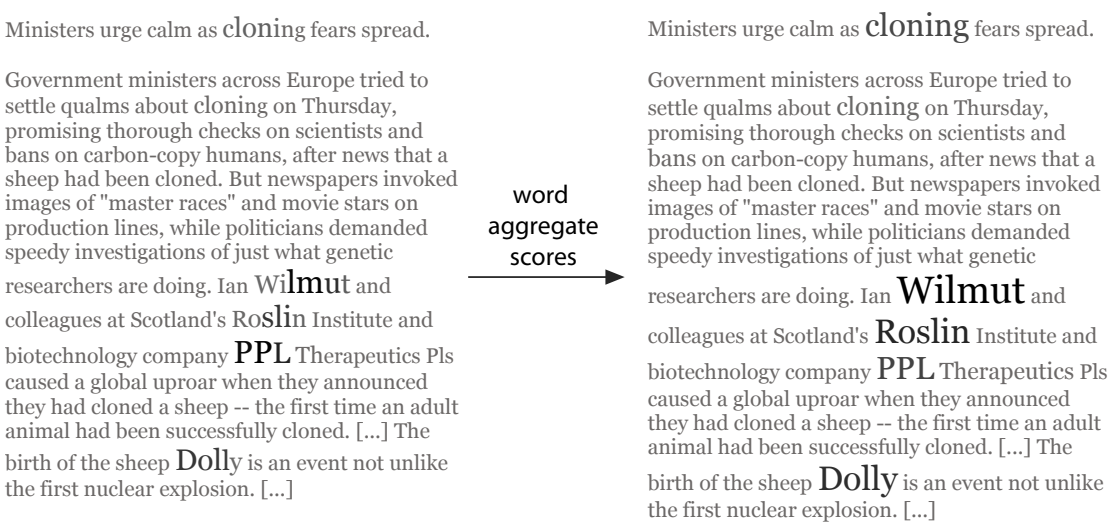


Figure 4.7: Contribution of individual symbols (left) and aggregate contribution of whole words (right) towards the total score of the top-scoring article for Feb '97.

find that the article's significance is largely due to a small number of keywords that are strongly related to the story – the name of the principal researcher in the cloning project "`Ian Wilmut`", the research institute "`Roslin`" and the company backing the project "`PPL Therapeutics`", as well as the word "`cloning`" itself. The visualization also shows that the score of subsequent occurrences of a keyword decreases rapidly, as evidenced by occurrences of the word "`cloning`".

4.8 Discussion

To our knowledge, our model-based approach for finding exemplar data points has few parallels in the current literature. We evaluate the potential applicability of this approach for various text mining tasks. The results of our cluster initialization and active learning experiments are competitive, and sometimes better, than those achieved by specialized methods.

In active learning experiments, we find that good results can be achieved using a simple modification of CERS which favors sampling of uncertain data points, that is, examples that lie relatively close to the classification boundary. Although apparently effective, the method by which this is achieved is extremely simple. Further research would be needed to determine whether a more refined approach could further improve active learning performance.

A natural direction for future work is the evaluation of CERS for other types of sequential data, for example, biological sequences or discrete time series data. Although

the developed method for cross entropy reduction sampling is specifically designed for sequential data, the basic principle of cross-entropy reduction sampling is more general. It remains to be seen whether the idea can be generalized to other types of data in a way that would be both efficient and effective.

Chapter 5

Spam Filtering Using Compression Models

In this chapter, we describe the use of adaptive statistical data compression methods for spam filtering. We argue that compression methods have several characteristics which make them particularly well suited for this task. By modeling email messages as sequences, compression-based filters omit tokenization and other error-prone preprocessing steps, making them very robust to text obfuscation which is widely used by spammers against tokenization-based filters. Furthermore, modeling of semi-structured information found in email headers, as well as characteristic sequences of punctuation and other character-level patterns, which are generally thought to be useful in spam filtering, are naturally included in the model. The models are also fast to construct and incrementally updatable for efficient filtering with online user feedback.

Compression-based filters essentially work by building two models from messages in the training set, one from examples of spam and one from legitimate email. To classify a message, the compression rates achieved using each of these two models are compared. The model that leads to better compression determines the classification outcome. We demonstrate that compression methods match or exceed the performance of tokenization-based text classifiers in online spam filtering experiments using standard datasets and evaluation methodologies. Compression models also compare favorably to prior methods considered in previous studies, when evaluated using the same datasets and equivalent cross-validation experiments. We consider two variants of compression-based filtering, differing in whether or not model adaptation is used when the compression models are applied for classification. The adaptive method, which we propose, has a natural interpretation in terms of the minimum description length principle and generally improves filtering performance in the spam filtering domain. We also demonstrate

that compression models are robust to the type of noise introduced by text obfuscation, which is commonly used by spammers against tokenization-based filters.

We first proposed the use of adaptive statistical data compression models for spam filtering within the framework of the spam filter evaluation conducted at the 2005 Text REtrieval Conference – TREC (Bratko and Filipič, 2005), in which compression-based filters showed very positive results on a number of real-world email collections (Cormack and Lynam, 2005b). Compression methods were also tested in spam filter evaluations at the TREC 2006 and TREC 2007 workshops (Cormack, 2006, 2007b). A high-level summary of the results of the TREC workshops is given in Section 5.6.1.

Part of the material presented in this chapter is the result of joint work with Gordon Cormack, published in (Bratko et al., 2006a). Specifically, Cormack originally compiled the results in Section 5.6.3, in which compression methods are evaluated using cross-validation experiments. Note that Cormack is the original author of the dynamic Markov compression algorithm (DMC; Cormack and Horspool, 1987), and is also the author of the DMC-based spam filter used to obtain the DMC results reported in the filtering method comparison in Section 5.6.2.

Data compressors. We report the results of online filtering experiments using a filter based on the prediction by partial matching (PPM) compression algorithm (Cleary and Witten, 1984). A similar PPM-based filter was the best performing spam filter at the TREC 2005 workshop (Cormack and Lynam, 2005b), and was included in later official evaluations at TREC 2006 (Cormack, 2006) and TREC 2007 (Cormack, 2007b). A filter based on the context tree weighting (CTW) compression algorithm (Willems et al., 1995) was also evaluated at TREC 2005, with performance that was comparable, although slightly inferior, to PPM (for details, see Bratko and Filipič, 2005). Since the CTW algorithm is also more complex and computationally less efficient, it was not tested in later evaluations. When comparing the performance of compression filters and other methods, we also report results obtained using the dynamic Markov compression (DMC) algorithm (Cormack and Horspool, 1987).

5.1 Problem Description

Electronic mail is arguably the "*killer app*" of the internet. It is used daily by millions of people to communicate around the globe and is a mission-critical application for many businesses. The SMTP protocol was among the pioneering internet standards, developed many years before email became ubiquitous. With no perceivable threats of abuse, the original protocol specification contains few considerations regarding security

and authentication of senders. This is perhaps the root cause for the rise of email spam, that is, "*unsolicited, unwanted email sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient*" (Cormack and Lynam, 2005a).

5.1.1 Scope and Scale of the Spam problem and the Solutions

An overwhelming amount of spam is flowing into users' mailboxes daily. In 2010 an estimated 89% of all email was attributed to spam, according to Symantec MessageLabs, a leading anti-spam vendor (Wood et al., 2011). This translates to hundreds of billions of unwanted messages per day. Botnets of several million infected and remotely controlled computers generally account for 80-90% of all spam sent globally. Although the vast majority of spam is now stopped before it ever reaches a user's mailbox (Goodman et al., 2007), trends in spam volumes show little sign of abating (see Figure 5.1). Not only is spam frustrating for email users, it strains the IT infrastructure of organizations and costs businesses billions of dollars in IT costs and lost productivity. Unsolicited email has also evolved from an annoyance into a serious security threat, and has established itself as a prime medium for phishing of sensitive information, as well the spread of malicious software.

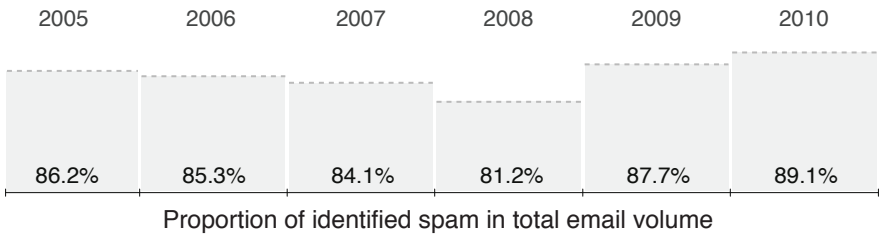


Figure 5.1: Combined annual spam rates from 2005 to 2010, as reported by the anti-spam solution provider Symantec MessageLabs (Wood et al., 2011).

Whereas anti-spam legislation and legal action have had limited success, diverse technical measures are commonly used to combat spam. These include various sender authentication protocols, global blacklists and whitelists of mail servers and domains, tests of compliance with email and networking protocols, spam traps, challenge response tests, and even taxing senders in computational costs (cf. Goodman et al., 2005, 2007; Lynam, 2009). The use of content-based filters, capable of discerning spam from legitimate email messages automatically, is also a popular and successful approach. Machine learning methods are particularly well suited for this task, since they are capable of adapting to the evolving characteristics of spam, and data is often available for training such models. Many learning-based methods have been proposed for spam filtering and several reviews of these methods are available (Goodman et al., 2007; Cormack and

Lynam, 2007; Cormack, 2008; Blanzieri and Bryl, 2008; Guzella and Caminhas, 2009).

5.1.2 The Adversarial Nature of Spam Filtering

Perhaps the most distinguishing characteristic of spam filtering compared to other text categorization tasks is that spam filters face an active adversary which constantly attempts to evade filtering. This has often been described as an "arms race" between spammers and engineers working against spam, resulting in continuous evolution of techniques on both sides.

One tactic commonly employed by spammers is the use of various non-standard character sets and message encodings to hide or mask text from content filters, while maintaining legibility for humans. Other tactics include mutating messages to avoid fingerprinting, or adding seemingly random text such as book passages or fragments of recent news stories in an attempt to confuse statistical filters. It is also common for spammers to hide their messages in images and other types of binary attachments.

Content filters traditionally extract word-like tokens from messages and use these tokens as attributes for training statistical models. One tactic used by spammers against tokenization-based filters is to break up words by inserting HTML markup or by using other formatting and layout tricks which do not change the rendered presentation. Another tactic which is particularly notorious is the intentional use of common misspellings and the use of "leetspeak"[1]. This situation has motivated some interesting research, leading to the conclusion that there are over 6×10^{20} ways to spell "`v1@gra`" that are legible to the human eye (Hayes, 2007), and that hidden Markov models can be used to reverse such intentional misspelling (Lee and Ng, 2005; Lee et al., 2007). Although tokenization-based spam filters may be programmed to anticipate and even correct all plausible and implausible character substitutions and spelling variations, spammers have a reputation of inventing new, creative ways of getting their message across.

5.1.3 Other Considerations for Learning-based Spam Filtering

Users often dismiss the use of spam filters for fear of losing important legitimate email due to filtering errors. Indeed, the relative costs of misclassification are heavily skewed in spam filtering. Although the exact tradeoff will vary for each individual user, labeling a legitimate email as spam, typically referred to as a *false positive*, usually carries a much greater penalty than vice-versa. This must be taken into account when designing and evaluating filters.

[1] Leet is an alternative alphabet of letters originating from internet jargon, for example, writing `\/1@gra` instead of `Viagra`.

Email messages are semi-structured documents consisting of several standard and optional document fields, and it is widely accepted that non-textual information in message headers is an important resource for content-based filters. Certain learning-based methods put greater emphasis on the headers of emails (Assis, 2006), and some even operate using header information exclusively (e.g. Segal, 2005).

Since spam evolves continuously and many practical applications are based on online, interactive user feedback, the task calls for efficient algorithms which should also support incremental updates.

5.2 Related Work

When applied to the spam filtering problem, we argue that a major advantage of compression models is that they implicitly operate on variable-length character-level or bit-level substrings, rather than tokenization-based feature vectors. In the following, we review other spam filtering methods which also rely on character-level features. First, however, we discuss closely related work on the use of data compression methods for various text classification tasks other than spam filtering.

Text classification using data compression methods

The use of data compression models, and closely related character-level language models, has been considered in various text classification settings. The applications include classification by topic (Frank et al., 2000; Teahan and Harper, 2003; Peng et al., 2004; Marton et al., 2005), language and dialect identification (Teahan, 2000), classification by genre (Teahan, 2000; Peng et al., 2004), authorship attribution (Teahan, 2000; Benedetto et al., 2002; Peng et al., 2004; Marton et al., 2005), plagiarism detection in computer program source code (Chen et al., 2004), computer user identification from Unix shell session logs (Sculley and Brodley, 2006), detection of malicious programs (Zhou and Inge, 2008), automated vandalism detection in Wikipedia (Smets et al., 2008; Itakura and Clarke, 2009), and classification of short messages or "tweets" (Nishida et al., 2011). A review of this body of literature is also given in Section 2.5.

Although many of these studies predate our work, to our knowledge, no prior studies exist on the use of data compression methods for spam filtering. Spam filtering is distinguished from other text classification tasks in many important aspects, particularly by the adversarial relationship between spammers and email filters, unbalanced misclassification costs, the importance of modeling semi-structured header information in addition to textual data, and the need for online incremental learning. Compression

models excel in dealing with these aspects compared to many standard text classification methods, which makes them particularly appealing for this task.

We compare two different compression-based classification criteria, one of which relies on adaptation of compression models, an approach which has previously not been considered. This approach has an intuitive interpretation in terms of the minimum description length principle and consistently improves filtering performance in our experiments.

Character-level methods in spam filtering

Standard machine learning algorithms accept data in the form of fixed-length feature vectors. When using these methods, text documents are most often modeled with the bag-of-words (BOW) representation. This representation has traditionally also been used in spam filtering, even though it is widely accepted that tokenization is a vulnerability of keyword-based spam filters (Wittel and Wu, 2004).

Instead of word tokens, some filters extract character n-grams, that is, all consecutive sequences of n characters found in the message, and apply machine learning algorithms on the resulting feature vector (e.g. Goodman et al., 2005). When we first proposed the use of data compression models for spam filtering (Bratko and Filipič, 2005), this approach was not very common, but has since gained in popularity, with several researchers reporting that feature vectors based on character n-grams outperform word-based features (Sculley and Wachman, 2007a; Cormack, 2007a; Sculley, 2008). We note that some authors refer to the success of data compression methods as motivation for adopting a character-level representation (cf. Sculley et al., 2006; Xu et al., 2007). We discuss this development in more detail in Section 5.6.1.

Some studies go beyond character n-gram features, and consider feature spaces that accommodate approximate or partial matches of character substrings. This is achieved either by extending the feature space explicitly, for example, by adding n-grams with "gaps" and "wildcards" (Sculley et al., 2006), or via the kernel trick in combination with classifiers capable of operating with kernels (Amayri and Bouguila, 2010). Interestingly, both Sculley et al. (2006) and Amayri and Bouguila (2010) obtain best results with the baseline representation which uses contiguous character n-grams as features.

Two prior studies that we are aware of consider algorithms that operate natively on variable-length character sequences, similarly to the compression-based methods that we consider. IBM's Chung-Kwei system (Rigoutsos and Huynh, 2004) uses pattern matching techniques originally developed for DNA sequences. Messages are filtered based on the number of occurrences of patterns associated with spam and the extent to which they cover the target document. Pampapathi et al. (2006) propose a filtering technique

based on the suffix tree data structure. They investigate a number of ad hoc scoring functions on matches found in the target document against suffix trees constructed from spam and legitimate email. The techniques proposed in these studies are different from methods based on statistical data compression models which we consider in this thesis. In particular, while these systems use character-based features in combination with some ad hoc scoring functions, compression models were designed for the specific purpose of *probabilistic* modeling of sequential data. This property of data compression models allows them to be employed in an intuitively appealing and principled way.

5.3 Methods

In essence, compression models can be applied to text categorization by building one compression model from the training documents of each class and using each of these models to evaluate the probability of the target document. The class label is then chosen according to the compression model which assigns the greatest probability to the target document. We describe two approaches to classification, in which data compression models are used to measure two different information-based quantities.

We first describe the minimum cross-entropy (MCE) approach (Frank et al., 2000; Teahan, 2000). This approach aims to measure the cross-entropy between the probability distributions embodied by the compression models induced from the training data and the unknown probability distribution from which the document we wish to classify was sampled. The class for which the cross entropy is minimized is selected. The second method measures the increase of the compressed size of the training dataset resulting from adding the classified document to the training data. It selects the class for which this description length increase is minimal, which is why we consider this a minimum description length (MDL) approach. The difference in practice is that in the MDL approach, the compression models are *adapted* (i.e. updated after each symbol) while evaluating the probability of the target document, as would normally be the case when using these models for data compression. In the MCE approach the models induced from the training data are kept fixed during classification, as would be customary in the application of probabilistic models for classification in the machine learning literature. In subsequent sections, we also refer to the MDL approach as using *adaptive* models and the MCE approach as using *static* models.

We denote by C the set of classes and by $y : \Sigma^* \to C$ the (partially specified) function mapping documents to class labels. Given a set of training documents D for which class labels are known, the task is to assign a target document $\mathbf{d}$ with an unknown label to one of the classes $c \in C$. All logarithms are assumed to have a base of 2, i.e.

$\log(x) \equiv \log_2(x)$.

5.3.1 Classification by Minimum Cross-entropy

The *cross-entropy* $H(X, \theta)$ determines the average number of bits per symbol required to encode messages produced by a source X when using a model θ for compression:

$$H(X, \theta) = E_{\mathbf{x} \sim P} \left[\frac{1}{\overline{\mathbf{x}}} L(\mathbf{x}|\theta) \right] \quad . \tag{5.1}$$

Here, P denotes the probability distribution associated with the source variable X. $L(\mathbf{x}|\theta)$ denotes the ideal code length for $\mathbf{x}$ under the model θ:

$$L(\mathbf{x}|\theta) = -\log p(\mathbf{x}|\theta) \quad . \tag{5.2}$$

Note that $H(X, \theta) \geq H(X)$ always holds, that is, the best possible model achieves a compression rate equal to the entropy of the source.

The exact cross-entropy is hard to compute, since it would require knowing the source distribution P. It can, however, be approximated by applying the model θ to sufficiently long sequences of symbols, with the expectation that these sequences are representative samples of all possible sequences generated by the source (Brown et al., 1992; Teahan, 2000):

$$H(X, \theta) \approx \frac{1}{\overline{\mathbf{x}}} L(\mathbf{x}|\theta) \quad . \tag{5.3}$$

As $\overline{\mathbf{x}}$ becomes large, this estimate will approach the actual cross-entropy in the limit almost surely if the source is ergodic (Algoet and Cover, 1988). Recall that if θ is a Markov model with limited memory k, then

$$L(\mathbf{x}|\theta) = -\log \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_{i-k}^{i-1}, \theta) \quad , \tag{5.4}$$

where $p(x_i|\mathbf{x}_{i-k}^{i-1}, \theta)$ is the probability of x_i given $\mathbf{x}_{i-k}^{i-1}$ according to θ.

Following Teahan (2000), we refer to the cross-entropy estimated on the target document $\mathbf{d}$ as the *document cross-entropy* $H(X, \theta, \mathbf{d})$. This is simply a substitution of $\mathbf{x}$ with $\mathbf{d}$ in the right hand of Equation 5.3. We expect that a model that achieves a low cross-entropy on the target document approximates the information source that actually generated the document well. This is therefore our measure for classification:

$$y(\mathbf{d}) \quad = \quad \arg\min_{c \in C} \; H(X, \theta_c, \mathbf{d})$$

$$= \arg\min_{c \in C} -\frac{1}{\overline{\mathbf{d}}} \log \prod_{i=1}^{\overline{\mathbf{d}}} p(d_i | \mathbf{d}_{i-k}^{i-1}, \theta_c) \ , \tag{5.5}$$

where θ_c denotes the compression model built from all examples of class c in the training data.

5.3.2 Classification by Minimum Description Length

The MCE criterion assumes that the test document $\mathbf{d}$ was generated by some unknown information source. The document is considered a sample of the type of data generated by the unknown source. Classification is based on the distance between each class and the source that generated the document. This distance is measured with the document cross-entropy, which serves as an estimate of the cross-entropy between the unknown source and each of the class information sources.

However, we know that the document did not originate from some *unknown* source and that it ultimately must be attributed to one of the classes. The MDL classification criterion tests, for each class $c \in C$, the hypothesis that $y(\mathbf{d}) = c$, by adding the document to the training data of the class and estimating how much this addition increases the description length of the dataset. This is in line with the approach suggested by Kontkanen et al. (2005) in their MDL framework for clustering, in which the cluster assignment should be such that it results in a minimal description length of the data, measured by a suitable universal model. In our case, the description length is measured using adaptive universal compression models, as stipulated by the prequential form of the MDL principle (Grünwald, 2005).

Let D_c denote the set of labeled examples for class c:

$$\mathrm{D}_c = \{\mathbf{x} \,; \mathbf{x} \in \mathrm{D}, y(\mathbf{x}) = c)\} \ . \tag{5.6}$$

The MDL classification criterion is then given by the minimization of

$$\Delta L(\mathrm{D}, c, \mathbf{d}) = L(\mathrm{D}_c \cup \{\mathbf{d}\}) - L(\mathrm{D}_c) \ , \tag{5.7}$$

where $L(*)$ is the description length, or compressed length, of a set of strings. We are searching for the classification hypothesis that yields the most compact description of the observed data, i.e. minimizing the description length increase $\Delta L(\mathrm{D}, c, \mathbf{d})$.

We measure the description length increase $\Delta L(\mathrm{D}, c, \mathbf{d})$ using adaptive compression algorithms which allow efficient estimation of this quantity. Adaptive models can be used to estimate the increase in description length without re-evaluating the entire

dataset:

$$\Delta L(\mathrm{D}, c, \mathbf{d}) = -\log \prod_{i=1}^{\overline{\mathbf{d}}} p(d_i | \mathbf{d}_{i-k}^{i-1}, \theta_c(\mathbf{d}_1^{i-1})) \ . \tag{5.8}$$

In this equation, $\theta_c(\mathbf{d}_1^{i-1})$ denotes the current model at position i, constructed from the input sequence $\mathbf{d}_1^{i-1}$ in addition to all of the training data for class c.

Typically, the description length increase $\Delta L(\mathrm{D}, c, \mathbf{d})$ will be larger for longer documents. We therefore use the per-symbol description length increase in the final class selection rule:

$$\begin{aligned} y(\mathbf{d}) &= \arg\min_{c \in C} \ \frac{1}{\overline{\mathbf{d}}} \Delta L(\mathrm{D}, c, \mathbf{d}) \\ &= \arg\min_{c \in C} \ -\frac{1}{\overline{\mathbf{d}}} \log \prod_{i=1}^{\overline{\mathbf{d}}} p(d_i | \mathbf{d}_{i-k}^{i-1}, \theta_c(\mathbf{d}_1^{i-1})) \ . \end{aligned} \tag{5.9}$$

The additional $1/\overline{\mathbf{d}}$ factor does not affect the classification outcome for any target document, but it does help to produce scores that are comparable across documents of different length. This is crucial when thresholding is used to reach a desirable tradeoff in misclassification rates. Note that the only difference in implementation in comparison to the MCE criterion in Equation (5.5) is that the model is adapted while evaluating the target. It is clear, however, that Equation (5.9) no longer amounts to measuring the document cross-entropy $H(X, \theta_c, \mathbf{d})$ with respect to model θ_c, since a different model is used at each position of the sequence $\mathbf{d}$.

Intuitively, the description length increase $\Delta L(\mathrm{D}, c, \mathbf{d})$ measures the surprise at observing $\mathbf{d}$ under the hypothesis $y(\mathbf{d}) = c$, which is proportional to the (im)probability of $\mathbf{d}$ under the model θ_c for c. The intuition behind adapting the model θ_c is that it *conditionally* continues to learn about c from the target document: If the hypothesis $y(\mathbf{d}) = c$ actually holds and an improbable pattern is found in the initial part of $\mathbf{d}$, then the probability that the pattern reoccurs in the remainder of the document should increase. Although we do not know whether the hypothesis $y(\mathbf{d}) = c$ is true or not, it is assumed to be true for the purpose of testing its tenability.

Let us consider an illustrative example as to why the MDL classification criterion might be preferable to the MCE approach. Consider a hypothetical spam filtering problem in which an imaginary researcher uses a compression-based classifier to filter spam from his email. In addition to research-related email, our researcher also receives an abundant amount of spam that advertises pharmaceuticals. At some point, he receives an email on machine learning methods for drug discovery. This is a legitimate email, but it contains many occurrences of the term "drugs" which the filter strongly associates

with spam. In this scenario, the prevalence of this term might cause the MCE criterion to label the email as spam, while the MDL criterion might consider the email legitimate. This is because while the first occurrence of the term "drugs" is surprising under the hypothesis that the document is legitimate, subsequent occurrences are more and more probable as the model is adapted. The repeated occurrences are, in a sense, redundant, and the MDL criterion favors heterogeneous evidence compared to the MCE method. We present a real-world example of such behavior in the results section.

It is interesting to note that Benedetto et al. (2002), who consider the use of the LZ77 compression algorithm (`zip`) for language identification and authorship attribution, notice that LZ77 adapts to the target text and take measures to prevent this behavior. Marton et al. (2005) also discuss the fact that compression models adapt to the classified document, and characterize this as an unavoidable "approximation", due to the use of off-the-shelf compression programs. We, on the other hand, believe this effect is beneficial, which is supported by the results of our experiments.

5.3.3 Comparison to Bayes Rule

According to Bayes rule, the most probable class, which maximizes the class probability $p(c|\mathbf{d})$ given the target document $\mathbf{d}$, is given by

$$\begin{aligned} y(\mathbf{d}) &= \arg\max_{c \in C} \; p(c)\, p(\mathbf{d}|c) \\ &= \arg\min_{c \in C} \; -\log p(c) \; -\log p(\mathbf{d}|c) \end{aligned} \tag{5.10}$$

We may use a data compression model θ_c, constructed from training documents belonging to class c, to assess the class-conditional probability $p(\mathbf{d}|c)$ of the document $\mathbf{d}$. In this case, the compression-based classification criteria in Equations (5.5) and (5.9) differ from Bayes rule in two minor points:

- the class prior $p(c)$ is omitted, which effectively implies equal prior probabilities of spam and legitimate email;
- an additional document-length normalization factor of $1/\overline{\mathbf{d}}$ is used in the compression-based filtering criteria.

Additionally, the MCE and MDL classification criteria differ from each other in whether or not the compression models are applied adaptively to estimate $p(\mathbf{d}|c)$.

When classifying according to Equation (5.10), the class prior $p(c)$ is generally overwhelmed by the conditional probability $p(\mathbf{d}|c)$, and thus using equal priors has little effect on the classification outcome. This can be seen if we consider that $-\log p(c)$ is

the code length required to encode the class label, which will on average require 1 bit or less, since there are only two possible values. The code length $-\log p(\mathbf{d}|c)$, required to encode the entire document $\mathbf{d}$, will typically be much greater, and thus also has much greater impact on the classification outcome.

The normalization by $1/\overline{\mathbf{d}}$ in the compression-based criteria does not affect the classification outcome for a particular target document. Instead, it helps to make scores comparable among target documents with different lengths. The length normalization is equivalent to using $\sqrt[\overline{\mathbf{d}}]{p(\mathbf{d}|c)}$ instead of $p(\mathbf{d}|c)$ in the formulation of Bayes rule in Equation (5.10). The utility and motivation for such normalization is perhaps more apparent from the information-based perspective offered by the MCE and MDL classification criteria. The intent is to measure *compression rate* instead of compressed size, as compression rates are more naturally compared across different-length documents.

5.4 Experimental setup

Several aspects of spam filtering critically influence the design of experiments and the evaluation of filtering techniques:

- The cost of misclassification is highly unbalanced. Although the exact tradeoff will vary in different deployment environments, it tends to be biased toward minimizing false positives (i.e. misclassified legitimate messages).

- Messages in an email stream arrive in chronological order and must be classified upon delivery. It is also common to deploy a filter without *any* training data. Although many studies use cross-validation experiments, the appropriateness of cross-validation is questionable in this setting.

- Many useful features may be gleaned from various message headers, from message formatting, punctuation patterns and structural features. It is therefore desirable to use raw, unobfuscated messages with accompanying meta data intact for evaluation.

These aspects distinguish spam filtering from classical text categorization tasks, and are reflected in the design of experiments and the choice of appropriate measures for spam filter evaluation. This section provides an overview of spam filter evaluation methodology, the test corpora used, as well as some relevant implementation details of our PPM compression-based spam filter.

5.4.1 Online Spam Filter Evaluation Protocol

A typical spam filter deployment can be simulated by an online learning scheme (Cormack and Lynam, 2007). In this setup, messages are presented to the classifier in chronological order. For each message, the classifier must produce a score as to how likely it is that the message is spam, after which it is communicated the gold standard judgment. This allows the classifier to update its model before assessing the next message. The setup aims to simulate a typical setting in personal email filtering, which is usually based on online user feedback. It is additionally assumed that the user promptly corrects the classifier after every misclassification, and that the feedback is consistent (without labeling errors).

Most of the results reported in this chapter are obtained using the described scheme. Cross-validation experiments on predefined splits were performed to compare compression models to prior methods which were also evaluated with cross-validation in previous studies.

5.4.2 Filtering Performance Measures

Special care must be taken in the choice of evaluation measures for spam filtering. Classification accuracy, that is, the total proportion of correctly classified messages, is a poor performance measure in this application domain, since it does not distinguish between different types of classification errors (Androutsopoulos et al., 2000).

In the binary spam filtering problem, spam messages are usually associated with the positive class, since these are the messages filtered by the system. It is assumed that the scores produced by a learning system are comparable across messages, so that a fixed filtering threshold can be used to balance between the amount of spam that escapes filtering and the risk of losing legitimate email (false positives). Such scores lend themselves well to Receiver Operating Characteristic (ROC) curve analysis. The ROC curve is a plot of the proportion of successfully filtered spam messages on the Y axis, as a function of the false positive rate on the X axis (the proportion of misclassified legitimate email).[2] Each point on the curve corresponds to an actual tradeoff between spam misclassification and false positive rate, achieved by the classifier at a certain threshold value. The curve thus captures the behavior of the system at all possible filtering thresholds.

A good performance is characterized by a curve that reaches well into the upper left quadrant of the graph. The area under the ROC curve (AUC) is then a meaningful

[2]It is traditional to name the axes of an ROC plot $1 - \text{specificity}$ (X axis) and sensitivity (Y axis). Sensitivity is the proportion of correctly identified positive examples and specificity is the proportion of correctly identified negative examples.

statistic for comparing filters. If we assume that high score values are associated with the positive class, the area under the curve equals the probability that a random positive example receives a higher score than a random negative example:

$$\text{AUC} = P(score(\mathbf{x}) > score(\mathbf{y}) \mid class(\mathbf{x}) = 1,\ class(\mathbf{y}) = 0) \ .$$

Typical spam filters achieve very high values in the AUC statistic. For this reason, we report on the complement of the AUC value, that is, the area *above* the curve (1–AUC). Better performance is characterized by a lower 1–AUC result. Bootstrap resampling is used to compute confidence intervals for AUC values and to test for significance in paired comparisons. A logit transform is first applied, which maps the range of AUC values from $[0, 1]$ to $(-\infty, \infty)$, and reduces the skewness of the sample distribution to permit a normal approximation in the analysis (cf. Cormack and Lynam, 2006; Park, 2011).

5.4.3 Datasets

We report results of online filtering experiments using four publicly available datasets. The datasets contain raw, unobfuscated messages, including all header information and message attachments. Three other datasets, namely Ling-Spam, PU1 and PU3, were used for cross-validation experiments for comparisons with prior studies using the same data. We used the predefined 10-fold cross-validation splits which are included in the datasets. Note that the Ling-Spam, PU1 and PU3 datasets do not contain message headers, so the original chronological order which would be required for online spam filter evaluation could not be recovered. Some basic corpora statistics are given in Table 5.1.

Dataset	Messages	Spam	Good	Spam rate	Test protocol
`trec05p`	92189	52790	39399	57.3%	online
`trec06p`	37822	24912	12910	65.9%	online
`trec07p`	75419	50199	25220	66.6%	online
`spamassassin`	6033	1884	4149	31.2%	online
`Ling-Spam`	2893	481	2412	16.6%	10-fold c.v.
`PU1`	1090	480	610	44.0%	10-fold c.v.
`PU3`	4130	1820	2310	44.1%	10-fold c.v.

Table 5.1: Basic statistics for the test datasets.

The TREC 2005 public[3] corpus, labeled `trec05p`, contains messages received by employees of the Enron corporation over a one year period. The original Enron data

[3] `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`

was carefully augmented with the addition of approximately 50,000 spam messages, so that they appear to be delivered to the Enron employees in the same time period as the legitimate email.

The TREC 2006 public[4] corpus, labeled `trec06p`, consists of a collection of spam and legitimate email crawled from the web, and augmented with additional spam messages obtained from spam traps in May 2006. Messages from spam traps were manipulated so that they appear to be delivered to the same users during the same time period as the legitimate email. Names and email addresses in spam were also replaced so as to make them consistent with addressees of the legitimate email.

The TREC 2007 public[5] corpus, labeled `trec07p`, contains all messages delivered to a single server over a three month period between April and July 2007. Some mailboxes on the target server were configured to receive non-personal legitimate email from a number of different services, whereas other mailboxes served as spam traps. The messages were unaltered except for certain systematic substitutions of addresses and names.

The SpamAssassin[6] dataset contains legitimate and spam email from the SpamAssassin developer mailing list. This dataset is smaller and pre-dates the TREC corpora. The SpamAssassin dataset is arguably the most widely used resource in popular evaluations of freely-available spam filters, often carried out by enthusiasts or spam filter creators.

Ling-Spam[7] is a collection of email messages posted to a linguistics newsgroup, which were augmented with spam received by the authors of the dataset. The messages are stripped of all attachments and headers, except for the subject field. A fair number of research studies report results on the Ling-Spam corpus. We used the "bare" version of this dataset in our evaluation.

The PU1 and PU3[8] datasets are relatively small personal email collections. In order to preserve privacy, the words in the messages are replaced with numerical identifiers and punctuation is discarded. Non-textual message headers, sender and recipient fields, attachments and HTML tags are not included in these datasets. Duplicate spam messages received on the same day are also removed.

Three of the described corpora were created to facilitate the large-scale spam filter evaluations conducted at TREC from 2005 to 2007. PPM compression-based spam filters were included in each of the TREC evaluations, and we report a high-level summary

[4] `http://plg.uwaterloo.ca/~gvcormac/treccorpus06/`

[5] `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/`

[6] The SpamAssassin dataset is available at `http://spamassassin.org/publiccorpus/`

[7] The Ling-Spam dataset is available at `http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz`

[8] The PU1 and PU3 datasets are available for download at `http://www.iit.demokritos.gr/skel/i-config/downloads/`

of these results in Section 5.6.1. Several private email datasets were also used in the TREC trials. Detailed descriptions of all datasets used at TREC can be found in the TREC proceedings (Cormack and Lynam, 2005b; Cormack, 2006, 2007b).

5.4.4 Filter Implementation Details

Our PPM-based spam filter uses our own custom implementation of the PPM compression algorithm. We use PPM escape method D (Howard, 1993) for blending probability estimates, since this method is known to perform well for data compression (cf. Teahan, 1995). Whenever the PPM escape mechanism is triggered, which causes PPM to back-off to a lower-order context, symbols with non-zero counts in higher-order contexts are excluded from the lower-order statistics. This technique is known as exclusion (see Sayood, 2005, section 6.3.4). Our implementation did not, however, use update exclusion – the practice of selectively updating next-symbol statistics only for contexts which "need to be visited" when computing the probability of the current symbol (see Moffat, 1990). This design ensures that the PPM model is insensitive to the order in which training documents are presented, and enables us to efficiently revert incremental model updates, or decrementally remove a particular document from the training data. Unless noted otherwise, we use order-6 PPM models over the 256-valued alphabet of byte codes. The effect of varying the PPM model order is further discussed and evaluated in Section 5.5.2.

Optimizing memory use. Our implementation of the PPM algorithm was carefully optimized to minimize the amount of memory required to store the PPM model. To this end, statistics are maintained in a custom suffix tree data structure (Weiner, 1973), specially engineered to have a small memory footprint, as detailed in (Bratko et al., 2006b). Consequently, the classifier can handle large training sets in excess of 100,000 messages and several 100 megabytes of training data, despite the implicitly rich feature space. For example, memory usage of the filter peaks at 410 MB after training an order-6 PPM model on the full TREC 2005 public corpus, which contains 92,189 messages and roughly 324 MB of text after MIME decoding.

Filtering speed. The PPM filter is comparatively fast, with classification and incremental training times both linear in the length of the message. The execution time of the adaptive PPM filter in online filtering experiments is reported in Table 5.2. The execution time was measured on an Intel Core i7 processor with a clock speed of 2.66 GHz. In the experiment, an adaptive order-6 PPM filter typically handles just under 40 messages per second (both classification and training) if messages are truncated at

2500 bytes. In general, the use of higher-order PPM models decreases filtering throughput. The filtering throughput, measured in kilobytes per second, is generally consistent across different datasets, ranging from 70 to 90 kilobytes per second for an adaptive order-6 PPM model.

Filtering speed, full-length messages

	spamassassin		trec05p		trec06p		trec07p	
Order	msg/sec	kB/sec	msg/sec	kB/sec	msg/sec	kB/sec	msg/sec	kB/sec
2	27.3	143.2	32.0	116.3	31.9	109.1	25.2	117.5
3	25.7	134.7	29.7	107.8	29.9	102.0	24.9	116.1
4	22.4	117.7	26.2	95.2	27.9	95.2	22.4	104.3
5	19.8	103.8	23.4	85.1	26.5	90.7	19.6	91.2
6	**17.7**	**92.8**	**20.3**	**73.5**	**24.3**	**82.9**	**17.4**	**81.2**
7	16.2	84.9	19.0	69.0	21.5	73.5	14.8	69.0
8	15.0	78.7	17.4	63.3	20.5	70.2	14.0	65.5

Filtering speed, truncated messages

	spamassassin		trec05p		trec06p		trec07p	
Order	msg/sec	kB/sec	msg/sec	kB/sec	msg/sec	kB/sec	msg/sec	kB/sec
2	62.9	141.2	58.1	113.5	64.8	119.2	49.3	110.2
3	52.5	117.9	55.3	108.0	64.5	118.8	47.2	105.4
4	46.4	104.3	49.0	95.8	51.8	95.3	43.2	96.4
5	43.7	98.2	39.8	77.8	41.7	76.8	37.3	83.2
6	**39.7**	**89.2**	**37.1**	**72.4**	**38.0**	**70.0**	**32.1**	**71.7**
7	37.0	83.2	34.8	68.0	34.6	63.7	29.6	66.2
8	33.9	76.2	30.7	60.0	30.9	56.9	26.8	60.0

Table 5.2: Filtering speed of online spam filtering using the adaptive PPM filter on the `spamassassin`, `trec05p`, `trec06p` and `trec07p` datasets. The measurment includes both classification and incremental training for each message in the email stream.

Message preprocessing. In terms of message pre-processing, our PPM-based classifier discards all non-text message attachments and decodes any message parts that are presented in base64 or quoted-printable encoding. Furthermore, messages are truncated at 2500 characters in most of the reported experiments. We discuss and evaluate the effect of message truncation in Section 5.5.3. No other pre-processing is applied, so that all header meta-data and formatting is preserved, as well as all punctuation and letter capitalization patterns.

5.5 Experimental Results

In this section, we report the results of experiments which study the spam filtering performance and behavior of PPM-based filters, subject to various settings. We compare the relative performance of the MCE and MDL classification criteria, that is, the effect of adapting the compression model at classification time. We also measure the effect of varying the PPM model order parameter, and the effect of truncating messages at different cut-off points. Experiments that measure filtering performance under varying degrees of noise in the data are also reviewed, aiming to demonstrate the effect that typical obfuscation tactics employed by spammers may have on compression and tokenization-based filters. We conclude the section with a visualization of the classification process, using selected examples that reveal interesting patterns detected by the classifier.

5.5.1 Comparison of MCE and MDL Classification Criteria

The effect of adapting the PPM compression model when evaluating document probabilities is reported in Table 5.3, which compares 1–AUC results of static and adaptive versions of the PPM classifier. The adaptive models clearly outperform their static counterparts on all four datasets. The area above the ROC curve is roughly halved when using an adaptive filter in all trials. The difference is found to be significant for three out of four datasets.

Dataset	MCE	MDL
`trec05p`	0.0220 (0.0154 - 0.0314)	**0.0142*** (0.0104 - 0.0193)
`trec06p`	0.0408 (0.0299 - 0.0555)	**0.0282*** (0.0205 - 0.0389)
`trec07p`	0.0147 (0.0053 - 0.0411)	**0.0082** (0.0030 - 0.0224)
`spamassassin`	0.3191 (0.1878 - 0.5416)	**0.1794*** (0.1047 - 0.3075)

Table 5.3: Comparison of filtering performance using order-6 PPM in combination with the MCE and MDL classification criteria on the SpamAssassin and TREC public corpora. Results are in the area above the ROC curve 1–AUC(%) statistic and include 95% confidence intervals for this measure. The best result for each dataset is in bold. Statistically significant differences are marked with a '$\star$' sign ($p < 0.01$, one-tailed t-test).

The ROC curves of adaptive and static PPM models are depicted in Figure 5.2, and reveal an interesting and remarkably consistent pattern. Note that the ROC graphs are plotted in logarithmic scale for clarity, so a non-convex ROC area is not necessarily unexpected. Although adaptive models dominate throughout the curve in most experiments, the gain in the 1–AUC statistic can mostly be attributed to the fact that the adaptive models perform better at the extreme ends of the curves. Performance is comparable when misclassification rates are roughly balanced for spam and legitimate

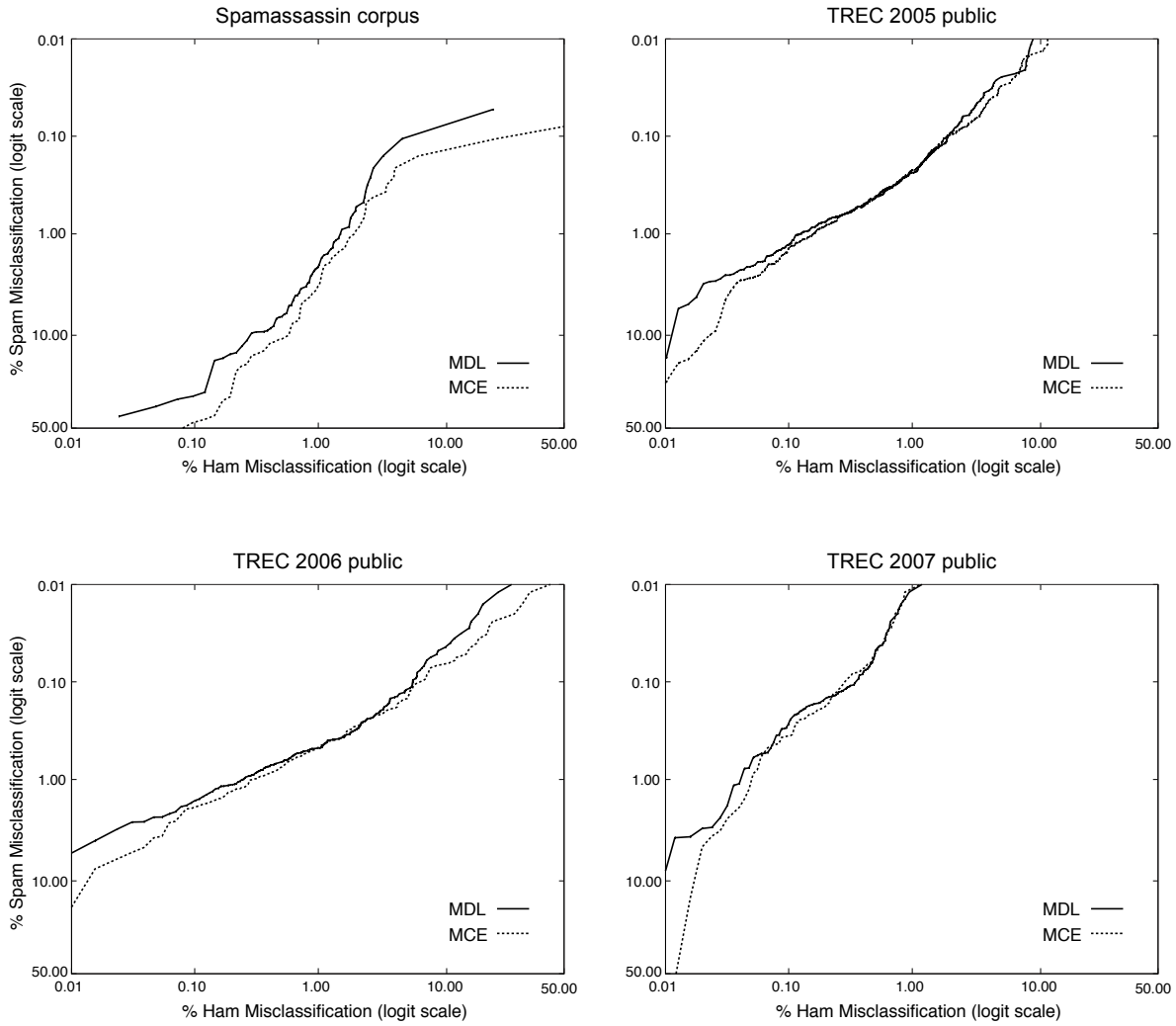


Figure 5.2: ROC curves on the SpamAssassin and TREC public corpora. Performance of MCE and MDL classification criteria are compared using order-6 PPM compression models.

Figure 5.3: A visualization of how each letter in the subject and body of message 038/267 from the `trec05p` corpus contributes towards the "spam" classification when using the MCE (left) and MDL (right) classification criteria. A letter is rendered in gray if it points towards a "ham" classification, and colored progressively blue whenever it contributes towards a "spam" classification. The models are adapted in the MDL variant, which is depicted in the right column.

mail. The logarithmic scale should again be taken into consideration when examining the magnitude of this effect. This suggests that the adaptive model makes less gross mistakes, which are costly in terms of the AUC measure. Performance at the extreme ends of the curve is especially important when the cost of misclassification is unbalanced, so this is certainly a desirable property for spam filtering where minimizing one type of error is typically more important than overall classifier accuracy. The same pattern was observed in experiments using dynamic Markov compression (DMC) instead of PPM (Bratko et al., 2006a).

A visualization of the effect that adaptation of the compression model may have on the classification outcome is shown in Figure 5.3, using a specific real-world example: message 038/267 from the TREC 2005 public corpus. In the visualization each character is colored progressively blue according to how "spammy" it looks to the PPM filter. The example that is shown depicts a legitimate email which is misclassified as spam by the static version of the PPM classifier. Furthermore, this misclassification is done with very large confidence on the classifier's part, with $p(\texttt{spam}) = 0.87$. Upon inspection, we find that this is chronologically the first HTML formatted message found in the TREC 2005 public corpus which is *not* spam. The abundance of HTML tags otherwise common in spam cause the gross misclassification. Remarkably, the adaptive version of PPM classifies this message correctly with $p(\texttt{spam}) = 0.44$, due to the fact that the adaptive method quickly adapts to the HTML tags, while other patterns in the message point towards the opposite classification result.

5.5.2 Varying PPM model order

Spam filtering performance for PPM models with different choices of the PPM order parameter is shown in Figure 5.4. Performance generally improves with increasing order up to a point and then stabilizes or decreases slightly, depending on the dataset. Note the logarithmic scale in the graph which amplifies differences at higher performance levels (lower 1–AUC values). A PPM model of order 5 or greater is generally required to maximize performance, except for the `trec07p` dataset, where performance is optimal when using simpler order-3 models.

PPM models of different order were also tested as part of the TREC 2005 spam filter evaluation (Cormack and Lynam, 2005b). Due to limits on the number of systems each group was entitled to submit for the TREC evaluations, only order-4, order-6 and order-8 PPM models were tested at TREC 2005. It was concluded that the classifier is generally robust to the choice of the model order parameter (see Bratko and Filipič, 2005). Order-6 PPM was the best-performing variant when compared over all four datasets at TREC 2005. Only order-6 PPM was tested in later TREC evaluations.

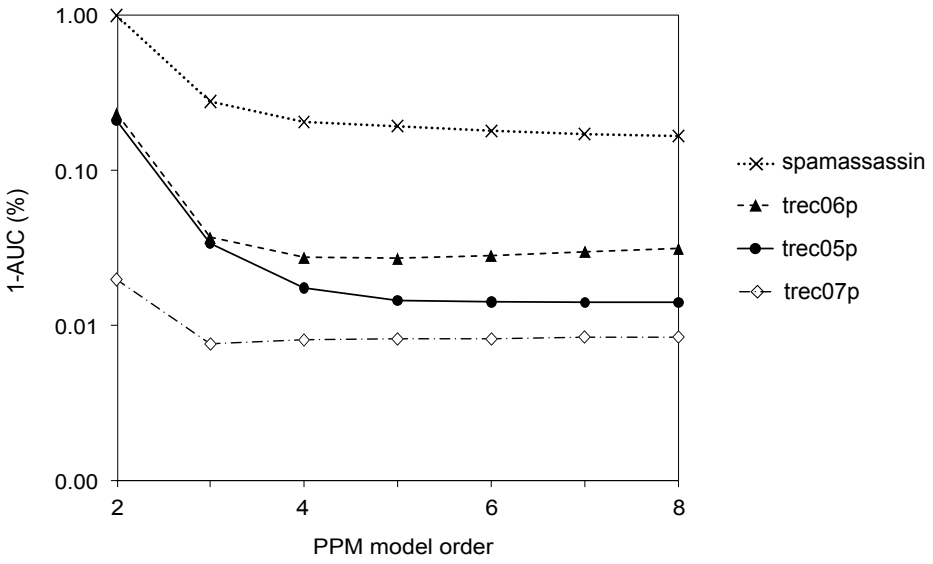


Figure 5.4: Effect of varying the model order parameter of the adaptive PPM filter on the Spamassassin and TREC public datasets. 1–AUC (%) performance is plotted in logarithmic scale.

5.5.3 Effect of Truncating Messages

Email messages, particularly message bodies, often vary in length greatly. In practice, differences in length of two or three orders of magnitude are common. For example, messages in the TREC 2005 public corpus range from roughly 500 bytes to several megabytes in length *after* decoding and removing non-textual attachments. There is much less variability in the length of email headers. For this reason, message truncation, the practice of discarding message content beyond a certain cutoff length, may not only improve filtering speed and memory efficiency, but also filtering performance. Sculley (2008) concludes that this is due to an implicit normalization of message length, which also increases, or rather, regulates the relative importance of message headers.

The effect of message truncation on PPM spam filtering performance is depicted in Figure 5.5. The truncation cutoff is varied between 1000 and 10.000 bytes at 500 byte increments. It should be noted that the results plotted in the figure are obtained by training the PPM model on full messages and then using only message prefixes for classification. We found that this strategy produces comparable results to those obtained when using the same truncation regime for training and classification, while allowing us to collect all measurements in a single pass. At settings that were tested explicitly, results improve marginally if messages are also truncated when training the PPM model.

We find that truncating documents can improve spam filtering performance, which

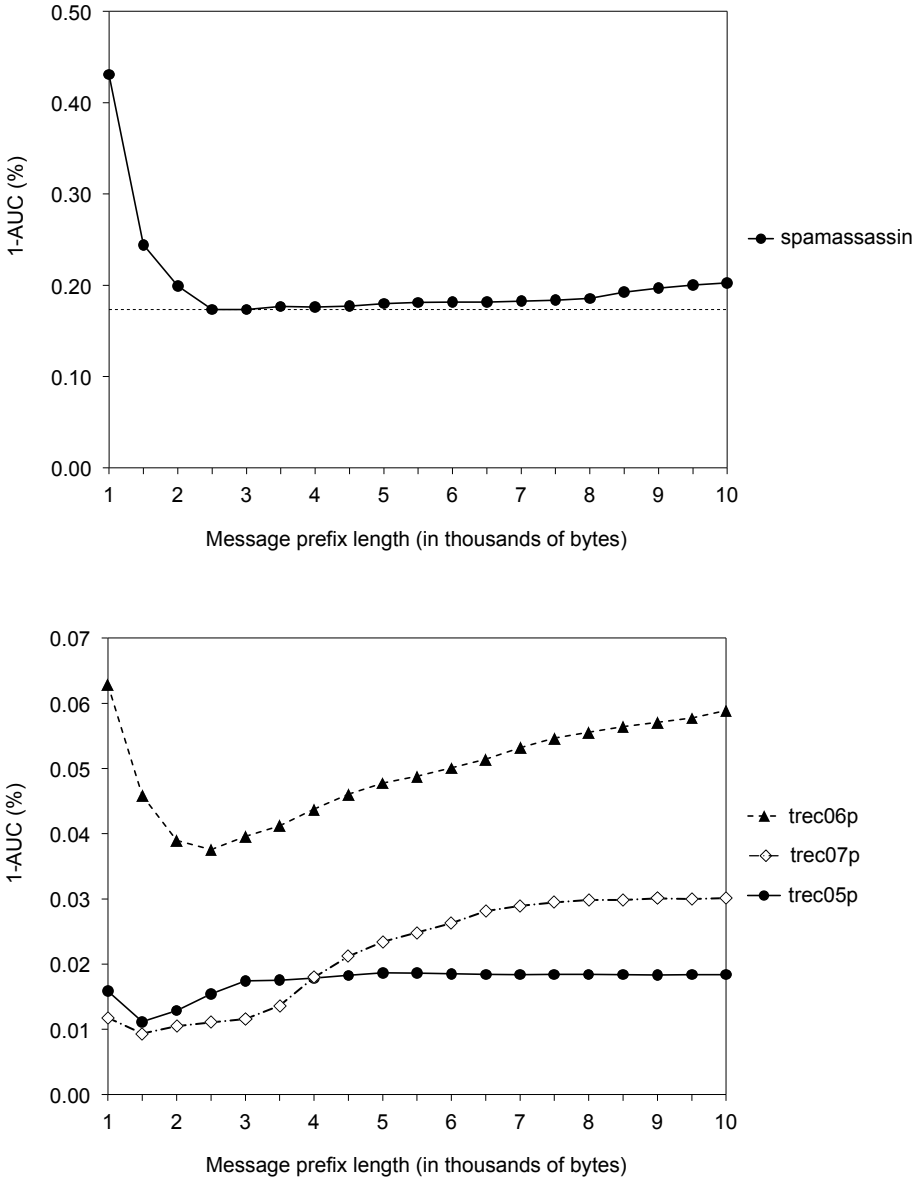


Figure 5.5: The effect of truncating messages on AUC performance using an adaptive order-6 PPM classifier on the Spamassassin dataset (top) and the TREC public spam corpora (bottom).

is consistent with observations of many other authors (e.g. Sculley et al., 2006; Sculley and Wachman, 2007a; Cormack, 2007a; Xu et al., 2007; Jorgensen et al., 2008). On the SpamAssassin dataset, performance of PPM peaks when messages are truncated at 2500 bytes, which is why we adopt this setting for other experiments, particularly for comparisons with other spam filters on the TREC public corpora, thus effectively using the SpamAssassin corpus as a tuning set.

5.5.4 Resilience to Text Obfuscation

One of the perceived advantages of statistical data compression models over standard text categorization algorithms is that they do not require any preprocessing of the data. Classification is not based on word tokens or manually constructed features, which is in itself appealing from the implementation standpoint. For spam filtering, this property is especially welcome, since preprocessing steps are error-prone and are often exploited by spammers in order to evade filtering. A typical strategy is to distort words with common spelling mistakes or character substitutions, which may confuse an automatic filter. We believe that compression models are much more robust to such tactics than methods which require tokenization.

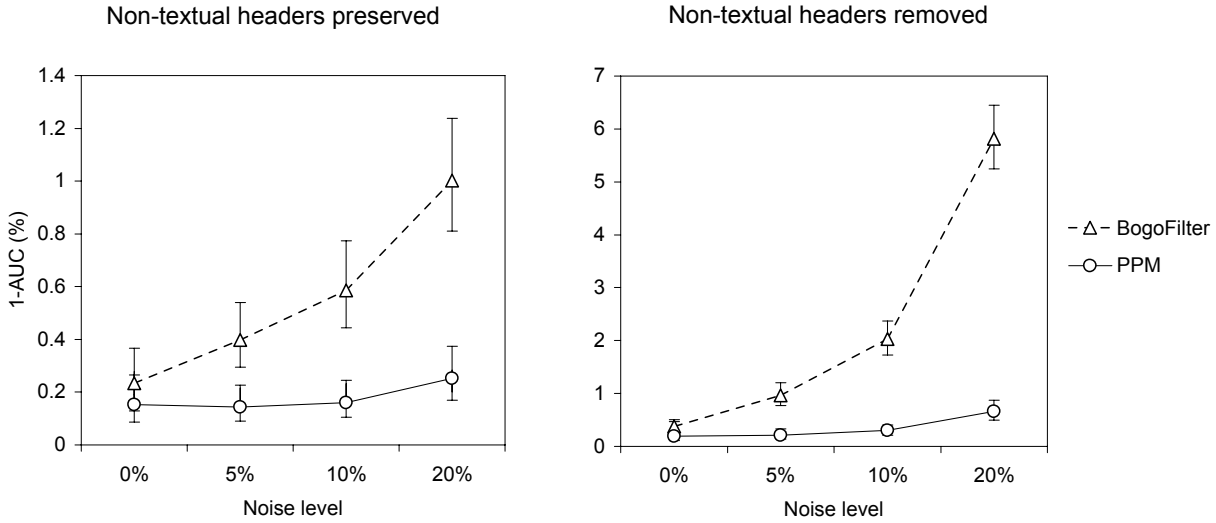


Figure 5.6: Effect of adding noise to the text on classification performance on the SpamAssassin dataset. The graph shows 1–AUC scores with 95% confidence intervals for PPM and the tokenization-based Bogofilter classifier at different levels of artificial random noise in the data.

To support this claim, we conducted an experiment in which all messages in the SpamAssassin dataset were distorted by substituting characters in the original text with random alphanumeric characters and punctuation. The characters in the original message that were subject to such a substitution were also chosen randomly. By varying

the probability that a character is distorted, we evaluated the effect of such noise on classification performance.

For the experiment, all messages were first decoded and stripped of non-textual attachments. Noise was added to the subject and body parts of messages. Adaptive compression models and Bogofilter, a representative tokenization-based filter[9], were evaluated on the resulting dataset. We then stripped the messages of all headers except subjects and repeated the evaluation. The results of these experiments are depicted in Figure 5.6. They show that compression models are indeed very robust to noise. Even after 20% of all characters are distorted, rendering messages practically illegible, PPM retains respectable performance. The advantage of compression models on noisy data is particularly pronounced in the second experiment, where the classifier must rely solely on the (distorted) textual contents of the messages. It is interesting to note that the performance of PPM increases slightly at the 5% noise level if headers are kept intact. We have no definitive explanation for this phenomenon. We suspect that introducing noise in the text implicitly increases the influence of the non-textual message headers, and that this effect is beneficial.[10]

We emphasize that all messages, whether legitimate email or spam, were equally distorted in our experiment. In real-world settings, we would only expect obfuscated content in spam. However, if we only distort spam in our experiment, the filters' task becomes one of distinguishing distorted messages (spam) from undistorted messages (ham), which turns out to be easier than the original task of distinguishing spam from legitimate email. Results of compression methods and tokenization-based filters similarly improve in this case. In actuality, spammers do not distort their messages randomly as in our experiment. Instead, spammers obfuscate only keywords that they suspect are indicative of spam, and might further mask the obfuscated content by adding irrelevant innocent-looking text to their messages. Our experiment is too crude to replicate such tactics, however, it does show that compression methods are generally more capable of learning from distorted text. Given this result, it is reasonable to conclude that compression methods are less affected by obfuscation of specific keywords commonly found in spam messages.

[9]Bogofilter (`http://www.bogofilter.org`) is an open source mail filter which exhibited good performance at the TREC 2005 evaluation (Cormack and Lynam, 2005b). Bogofilter version 0.94.0 with default parameters was used for this experiment.

[10]Note that messages were not truncated in this experiment, in-line with other experiments using the SpamAssasin data at the time (cf. Bratko et al., 2006a).

5.5.5 Visualizations

The primary motivation for using data compression for spam filtering is that the approach circumvents the need for tokenization, so that patterns of punctuation and other special characters which are otherwise hard to model as a 'bag-of-words' are naturally included in the model. Figure 5.7 shows some examples of the types of patterns picked up by the PPM compression model. The visualization shows how the PPM classifier "sees" messages. Character positions which are neutral are rendered in a light gray color, characters that indicate spam are colored in purple, while characters that point towards legitimate mail are colored progressively green.

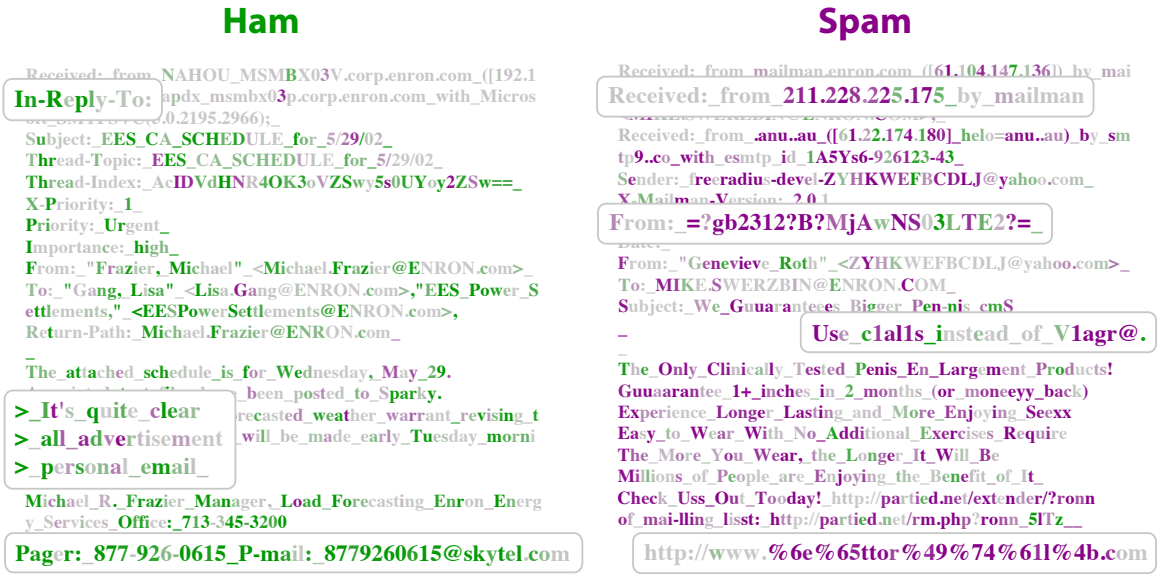


Figure 5.7: A visualization of how PPM filters email. Each character position is colored according to the relative character probability given by two PPM compression models – one trained on spam and one trained on good email. Purple indicates spam, green indicates legitimate email (ham). The examples were extracted from the SpamAssassin corpus through random manual inspection of messages.

The highlighted examples show character substitutions typically used by spammers, which are recognized easily by the PPM filter, as well as URL obfuscation tactics and the use of non-standard character sets. All of these techniques are intended to hide text from tokenization-based filters. Some interesting patterns that are typical of legitimate email are also highlighted, such as message headers that are only included in replies and line prefix characters that usually accompany quoted text in forwarded messages. These patterns are usually not preserved by bag-of-words style tokenization, as they are based on word phrases (i.e. "`in+reply+to`") or punctuation. Two examples of non-trivial patterns involving numbers are also shown, namely IP addresses and alternative

delimitation of phone numbers, both of which would require special consideration when using traditional parsers.

5.6 Comparisons with other methods

In this section, we compare compression-based spam filtering with results obtained using other spam filters on the same data. The comparisons include best-performing systems from spam filter evaluations conducted at TREC (Cormack and Lynam, 2005b; Cormack, 2006, 2007b), a selection of text classifiers optimized for spam filtering and evaluated on the TREC public datasets by Sculley (2008), and two open source mail filters: The "industry standard" SpamAssassin filter and Bogofilter – a filter which exhibited consistently good performance in the TREC 2005 evaluation (Cormack and Lynam, 2005b), and is included in the TREC spam filter evaluation toolkit[11]. All spam filters used for comparisons on the TREC corpora are listed in Table 5.4, together with references to the original publications and links to relevant online resources, where available.

5.6.1 The TREC Spam Track Evaluations

The Text REtrieval Conference (TREC) is an annual series of workshops, founded to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. The organization of the conference, and the effort required to gather and process the datasets and results, are co-sponsored by the US National Institute of Standards and Technology (NIST).

From 2005 to 2007, TREC included a track on spam filtering, with the goal of providing a standard evaluation of current and proposed spam filtering approaches (Cormack and Lynam, 2005b; Cormack, 2006, 2007b). Each year, the track coordinators prepared several large, high-quality email datasets to facilitate the evaluation. The datasets were adjudicated using a semi-automatic iterative process detailed in (Cormack and Lynam, 2005a): Starting from an initial label assignment, several reference spam filters were run, and all messages with uncertain classification according to any one of the filters were re-adjudicated manually. Additional randomly-selected messages were also inspected. This process was repeated several times until manual re-adjudication revealed no further label errors.

[11] The spam filter evaluation toolkit is a software package used for standard evaluation of spam filters that implement the TREC interface. Available at `http://plg.uwaterloo.ca/~gvcormac/jig/`.

Open source spam filters

Bogofilter-a	A popular, free mail filter based on the Fisher inverse chi-square technique (Robinson, 2003) – a derivative of naive Bayes. Version 0.95.2, as configured for TREC 2005 by the track coordinators (`http://www.bogofilter.org`).
Bogofilter-b	Bogofilter version 1.1.5.
SpamAssassin	SpamAssassin version 3.3.2, learning component only (`http://spamassassin.apache.org`).

Best performing filters at TREC evaluations

CRM114/OSB (Assis et al., 2005)	CRM114 (`http://crm114.sourceforge.net`), as configured for TREC by the authors, labeled `crmSPAM3` at TREC 2005. Uses a linear classifier similar to naive Bayes, combined with several additional empirically-derived heuristics. Operates on a special gappy word-pairs feature space known as orthogonal sparse bigrams (OSB), binary feature vectors. Trained only on uncertain examples.
OSBF-Lua (Assis, 2006)	OSBF-Lua (`http://osbf-lua.luaforge.net`), as configured for TREC by the authors. Linear naive Bayes-like classifier, combined with several additional empirically-derived heuristics. OSB features. Trained only on uncertain examples. Aggressive training on email headers. Labeled `oflS1` at TREC 2006.
t-Perceptron (Sculley et al., 2006)	Perceptron with margins (Krauth and Mézard, 1987). Features drawn from variable-length, inexact character n-grams with "gaps" and "wildcards". Binary features, l^2-normalized vectors. Labeled `tufS1F` at TREC 2006.
t-ROSVM (Sculley and Wachman, 2007b)	A linear SVM variant optimized for online learning (Sculley and Wachman, 2007a), binary 4-gram features, l^2 norm, SVM cost parameter $C = 100$. Labeled `tftS3` at TREC 2007.
w-LR (Cormack, 2007a)	Gradient descent logistic regression (Goodman and Yih, 2006), binary 4-gram features, no document vector normalization. Labeled `wat1` at TREC 2007.
DMC (Cormack, 2007a)	Compression-based filter using dynamic Markov compression (see Bratko et al., 2006a), labeled `wat2` at TREC 2007.
w-LR+DMC (Cormack, 2007a)	Ensemble of w-LR and DMC. Base classifier outputs combined using logistic regression. Labeled `wat3` at TREC 2007.

Selected text classifiers

MNB (Sculley, 2008)	Multinomial naive Bayes (Metsis et al., 2006) with binary features drawn from the first 3000 bytes of each message.
Perceptron-c (Sculley, 2008)	The classic perceptron algorithm (Rosenblatt, 1958), binary features, l^2-normalized document vectors.
Perceptron-m (Sculley, 2008)	Perceptron with margins (Krauth and Mézard, 1987), binary features, l^2-normalized document vectors.
LR (Sculley, 2008)	Logistic Regression using an online gradient descent update rule (Goodman and Yih, 2006), l^2-norm, binary features.
SVM (Sculley, 2008)	Online linear SVM (Kivinen et al., 2004) with full SVM optimization, cost parameter $C = 100$, l^2-norm, binary features.

Table 5.4: Reference systems for which we compare results on the TREC data.

The datasets used at TREC included private messages which were not disclosed to the workshop participants. Instead, participants were asked to implement a standard spam filter interface and submit up to four filter configurations for evaluation. The filters were run by the track coordinators and summary results were reported back to the participants. Publicly available corpora were also released each year, providing a standard benchmark for future spam filter evaluation and comparison. A high degree of agreement between results obtained on public and personal email corpora was observed in the official TREC evaluations, validating the use of the public corpora also for the evaluation of personal spam filters (Lynam, 2009). The scale of the evaluation in terms of the number of participating groups and the number of filters evaluated is summarized in Table 5.5.

	TREC '05	TREC '06	TREC '07
Universities and research institutions	8	6	11
Corporate research labs	1	-	-
Practitioners/developers	3	3	1
Non-participant reference filters	(5)	-	-
Total number of groups	12 (17)	9	12
Total filter configurations	44 (53)	32	36

Table 5.5: Spam track participation for each of the TREC workshops from 2005 to 2007. Numbers in parentheses include non-participant open source filters which were also evaluated by the TREC 2005 spam track coordinators.

Compression-based Filters at TREC

The use of data compression models for spam filtering was first proposed in the framework of the TREC 2005 spam track (Bratko and Filipič, 2005). A filter using adaptive order-6 PPM compression models emerged as the best performing method in the TREC 2005 evaluation (Cormack and Lynam, 2005b, labeled `ijsSPAM2` in the report). The PPM implementation used at TREC 2005 had relatively large memory requirements, and was designed to discard part of the training data when memory limits were reached in online filtering experiments. This mechanism was used up to twice per evaluation run, depending on the size of the dataset. Furthermore, the PPM filter at TREC 2005 used an alphabet of only 72 commonly used ASCII characters; all other byte codes were mapped to a single symbol.

An order-6 adaptive PPM filter was also submitted for evaluation at TREC workshops in 2006 and 2007, using a more efficient implementation in terms of memory use and filtering speed (Bratko et al., 2006b). The difference in implementation was however found to have little effect on filtering performance. For example, the newer,

more efficient PPM implementation achieved an 1–AUC(%) score of 0.017 on `trec05p`, compared to 0.019 for the original TREC 2005 version of the filter.

Spam Track Results Overview

The full results of the TREC spam track evaluations occupy over a hundred pages of appendices in each of the workshop proceedings (Cormack and Lynam, 2005b; Cormack, 2006, 2007b). A high level summary of the official results, over all three installments of the TREC spam track, is given in Table 5.6. The best and median 1–AUC result is reported for each dataset over all filter configurations that successfully completed the test. This is compared to the official 1–AUC result of our order-6 adaptive PPM filter. We also report the rank of the PPM-6 filter among all tested filter configurations, as well as the rank of PPM-6 when compared only with the best-performing filter on the target dataset from each group (subtracting 1 from this rank gives the number of groups "able to beat" PPM for the target dataset). The best-performing filter is also noted for each trial.

TREC 2005. Spam filtering based on adaptive data compression models, specifically an order-6 PPM filter (Bratko and Filipič, 2005), was the best performing method in the TREC 2005 evaluation (Cormack and Lynam, 2005b). Also competitive were several open source spam filters, specially configured or adapted for TREC by their developers.[12] Classification algorithms used in these filters include variants of naive Bayes (Assis et al., 2005; Meyer, 2005), the Winnow algorithm (Assis et al., 2005) and maximum entropy (logistic regression) models (Breyer, 2005). A plethora of parsing and feature extraction techniques are used in these filters. An even greater variety of classification algorithms and techniques was tested by various other groups participating in the evaluation. Descriptions of these systems are available in the TREC proceedings.

TREC 2006. Our order-6 PPM filter was also competitive to the best filters in the TREC 2006 evaluation (cf. Bratko et al., 2006b; Cormack, 2006). The best performing filter at TREC 2006 was, however, OSBF-Lua (Assis, 2006), a linear classifier combining naive Bayes with a unique, empirically-derived feature weighting, a feature space based on sparse word pairs, and an aggressive training strategy which emphasizes difficult examples and message headers (see Assis et al., 2006). Yerazunis (2006) also consider a compression-based filter at TREC 2006 with encouraging results. A noise-tolerant variant of the perceptron algorithm by Sculley et al. (2006), using inexact character

[12] Assis et al. (2005): CRM114; Meyer (2005): SpamBayes; Breyer (2005): dbacl.

TREC 2005

		1–AUC (%)			PPM rank among	
Dataset	Best filter	Best	PPM	Median	all runs	groups
`trec05p`	PPM	0.019	0.019	0.688	1 of 49	1 of 15
`MrX`[a]	Bogofilter-a	0.045	0.069	0.997	2/7 of 53	2/4 of 17
`SB`	CRM114/OSB	0.231	0.285	2.845	2 of 53	2 of 17
`TM`	PPM	0.135	0.135	2.626	1 of 49	1 of 16

TREC 2006

		1–AUC (%)			PPM rank among	
Dataset	Best filter	Best	PPM	Median	all runs	groups
`trec06p`	OSBF-Lua	.0540	.0605	.2605	7 of 30	3 of 9
`trec06c`	t-Perceptron	.0023	.0083	.0526	6 of 30	3 of 9
`MrX2`	OSBF-Lua	.0363	.0809	.1498	8 of 31	3 of 9
`SB2`	OSBF-Lua	.1249	.1633	.5792	5 of 32	2 of 9

TREC 2007

		1–AUC (%)			PPM rank among	
Dataset	Best filter	Best	PPM	Median	all runs	groups
`trec07p`	w-LR+DMC	.0055	.0299	.0406	14 of 36	4 of 12
`MrX3`	t-ROSVM	.0042	.0397	.0597	14 of 30	6 of 12

Table 5.6: A summary of the official TREC spam track results for the immediate feedback task.

[a] In TREC 2005 tests on the MrX dataset, the standard order-6 PPM filter was ranked 7th among all filter configurations, and 4th when compared to the best-performing filters from each of the other groups. However, order-4 and order-8 PPM filters were also evaluated at TREC 2005. In fact, the order-8 PPM filter was our best run on the MrX dataset, which achieved the second-best result among all 53 filter configurations tested by the track coordinators on this corpus.

substrings of varying length as features, also performed exceptionally well. As a side-note, we mention that the TREC 2006 spam track included a pool-based active learning task, in which our approach that favored training on recent examples proved successful (Bratko et al., 2006b).

TREC 2007. Our "vintage" order-6 PPM filter was again submitted for evaluation at the TREC 2007 spam track (Cormack, 2007b), in which our results were more modest, with performance just above the median of all filters tested. Two online adaptations of discriminative classifiers showed superiority at TREC 2007, namely relaxed online SVM (Sculley and Wachman, 2007b) and online logistic regression (Cormack, 2007a). Both of these filters used 4-grams of characters as features and aggressive learning strategies (low regularization). Other participants also submitted compression-based filters for evaluation at TREC 2007 (e.g. Kato et al., 2007; Cormack, 2007a). In particular, the filter based on dynamic Markov compression by Cormack (2007a) exhibited excellent performance, while a combination of logistic regression and dynamic Markov compression (Cormack, 2007a) was arguably the best performing filter overall.[13]

Summary and remarks

New filtering methods and techniques emerged at each of the TREC workshops, raising the bar for spam filter performance each year (cf. Cormack and Lynam, 2005b; Cormack, 2006, 2007b). While our system based on PPM compression was clearly a winning proposition in 2005, more recent evaluations favor discriminative classification algorithms (although various compression-based methods remain competitive).

In parallel with emerging classification algorithms, feature extraction and modeling techniques also evolved. One major development is the use of character-level features instead of features requiring tokenization. At TREC 2005, only two out of eleven participating groups (other than ourselves) considered the use of character-based features. Moreover, only one group submitted a filter configuration capable of operating completely without word-based tokenization, and none of the other groups used character-level features for their primary submission. Five open source filters additionally tested at TREC 2005 by the track coordinators also relied on bag-of-words style tokenization (in addition to various hand-crafted features). Two years later, at TREC 2007, all spam track participants for which we were able to obtain detailed system descriptions made use of character-level features. This includes all of the best-performing filters at TREC 2007 (Sculley and Wachman, 2007b; Cormack, 2007a; Xu et al., 2007). We note that

[13]Note that the official results for the TREC 2007 evaluation did not include aggregate filter scores over all corpora, and no filter performed uniformly best for all datasets.

some spam track participants refer to the success of our PPM-based filter at TREC 2005 as part of the motivation for exploring character-level document representations (cf. Sculley et al., 2006; Xu et al., 2007).

A review of the TREC results and system descriptions reveals another common development, namely the practice of truncating messages beyond a certain length threshold. For example, the three most successful groups in the TREC 2007 evaluation use this optimization in all of their filter configurations (Sculley and Wachman, 2007b; Cormack, 2007a; Xu et al., 2007). Our PPM filter submitted for TREC did not truncate messages, and it is the prevailing use of this technique at the TREC 2007 workshop that prompted us to consider this modification. Indeed, we find that message truncation is beneficial also when using PPM compression for filtering (see Section 5.5.3).

5.6.2 Comparison Using the TREC Public Datasets

We compare online spam filtering performance of data compression models against a variety of other spam filters in Table 5.7, using the three public TREC corpora as the testbed. The results in Table 5.7 are mainly compiled from various publications, although some additional evaluations were also performed for the purpose of this comparison.[14] The source of each result is given together with the values in the table. The comparison consolidates the results of the TREC evaluations over the years and compares compression models to the current state-of-the-art.

The results of both PPM and DMC compression methods are obtained using adaptive models. The DMC filter results were obtained using default model parameters which would be suitable also for data compression (as described in Bratko et al., 2006a). The PPM results were produced using the standard order-6 PPMD method. Both compression methods use only the first 2500 bytes of each message.[15]

The comparison includes the best-performing filters at the TREC evaluations, which were already described in the previous section, as well as two popular open source filters – SpamAssassin and Bogofilter. Where available, we also report results obtained by Sculley (2008) for an assortment of online text classifiers. The data modeling choices used for these classifiers, namely the use of character 4-grams, binary features, document vector normalization, and message truncation, were experimentally validated or based on previous practical experience, while the parameters of each of the tested algorithms were tuned to maximize performance on the SpamAssassin dataset (for details see Sculley, 2008).

[14] Additional results for TREC filters were obtained with the assistance of the spam track coordinators.

[15] The truncation length for PPM is chosen based on experiments using the SpamAssassin dataset. The same value was also used by Cormack (2007a) for the DMC filter at TREC 2007 and in (Bratko

Method/Dataset	trec05p	trec06p	trec07p
TREC winning result			
PPM | oflS1 | wat3	.019 (.015-.023)[a]	.054 (.034-.085)[b]	.006 (.002-.016)[c]
Popular spam filters			
SpamAssassin	.055 (.042-.072)[⋆]	.131 (.097-.176)[⋆]	.082 (.058-.115)[⋆]
Bogofilter-b	.048 (.038-.062)[⋆,a]	.087 (.066-.114)[⋆,f]	.027 (.017-.043)[⋆,f]
Selected text classifiers, words			
MNB	.351 (.326-.378)[g]	.145 (.116-.181)[g]	N/A
Perceptron-c	.105 (.091-.120)[g]	.163 (.130-.205)[g]	.042 (.027-.067)[g]
Perceptron-m	.037 (.030-.046)[g]	.047 (.034-.064)[g]	.019 (.011-.032)[g]
LR	.046 (.041-.052)[g]	.087 (.073-.104)[g]	.011 (.006-.021)[g]
SVM	.015 (.011-.022)[e]	.034 (.025-.046)[e]	N/A
Selected text classifiers, character 4-grams			
MNB	.871 (.831-.913)[g]	.477 (.426-.535)[f,g]	.168 (.151-.185)[f]
Perceptron-c	.060 (.051-.070)[g]	.229 (.186-.282)[g]	.050 (.033-.073)[g]
Perceptron-m	.022 (.018-.027)[g]	.049 (.034-.070)[g]	.011 (.006-.019)[g]
LR	.013 (.011-.015)[g]	.032 (.025-.041)[f,g]	.006 (.002-.016)[f,g]
SVM	.008 (.007-.011)[e]	.023 (.017-.032)[e]	N/A
Compression models			
DMC	.013 (.010-.018)[d]	.031 (.024-.041)[f]	.008 (.003-.017)[c,f]
PPM	.014 (.010-.019)[⋆]	.028 (.021-.039)[⋆]	.008 (.003-.022)[⋆]
TREC '06 and '07 top performers			
OSBF-Lua (oflS1)	.011 (.008-.014)[⋆]	.054 (.034-.085)[⋆,b,e]	.029 (.015-.058)[⋆,e]
t-ROSVM (tftS3)	.010 (.008-.013)[⋆]	.031 (.021-.044)[f]	.009 (.002-.041)[c,f]
w-LR (wat1)	.012 (.010-.015)[⋆]	.036 (.027-.049)[f]	.006 (.002-.017)[c,f]
w-LR+DMC (wat3)	.010 (.008-.012)[⋆]	.028 (.021-.037)[⋆]	.006 (.002-.016)[c]

Table 5.7: Comparison of 1–AUC(%) scores with 95% confidence intervals on TREC public spam corpora. In addition to PPM and DMC data compression models, the comparison includes best-performing filters from TREC in 2006 and 2007, as well as two popular open source spam filters, and a selection of online text classifiers due to Sculley (2008). Note that the DMC compression-based filter was also evaluated at TREC 2007 under the tag name wat2.

Superscripts denote the source of the results (multiple annotations indicate that the same result is found in multiple sources):

⋆ – This study
a – TREC 2005 official results (Cormack and Lynam, 2005b)
b – TREC 2006 official results (Cormack, 2006)
c – TREC 2007 official results (Cormack, 2007b)
d – Bratko et al. (2006a)
e – Sculley and Wachman (2007a)
f – Sculley and Cormack (2008)
g – Sculley (2008)

Comments and conclusions

Several conclusions can be drawn from the comparison in Table 5.7:

- There is little difference in filtering performance between the two compression-based methods (PPM and DMC).
- Compression-based filters, as well as other successful filters from the TREC evaluations, outperform popular and widely-deployed open source filters.
- The standard naive Bayes classifier, which is almost anecdotally associated with spam filtering, exhibits comparatively poor performance.
- Compression-based filters perform uniformly better than tokenization-based text classifiers.
- For the better-performing discriminative text classifiers, a feature space based on character n-grams consistently improves filtering performance compared to tokenization-based features.
- The compression-based filters match or exceed the performance of text classifiers based on character n-grams except for online SVM. Note however that the training cost of online SVMs is expensive compared to compression methods, and is "*prohibitive for large scale applications*" (Sculley and Wachman, 2007a).
- The best performing filters from later TREC workshops improve upon the results obtained in previous TREC evaluations when tested on the same data, indicating that spam filters have generally evolved in the course of the TREC evaluations (see also Lynam, 2009).
- Compression methods are competitive to the most successful filters tested at any of the TREC evaluations (the large overlaps in confidence intervals should be taken into account when comparing results, particularly for `trec06p`).
- Fusion of DMC and logistic regression yields some of the best results, exceeding the performance of the base classifiers in all tests, which indicates that compression models and discriminative text classifiers are not only competitive, but also complementary to each other.

---

et al., 2006a), allowing an apples-to-apples comparison.

5.6.3 Comparison with Prior Methods

In this section, we report the results of cross-validation experiments originally published in (Bratko et al., 2006a), in which PPM and DMC compression models were evaluated on the Ling-Spam, PU1 and PU3 datasets with pre-defined cross-validation splits. An implementation of the perceptron and SVM classifiers, as well as Bogofilter, a well-performing open source filter, were also tested in the study. All of our reference filters used tokenization-based features, in-line with previous publications using the same data. The performance of these filters was compared to results reported in prior studies using the Ling-Spam, PU1 and PU3 corpora, which were available at the time and from which it was possible to determine spam misclassification and false positive rates from the published results. Table 5.8 lists all systems that were included in this comparison, as well as the source of results reproduced from previous studies using the same data. Results of the cross-validation experiments are presented in Figures 5.8 and 5.9. The simplified ROC-style graphs plot the number of misclassified spam messages against the number of false positives.

On the Ling-Spam dataset (Figure 5.8), both compression models were found to perform comparably to the suffix tree approach of Pampapathi et al. (2006). These three classifiers dominate the other methods which were included in the comparison. Both the suffix tree classifier and the compression methods operate on character-level or binary sequences. All other previous methods and our reference filters used the standard bag-of-words representation for modeling text. This suggests that methods which model text as a sequence of symbols are more suitable than tokenization-based methods for the Ling-Spam dataset, and we believe this to be the case for spam filtering in general.

The PU1 and PU3 datasets contain pre-tokenized messages in which the original words are replaced with numeric identifiers. This representation was converted to binary format by replacing token identifiers with their 16 bit binary equivalents for experiments with the DMC classifier, since DMC uses a binary alphabet and is hurt by the artificial tokenization of these datasets. The PPM classifier was tested on the unprocessed original version of the PU1 and PU3 datasets. Digits in numeric identifiers were converted to alphabetical characters for experiments with Bogofilter, since the filter was found to handle digits differently from alphabetical character strings.

Figure 5.9 shows classification performance on the PU1 and PU3 datasets, which is perhaps the most surprising result reported in (Bratko et al., 2006a). Both compression models were found to outperform other classifiers almost uniformly across the range of 'interesting' filtering thresholds, despite the fact that the datasets contain

Label		Description
Bogofilter-c	⋆	Bogofilter version 0.94.0, default parameters (`http://www.bogofilter.org`).
SVM-c	⋆	An adaptation of the SVM^{light} package (Joachims, 1998a) for the PU1 dataset due to Tretyakov (2004), linear kernel with C = 1.
Perceptron-c	⋆	Implementation of the perceptron algorithm due to Tretyakov (2004).
a-Bayes	◇	Naive Bayes, multi-variate Bernoulli model with binary features (Androutsopoulos et al., 2000).
a-FlexBayes	◇	Flexible naive Bayes – uses kernel density estimation for estimating class-conditional probabilities of continuous valued attributes (Androutsopoulos et al., 2004).
a-LogitBoost	◇	LogitBoost (variant of boosting) with decision stumps as base classifiers (Androutsopoulos et al., 2004).
a-SVM	◇	Linear kernel support vector machines (Androutsopoulos et al., 2004).
c-AdaBoost	◇	Boosting of decision trees with real-valued predictions (Carreras and Márquez, 2001).
gh-Bayes	◇	Naive Bayes (exact model unknown) with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
gh-SVM	◇	Linear support vector machine with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
h-Bayes	◇	Multinomial naive Bayes (Hovold, 2005).
ks-Bayes	◇	Multinomial naive Bayes (Schneider, 2003).
p-Suffix	◇	Pattern matching of character sequences based on the suffix tree data structure and various heuristic scoring functions (Pampapathi et al., 2006).
m-Filtron	◇	Support vector machines with linear kernels (Michelakis et al., 2004).
s-Stack	◇	Stacking of naive Bayes and k-nearest neighbors (Sakkis et al., 2001).
s-kNN	◇	k-nearest neighbors with attribute and distance weighting (Sakkis et al., 2003).

Table 5.8: Reference systems and results of previous studies using cross-validation experiments on the LingSpam, PU1 and PU3 datasets. Entries are delimited by primary authors.

Symbols indicate the source of reported results:

⋆ – Bratko et al. (2006a)
◇ – Reproduced from other studies

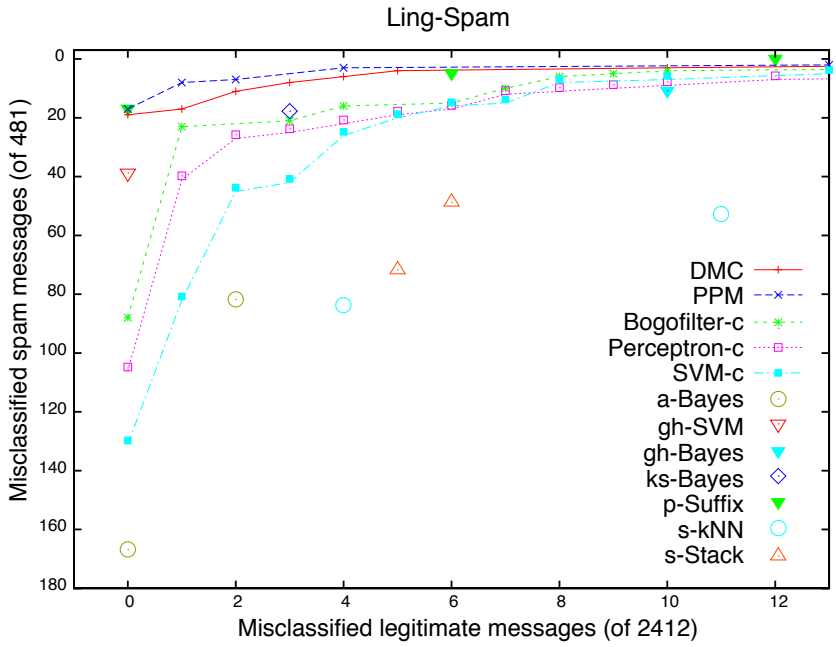


Figure 5.8: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and prior published results on the Ling-Spam dataset (Bratko et al., 2006a).

pre-processed data specifically prepared for tokenization-based filters. The good performance of compression models in these tests is attributed to the fact that tokens are *not* considered independently. Previous studies using word-level models have shown that modeling word dependencies can improve classification performance (Peng et al., 2004). By design, compression models can discover phrase-level patterns just as naturally as sub-word patterns. Marton et al. (2005) also find that compression models are capable of detecting phrase-level patterns that are useful for classification.

5.7 Discussion

We conclude this chapter with a summary of contributions of this thesis for spam filtering, and an outlook on future challenges for advancing compression-based spam filters.

Contributions

The use of compression-based methods for spam filtering was proposed in (Bratko and Filipič, 2005). This approach emerged as the winning method in the TREC 2005 spam filter competition (Cormack and Lynam, 2005b). Compression methods were also shown to match or improve prior published results, when evaluated on the same datasets using

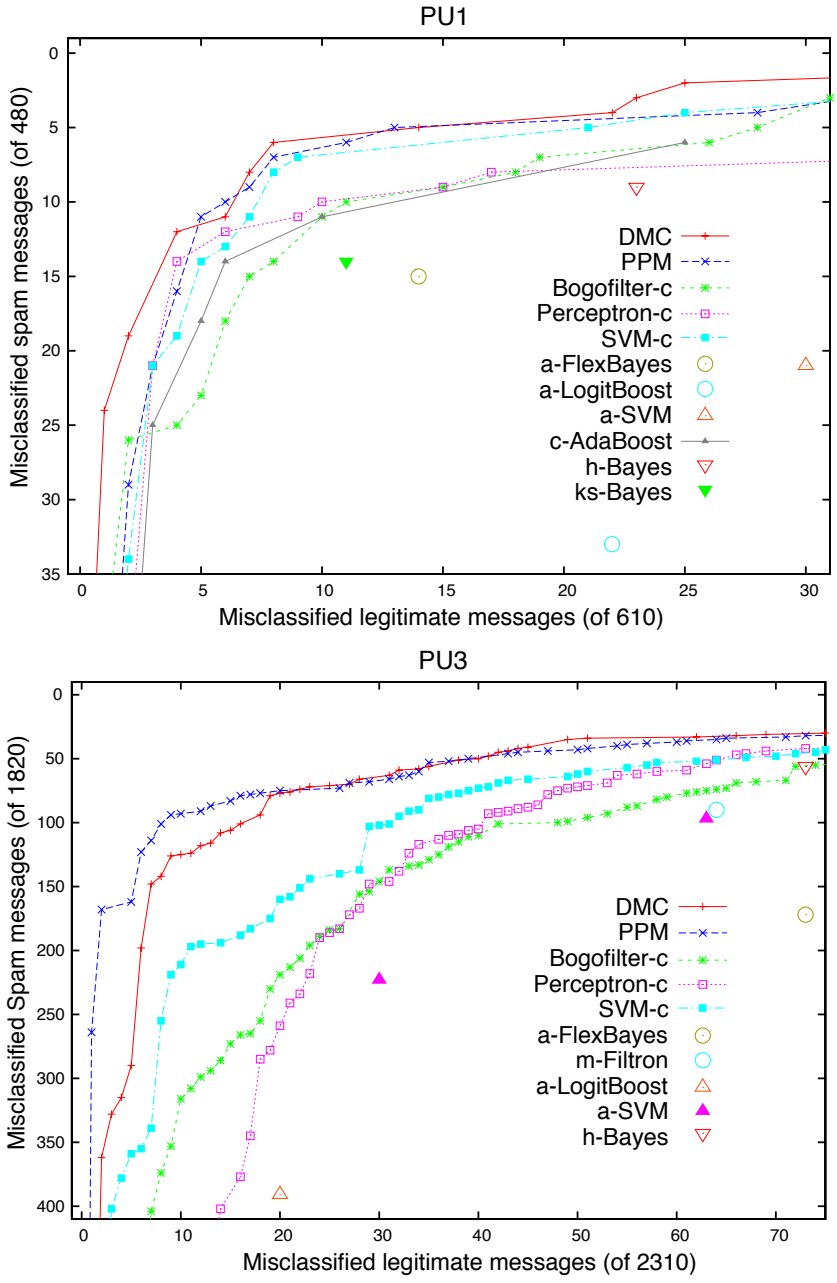


Figure 5.9: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and prior published results on the PU1 (top) and PU3 (bottom) datasets (Bratko et al., 2006a).

equivalent cross-validation experiments as those used in prior studies (cf. Bratko et al., 2006a; Cormack and Bratko, 2006). Although spam filtering techniques have since improved, as evidenced by the results of subsequent TREC workshops, we find that compression models continue to be a viable and competitive approach for spam filtering, which is a view that is shared also by others (Goodman et al., 2007; Sculley, 2008; Cormack, 2008). Successful trends in spam filtering include a shift in email modeling from word tokens towards the use of character-based features, a development which was to some extent also motivated by the success of data compression methods.

We find that updating compression models adaptively when classifying a target document is beneficial for classification performance, particularly in the AUC measure. The modification has a natural interpretation in terms of the minimum description length principle. Although we are unaware of any parallel to this in existing text classification research, the same approach could easily be adapted for the popular multinomial naive Bayes model (McCallum and Nigam, 1998a) and possibly also for other incremental classifiers. We believe this to be an interesting avenue for future research.

Our experiments indicate that compression models are less affected by the type of noise introduced in text by typical obfuscation tricks used by spammers. This should make them difficult for spammers to defeat, but also makes them attractive for other text categorization problems that contain noisy data, such as classification of text extracted with optical character recognition. The robustness of data compression techniques in the adversarial spam filtering setting has later been studied by Jorgensen et al. (2008) and Zhou et al. (2011), who confirm that compression-based methods are generally less sensitive to various adversarial attacks.

Challenges

Email messages are semi-structured documents. For text classification tasks which do not require online learning, significant improvements in performance have been achieved by modeling document structure (e.g. Bratko and Filipič, 2006). Document structure is used to improve online spam filtering in the header reinforcement method of Assis (2006), which puts greater influence on message headers. However, online methods which would exploit document structure are not common and, as noted by Sculley (2008), it appears that the most effective filtering methods have primarily ignored document structure, which largely remains an untapped resource also for compression-based methods.

Online versions of discriminative classifiers such as logistic regression have proven to be superior to their generative vector-based counterparts (such as naive Bayes) when applied to spam filtering, particularly when documents are represented using character n-gram features. Since compression methods are based on generative models, it is natu-

ral to consider how such methods could be applied in a more discriminative manner. One principled way of using generative models in discriminative classification frameworks is via the kernel trick, for example, by using Fisher kernels (Jaakkola and Haussler, 1999) and TOP kernels (Tsuda et al., 2002). However, adapting such an approach for online filtering would be challenging due to the need for incremental training. Another possibility involves the use of several ad hoc training techniques adopted in other filters, such as selective training only on examples that lie within a margin of uncertainty (Dagan et al., 1997, Sec. 4.2), repeated training on examples that are difficult to classify (Assis, 2006), as well as various feature weighting methods that effectively amplify discriminative features (Assis et al., 2006; Hazen and Margolis, 2008). All of the latter techniques aim to focus the learner towards discriminative modeling, specifically by boosting the influence of features[16] or examples which have high discriminative power, while at the same time suppressing non-discriminative features and non-informative data points.

We and others have observed that normalization of message length by truncating messages beyond a certain cutoff length may improve filtering performance. For filters based on PPM data compression models, we also find that it may be sufficient to limit message length only when performing classification, while the training regime can remain unchanged. This indicates that instead of truncating messages at a certain predefined length, more flexible and adaptive learning-based schemes could be developed without compromising computational efficiency in online filtering settings. Such schemes should offer similar benefits in terms of normalizing message length and might further improve filtering performance, but should also be more resilient to the obvious attack of adding innocent text at the beginning of each message.

[16]Such feature weighting methods could presumably be used in compression-based filters to weight prediction contexts, i.e. different suffixes leading up to the current character. However, we have so far been unsuccessful with some preliminary attempts in this direction.

Chapter 6

Lexical Stress Assignment Using Compression Models

This chapter describes a learning-based approach for automatic lexical stress assignment for Slovenian, a problem with practical implications for Slovenian speech synthesis. The task is to predict stress from the orthographic presentation of a word-form. To this end, we generate all possible stressed forms of a word, and select the candidate solution that can be compressed best using a compression model trained from examples of stressed word-forms.

In dealing with the stress assignment task, we address the issue of modeling long-range dependencies, that is, modeling dependencies that span beyond the context which is used for sequential prediction (the *order* of the compression model). We also evaluate other tactics that affect the accuracy of stress prediction for Slovenian, and report experimental results that compare favorably to other methods considered for this task.

The material presented in this chapter is the result of joint work with Tea Tušar, Domen Marinčič and Tomaž Šef, who worked on stress assignment solutions based on machine learning classifiers, while we investigated the application of data compression models for this task. An initial comparison between automatic stress assignment methods using boosted decision trees and our PPM compression-based approach is reported in (Tušar et al., 2005).

Data compressors. The stress assignment scheme described in this chapter can in principle be implemented using any statistical (context-based) compression method. We report results for stress assignment using the prediction by partial matching (PPM) compression algorithm (Cleary and Witten, 1984). We initially also experimented with the context tree weighting (CTW) algorithm (Willems et al., 1995), which was found

to perform similarly for this application, but is considerably slower due to its more complex weighting scheme.

6.1 Problem Description

Grapheme-to-phoneme conversion is a core task in speech synthesis. Although this is a challenging task for many languages, for Slovenian, the conversion is rather straightforward if the stressed form of words is known. The conversion can be done on the basis of less than 100 context-dependent letter-to-sound rules, adapted from (SAZU, 1990) with over 99% accuracy (Šef, 2001). However, lexical stress is not easily inferred from written text, since syllables in a word-form can be stressed almost arbitrarily (Toporišič, 1984). Unlike most other languages, stress can vary with different inflected forms of the same word, as in the examples given in Table 6.1. This makes the task of automatic stress assignment for Slovenian word-forms quite difficult in practice (cf. Šef et al., 2002; Šef and Gams, 2004; Tušar et al., 2005; Žganec Gros et al., 2006; Rojc and Kačič, 2007; Marinčič et al., 2009).

Slovenian	**English translation**	**Morphological annotations**
Rada se *pél̲jeva*.	We like *to drive*.	present, dual, first person
Pêl̲ji, prosim.	*Drive*, please.	imperative, singular, second person
Ti moraš *peljá̲ti*.	You have *to drive*.	infinitive

Table 6.1: Stressed forms of the the verb *peljati* ('to drive') in different inflected forms. Note that accents denoting stress are normally omitted in written text.

In Slovenian, each syllable in a word contains exactly one vowel, which may be either stressed or unstressed. Although most proper word-forms have only one stressed syllable, word-forms with two or more stressed syllables are also common. The standard stress-bearing vowels are *a*, *e*, *i*, *o*, *u* and the *reduced vowel*. A stressed reduced vowel can appear instead of the grapheme *e* or before the consonant *r*, provided there are no other vowels around it. Stressed vowels differ according to *quality* (narrow and wide vowels – only for the stressed vowels *e* and *o*) and *duration* (short and long vowels). We deal only with the quality of stress and disregard stress duration, following previous studies (Šef and Gams, 2004). The vowels and the possible stressed forms that we consider are given in Table 6.2.

Text to speech systems typically use pronunciation dictionaries for grapheme-to-phoneme conversion. However, heuristics must be used to handle out-of-vocabulary word-forms, in which case lexical stress assignment is often an important step in the transcription to phonemes (cf. Šef et al., 2002; Gros et al., 2005; Rojc and Kačič, 2007).

Unstressed	Stressed forms
a	***á*** (stressed)
e	***é*** (narrow stressed), ***ê*** (wide stressed), ***è*** (stressed reduced vowel)
i	***í*** (stressed)
o	***ó*** (narrow stressed), ***ô*** (wide stressed)
u	***ú*** (stressed)
r	***ŕ*** (stressed reduced vowel preceding *r*)

Table 6.2: Possible stressed forms for each vowel. Only the vowels *e* and *o* have multiple types of stress. Although *r* is not a vowel, we use it to denote the reduced vowel that appears before the consonant *r*. Note that we do not consider stress duration, and that the graphemes used for the reduced vowel are unstandard due to typeface limitations.

Moreover, Žganec Gros et al. (2006) find that reliable automatic stress assignment is the major limiting factor towards greater automatization in the construction of pronunciation dictionaries, which otherwise requires laborious manual validation to ensure acceptable quality of the phonetic transcriptions.

6.2 Related Work

Our compression-based approach for stress assignment follows the joint sequence model scheme (Deligne et al., 1995; Bisani and Ney, 2008), an established method for grapheme-to-phoneme conversion, also known as the joint multigram or joint *n*-gram model (Galescu and Allen, 2001; Bisani and Ney, 2002; Chen, 2003; Vozila et al., 2003). This scheme uses sequential probability models to determine the likelihood of a candidate phonetic transcription given the input text. We adapt this scheme for the stress assignment task, use context-based data compression models to estimate the joint probability of a candidate solution, and propose a strategy of switching between multiple models to account for long range dependencies between stressed positions.

Other related work falls into three categories: Studies of the stress assignment problem for the Slovenian language, work on stress assignment and related problems for other languages, and related compression-based methods, developed for other applications.

Lexical Stress Assignment for Slovenian

Šef and Gams (2004) compare stress assignment rules originally created by Slovenian linguistic experts and methods based on decision trees, which are found to be more accurate than expert-defined rules. Decision trees are trained to predict stress for each vowel independently based on surrounding letters and other features derived from the text. We compared boosted decision trees and compression models for stress assignment

in (Tušar et al., 2005).

The classification-based methods of Šef and Gams (2004) and Tušar et al. (2005) are later improved by Marinčič et al. (2009), by using a probabilistic heuristic to combine the predictions on individual vowels to predict stress for an entire word-form. Additional linguistic features were incorporated into the representation of examples, and several additional classification algorithms were evaluated. We compare these methods with our compression-based approach in our experimental evaluation.

In their text to speech system, Gros et al. (2005) predict stress for out-of-vocabulary words using hand-crafted rules which rely on (un)stressable word prefixes and suffixes. In cases where none of the rules apply, the most likely accent pattern is predicted based on the number of syllables in the word-form. We note that the rules used by Gros et al. (2005) are adapted from the same original source (Toporišič, 1984) as the expert-defined rules which are evaluated in our later experiments.

The Slovenian corpus-based text-to-speech system PLATTOS (Rojc and Kačič, 2007) uses classification and regression tree models to predict stress for out-of-vocabulary word-forms. This is in spirit similar to the use of decision trees by Šef et al. (2002), although it is not clear which features are used to represent syllables and words to predict stress in PLATTOS.

Stress Assignment for Other Languages

Black et al. (1998) use decision trees to predict stress for English using the context of letters surrounding stress-bearing vowels as features. Pearson et al. (2000) also use a decision tree approach for vowel stress prediction, which is augmented with an algorithm that uses word-level statistics of the frequency of different patterns of stress placement. Webster (2004) describe a greedy algorithm to iteratively find orthographic prefixes or suffixes which correlate most strongly with stress in a particular location, and evaluate the approach for English and German. A hidden Markov model (HMM) is used for stress assignment in English by Taylor (2005). The sequence of stress-bearing vowels and the consonant 'y' are used as the observed output when training the HMM, while stress is modeled with the hidden states. Dou et al. (2009) train SVM classifiers to rank possible full-word stress assignment patterns. Careful feature engineering is used to make the problem tractable, given that full-word solutions are presented as individual examples to the classifier.

A joint sequence model approach using n-gram language models to assess the likelihood of a stress assignment solution is described by Demberg et al. (2007), and evaluated for German and English. They report major improvements in stress assignment accuracy when the sequential prediction model is augmented to condition on the number of

preceding stressed vowels, which essentially reproduces our model switching approach described in (Tušar et al., 2005). Standard letter-level n-gram models are used also by Ungurean et al. (2009), to evaluate possible stress assignments for Romanian.

Related Applications of Data Compression Models

The general idea of using data compression models to evaluate the likelihood of a candidate solution, where the solution is obtained by somehow transforming or augmenting a given input text, has arisen in several other application domains, most notably for correction of errors in English text (Teahan et al., 1998), Chinese word segmentation (Teahan, 2000), and generic extraction of entities from unstructured text (Witten et al., 1999a). These related applications are described in greater detail in our review of compression-based methods in text mining (Section 2.5). Although applications that are considered in these studies are unrelated to the stress assignment task considered here, the basic ideas underlying the solutions are the same. For example, assignment of stress (insertion of accent marks) can be approached similarly to text segmentation (insertion of word delimiters). As usual, there are some specializations for each application domain, including our own. For example, we model long range dependencies by switching among an array of compression models after each stressed syllable in a word-form[1], and consider that the direction in which text is processed is also relevant for our task.

6.3 Methods

We decompose the problem of stress assignment into three sub-problems. The first task is to determine which vowels in a word-form are stressed. Once the stressed vowels are identified, the type of stress must be determined for stressed vowels *e* and *o* (these two vowels have several possible types of stress). This decomposition of the problem is modeled after stress assignment rules created by Slovenian linguists (Toporišič, 1984). Figure 6.1 depicts the process for a particular word-form. In the final result, all vowels (and reduced vowels) in a given word receive one of the possible labels in Table 6.2.

The three sub-tasks just described are approached in a similar way: We first train one or more compression models from the correct stressed forms of words, which are provided in a training set. To predict stress for a new word-form, all possible solutions

[1]We note that Witten et al. (1999a) also switch between different PPM models in their entity extraction framework, with the distinction that prediction contexts are reset after such transitions in their implementation. Moreover, both the motivation and the desired effect are different, since they are effectively looking to distinguish segments in text, while we are trying to account for long-range dependencies in the models.

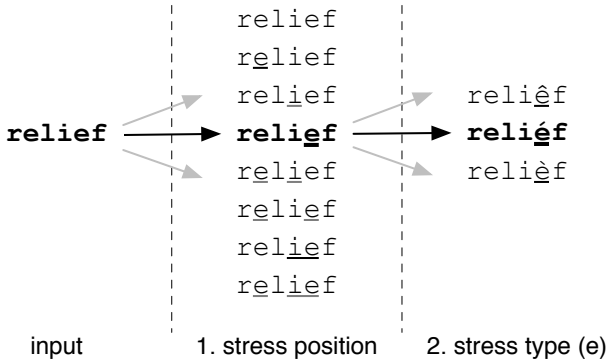


Figure 6.1: Predicting stress position and stress type for the word *relief*.

are generated, and the solution which receives the greatest probability according to the trained compression model(s) is selected. In other words, we select the solution that is compressed the most by the trained model(s). A graphic example for predicting stress position for a particular word-form is given in Figure 6.2. We describe some of the specifics in the following.

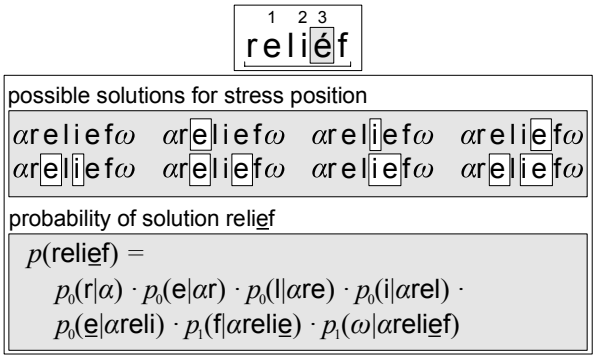


Figure 6.2: Predicting stress position for the word-form *relief* (same meaning in English and Slovenian). All solutions that are considered by the method are listed in the top part of the figure. The evaluation of the correct solution (*reliéf*) is depicted in the bottom part of the figure. $p_0(\cdot)$ denotes the letter probability of the first compression model. After encountering the stressed vowel *é*, we switch to the second model $p_1(\cdot)$.

Long-range dependencies. The probability of a candidate solution $\mathbf{x}$, estimated by a variable order Markov model with limited memory k, is

$$p(\mathbf{x}) = \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_{i-k}^{i-1}) \ , \tag{6.1}$$

where $\mathbf{x}_{i-k}^{i-1}$ is the prediction context for position i, which is at most k letters. A typical value for k might be between 3 and 5, since data sparsity precludes the use of higher-order models. However, most word-forms have only one stressed vowel, and it may be useful to ensure that stress assignments are not considered independently, even if the distance between vowels is larger than the order of the compression model. We therefore consider an additional dependency on the number of stressed vowels $c(\mathbf{x}_1^{i-1})$ in the sequence leading up to the current position:

$$p_s(\mathbf{x}) = \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_{i-k}^{i-1}, c(\mathbf{x}_1^{i-1})) \ . \tag{6.2}$$

The dependency on $c(\mathbf{x}_1^{i-1})$ is realized by switching between different compression models after each stressed vowel. The first model is trained from substrings up to and including the first stressed vowel of each word-form in the training set, and is used to predict letters up to the first stress position in the candidate solution. A second model is trained from letters (and their preceding context) between the first and the second stressed vowel in the training examples, and so on.

Indicative affixes. An inspection of expert-defined stress assignment rules reveals that many common word-form prefixes and suffixes affect the placement of lexical stress (Toporišič, 1984). We append special letters to the beginning and the end of a word-form (denoted as 'α' and 'ω' in Figure 6.2), thus ensuring that the model "knows" whether a particular sequence of letters is a prefix or a suffix of the word-form that is being evaluated. For modeling characteristic suffixes, a reversed presentation of letters might be more suitable, in which the probability of each solution would be computed sequentially from right to left. We experiment with using the reversed representation of word-forms, and with predicting the concatenation of both the original word-form and its reverse (here, different models are trained to predict each direction).

Candidate solutions. When using a single compression model to evaluate a potential stress assignment solution, only combinations that contain between one and three stressed syllables are considered. This prevents solutions which are known not to occur in practice and improves performance. However, this restriction is not needed when using the switching scheme which accounts for long range dependencies between vowels. This is because pathological solutions, such as, for example, a word with no stressed syllables, receive extremely low probability if models are switched after each stressed vowel. In any case, the space of solutions is not too large and permits exhaustive search (the Viterbi algorithm could be used otherwise).

6.4 Experimental Setup

In the following, we compare stress assignment based on data compression models with classification-based stress assignment methods and a machine implementation of stress assignment rules originally created by linguists. We first describe our experimental setup in this section.

6.4.1 The Pronunciation Dictionary

A Slovenian pronunciation dictionary (Šef et al., 2002) forms the basis for our experiments. The pronunciation dictionary used was created semi-automatically from the MULTEXT-East lexicon (Erjavec and Ide, 1998), and reviewed manually by an expert. In total, the dictionary contains over 500,000 word-forms with over 2,000,000 syllables. The word-forms are derived from around 20,000 lemmas, with an average of 30 word-forms per lemma. This represents a small, but relatively frequently used part of the Slovenian vocabulary (there are over 300,000 documented lemmas in the Slovenian language). For each word-form in the dictionary, the lemma, stressed form, and morphological information are given. Statistics on the proportion of stressed vowels and various types of stress in the dictionary are presented in Table 6.3.

	Vowel frequency		**Stress type statistics**		
	occurrences	stressed	narrow	wide	reduced
a	515724	31.95%	-	-	-
e	484614	32.84%	66.62%	33.22%	0.16%
i	525864	25.04%	-	-	-
o	369889	23.98%	82.47%	17.53%	-
u	93606	36.05%	-	-	-
r	28855	51.37%	-	-	-
all vowels	2018552	29.37%	-	-	-

Table 6.3: Frequencies of the stress-bearing vowels in the dictionary. Statistics are also given for the consonant *r*, but only for syllables where *r* it is preceded by a reduced vowel.

6.4.2 Evaluation Procedure and Measures

In practice, our methods would be used on running text to assign stress to words which are *not* included in the dictionary. Although the frequency of such out-of-vocabulary words depends on the text itself, these are rarely used words, so the variance of their frequencies should not vary too much. We therefore conducted standard cross-validation experiments on the dictionary to compare various automatic stress assignment methods. We evaluated all methods with 3-fold cross-validation (as in Marinčič et al., 2009). The

word-forms from the dictionary were divided into three corpora of similar size in such a manner that word-forms with the same lemma were always placed in the same corpus. In this way, the entries in different corpora were not unrealistically similar.

Several reasonable measures can be used to evaluate stress assignment methods. The subtasks of predicting stress position and the type of stress can be evaluated individually or in combination. Prediction accuracy can be measured at the syllable level, for example, the proportion of syllables that are correctly predicted as stressed or unstressed, or at the word level, in which case all constituent syllables must be stressed correctly for a positive outcome. Since humans normally perceive errors in stress at the word level, the most important metric is consolidated word-level performance for the combined task of predicting both stress position and stress type. In the analysis we also report syllable-level results for the individual subtasks of predicting whether a vowel is stressed and determining the type of stress for stressed vowels *e* and *o*.

6.4.3 Expert-defined Rules for Stress Assignment

As a baseline, we report the performance of a rule-based approach derived from stress assignment rules created by Slovenian linguists almost 30 years ago (Toporišič, 1984). These rules occupy around 10 pages in their original source and demand memorization of many exceptions. A slightly modernized machine-readable version of these rules is implemented by Šef and Gams (2004) in the form of 68 IF-THEN rules, of which 19 rules are used for predicting stress position and the remaining 49 rules for predicting stress type. The rules make use of the letters surrounding the stressed vowels, morphological information about the word-form, as well as 230 characteristic word prefixes and suffixes that are known to affect stress. The least specific rule for stress position predicts the most frequent stress position found in other word-forms with the same number of syllables. This rule is used for word-forms that do not match any of the other, more specific stress assignment rules, which is approximately 25% of all word-forms in our dictionary. When comparing expert-defined rules with learning-based methods, the most frequent stress assignment pattern for word-forms with a certain number of syllables was derived from the same training data that was used for training learning-based methods.

6.4.4 Classification-based Methods Used for Comparison

We compare compression-based stress assignment with competing classification-based methods (Marinčič et al., 2009). For classification-based methods, each vowel occurring in the dictionary represents an example which is described with a set of attributes.

The attributes contain morphological information on the word-form in which the vowel appears, the context of letters surrounding the vowel, and the presence of certain characteristic prefixes and suffixes, which are adapted from linguistic rules for stress assignment. The stress assignment problem is again decomposed into separate tasks for predicting stress position and stress type. Six classifiers are trained to predict whether a syllable is stressed (one for each of the vowels *a*, *e*, *i*, *o*, *u* and one for the reduced vowel), and two classifiers are trained to predict the type of stress for stressed vowels *e* and *o*. The classifiers considered were decision rules based on the PART algorithm (Frank and Witten, 1998), C4.5 decision trees (Quinlan, 1993) and boosted decision trees using the AdaBoost boosting algorithm (Freund and Schapire, 1996). The choice of classifiers was influenced by their ability to handle the large amount of training data.

The predictions made on the individual vowels are combined to produce the final stress assignment for the whole word-form using a special probabilistic heuristic. The method essentially combines the prior probability of possible combinations of stressed and unstressed syllables and the confidence of the individual vowel classifications as predicted by the classifiers. We refer the reader to Marinčič et al. (2009) for a more detailed description and analysis of these methods.

6.4.5 Compression Model Parameters

The reported results were obtained using the PPM compression scheme (Cleary and Witten, 1984). A detailed description of the PPM algorithm is given in Section 3.3. In our experiments, we used PPM escape method D (Howard, 1993, see also Section 3.1.2 of this thesis). Our implementation also used the exclusion principle, which is described in Section 3.3.1. The symbol alphabet contained 34 symbols corresponding to distinct letters (including all stressed forms of vowels) found in the pronunciation dictionary.

6.5 Evaluation Results

We first evaluate the effect of compression model switching and other parameter choices for compression-based stress assignment. This is followed by a comparison of compression-based stress assignment and other stress assignment methods.

6.5.1 Compression-based Stress Assignment Variants

Figure 6.3 depicts the performance of different variants and parameterizations of the PPM compression-based stress assignment scheme. The accuracy at word level is plotted for the full task of predicting stress position and stress type. In all cases, the same

variant and parameter settings are used for the subtasks of predicting stress position and type.

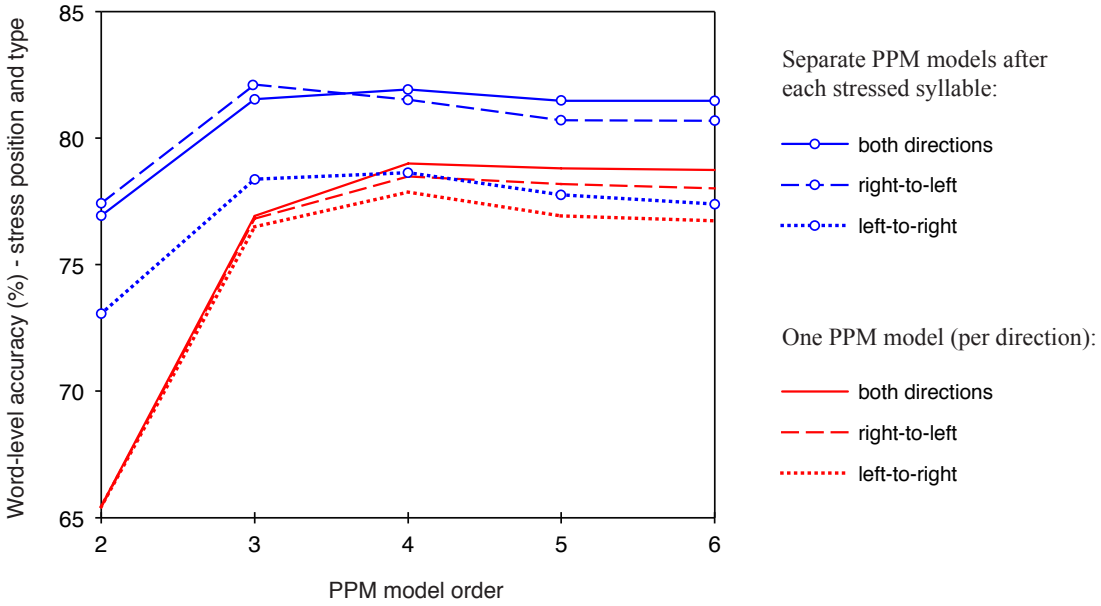


Figure 6.3: Word-level accuracy for the full stress assignment task. The figure depicts the effect of varying the PPM model order parameter, of switching the PPM model after each stressed position, and the direction in which letters that comprise words are presented.

We find that performance is optimal for order-3 or order-4 PPM models, and decreases slightly if the PPM model order parameter is increased beyond 4. Modeling long-range dependencies on the number of preceding stressed syllables by switching between different PPM models improves performance in all tests.

Interesting results are obtained when word-forms are reversed, that is, when letters are sequentially predicted from right to left when scoring solutions. Results are consistently improved compared to the "normal" approach of predicting in the direction from left to right. Using the geometric mean of both scores, which effectively calculates a "joint" probability of seeing both the normal and the reversed sequence, usually performs best. We note that although stress assignment performance improves, the perplexity of predicting from right to left is higher than predicting in the normal direction of the text, that is, words can be compressed better when presented in the normal direction of writing. The better performance using the reversed representation supports the observations of linguists that suffixes of word-forms are an important predictor of lexical stress (Toporišič, 1984).

The effect of switching compression models after each stressed syllable is further analyzed in Figure 6.4. In comparison to using a single compression model, the switching

scheme is particularly effective for longer word-forms, which is as we expect, since the scheme is designed to account for long range dependencies in the number of stressed vowels. Comparing the average number of predicted stressed syllables at different word lengths and the average number of actual stressed syllables found in the original data, we find that the single model approach tends to over-estimate the number of stressed syllables, while predictions based on the switching approach more accurately follow the true distribution in the number of stressed vowels.

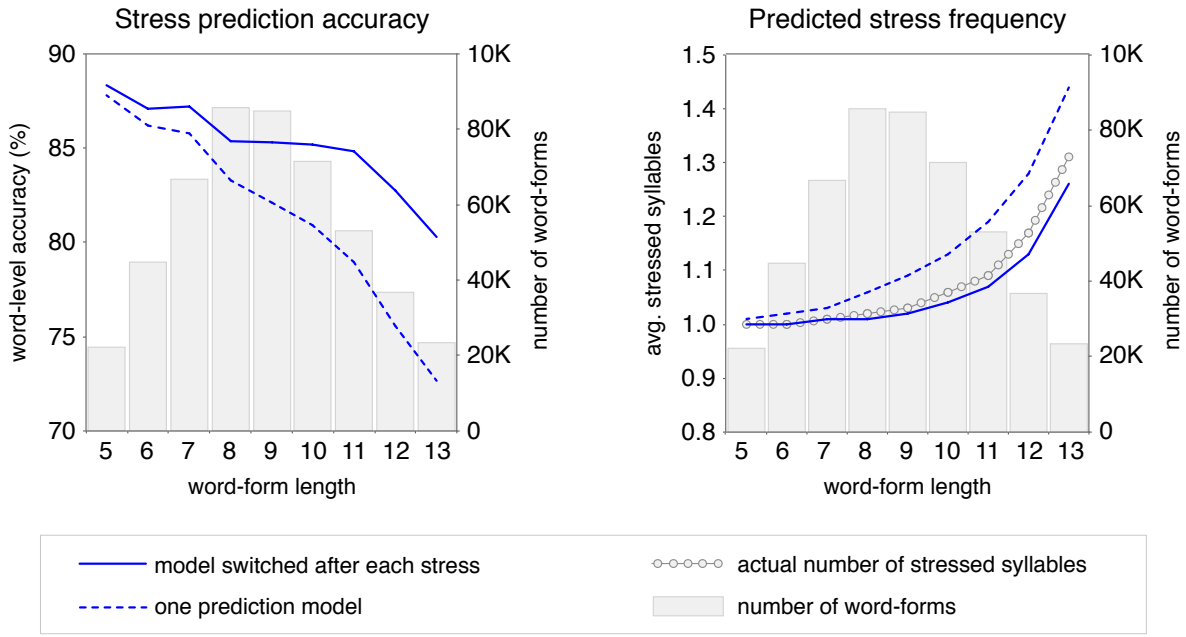


Figure 6.4: Word-level accuracy for the task of predicting stress position at different word-form lengths (left), and the average number of predicted stressed syllables at different word lengths, compared to the average number of actual stressed syllables found in the original data (right). Candidate solutions are evaluated in both directions in all tests. The total number of word-forms of different length is also shown, indicating the overall importance of solid performance at various word lengths (the range covers over 90% of all word-forms).

The running time of 3-fold cross-validation experiments for the PPM-based stress assignment method is reported in Table 6.4. The experiments were run on an Intel Core i7 processor running at 2.66 GHz. The testing time, i.e. the time required for stress prediction on the held-out test data, is considerably greater than the training time. This is because all possible stress assignment solutions for a word-form must be evaluated when predicting stress position. The number of possible solutions is much lower when predicting stress type. Consequently, prediction of stress position accounts for over 90% of the testing time, while prediction of the type of stress for stressed vowels *e* and *o* accounts for less than 10% of the testing time. The throughput of the stress

assignment method is over 1500 word-forms per second, which is more than sufficient for use in speech synthesis systems.

	Training time (seconds)	**Testing time** (seconds)	**Prediction throughput** (word-forms/second)
PPM, order 3	81	323	1660.8
PPM, order 4	93	339	1582.4
PPM, order 5	102	343	1563.9

Table 6.4: Running time of experiments using the PPM compression-based stress assignment method for the full task of predicting both stress position and stress type.

6.5.2 Comparison to Other Methods

The accuracy of compression-based stress assignment, linguistic rules and classification-based methods is compared in Table 6.5. The methods are compared on the tasks of identifying stressed syllables, determining stress type for stressed vowels *e* and *o*, and the full task of predicting both stress position and type for a given word-form. Detailed syllable-level results for predicting stress position for each of the stress-bearing vowels are given in Table 6.6.

	Position (syllable)	**Type (e)** (stressed 'e')	**Type (o)** (stressed 'o')	**Position & Type** (word-form)
Linguistic rules	65.44	63.00	80.80	31.37
Decision rules	88.98	90.73	84.04	72.05
Decision trees	88.83	92.52	85.39	72.21
Boosted DT	**92.99**	93.37	87.17	80.34
PPM, order 3	92.27	**93.54**	**88.29**	81.55
PPM, order 4	92.55	93.14	87.55	**81.93**
PPM, order 5	92.39	92.79	87.04	81.48

Table 6.5: Accuracy (%) achieved by the PPM-based stress assignment method, compared to stress assignment rules developed by linguists and classification-based methods.

For the vowel-level subtasks of predicting stress position and stress type, the accuracy of PPM compression models is similar to boosting, which is the most accurate of the classifiers considered by Marinčič et al. (2009). Stress assignment using compression models and switching produces the best results for the full stress assignment task at word-level, which is also the most important metric in practice.

Although linguistic rules perform quite poorly in the comparison, automatically induced decision trees of similar complexity were found to have comparable performance (Marinčič et al., 2009), supporting the notion that the stress assignment task has no

short and simple solution.

When interpreting the results in Table 6.5, it is important to note that the classification based methods also require morphological information about the word-form. In practical applications such information is not given and must be inferred, and therefore may not be completely reliable, which could negatively affect the overall performance of these methods.

	a	*e*	*i*	*o*	*u*	*r*
Linguistic rules	68.81	61.26	56.58	77.49	72.59	59.64
Decision rules	89.04	87.62	90.56	89.44	87.54	80.53
Decision trees	88.72	86.38	90.62	90.19	89.24	80.56
Boosted DT	93.33	91.28	94.14	93.88	92.42	85.58
PPM-3	93.44	89.40	93.34	92.99	93.10	87.91
PPM-4	93.40	90.34	93.49	93.39	92.13	88.08
PPM-5	93.24	90.13	93.49	92.91	92.57	88.37

Table 6.6: Accuracy (%) achieved by the methods for predicting stress position.

6.6 Discussion

The compression-based approach for lexical stress assignment described in this chapter compares favorably to previous and current stress assignment methods for Slovenian (cf. Šef and Gams, 2004; Tušar et al., 2005; Marinčič et al., 2009). Modeling long-range dependencies by switching between compression models, and considering also reversed representations of word-forms, improves prediction accuracy for this task. No language-specific features are used in our approach, permitting potential generalizations to other languages. Indeed, modeling long-range dependencies in a manner similar to the switching approach described in this chapter (also in Tušar et al., 2005) was later independently found to be successful for stress assignment in German and English by Demberg et al. (2007).

In comparison to competing methods for Slovenian stress assignment, our approach does not require any morphological information, part of speech tags, or other linguistic features which are not readily available in running text. Although this is an advantage for practical applications, the constructive use of additional linguistic features could also be an opportunity for further improving the accuracy of compression-based stress assignment. For example, in Slovenian the imperative forms of verbs often differ to non-imperative forms in the position or the type of stress. Consequently, we find that a modest increase in word-level accuracy (from 81.93% to 82.29%) is achieved simply by training separate models for imperative verbs. However, we have not been success-

ful so far in devising a more principled way to include morphological information in compression-based stress assignment.

A promising direction for future work might be to combine predictions based on compression models with discriminative sequence labeling algorithms, such as conditional maximum entropy models (Mccallum and Freitag, 2000) or conditional random fields (Lafferty et al., 2001), possibly by introducing compression-based predictions as features in the discriminative methods. Such discriminative frameworks are often more flexible in terms of feature engineering, which could yield further improvements, since other methods for Slovenian stress assignment use comparatively rich feature spaces.

Chapter 7

Conclusion

We have described several advances in text mining resulting from the use of adaptive compression models to estimate information-based criteria for classification, active data selection and text annotation. Most of the methods considered are designed for learning from arbitrary discrete sequences, of which text is just one possible example.

Methodological contributions of the thesis primarily include the cross-entropy reduction sampling method, a method capable of extracting a sample of sequences that are representative of a larger dataset. This representative sample is useful in many text mining settings, for example, as a starting point for iterative clustering, or as a basis for supervised training of text classifiers. In the context of using compression models for classification, we suggest that models should be adapted when evaluating the probability of a target sequence. We show that this approach measures an alternative information-based classification criterion linked to the minimum description length principle. The approach significantly improves performance for online filtering of email spam.

The work described in this dissertation contributes to the state-of-the-art in two specific application domains that involve learning from text. First, we propose the use of compression models for spam filtering, and show that compression-based filters are competitive to the best known methods for this task. Our work was among the first studies to demonstrate that character-based modeling is often more successful for spam filtering than word-based tokenization. We demonstrate that compression methods are capable of detecting patterns in text that are otherwise hard to model using word-based tokenization, and are less affected by obfuscation, which is important in the adversarial spam filtering setting. Second, we consider the use of compression models for lexical stress assignment in the context of Slovenian speech synthesis. We develop modeling techniques that are specific for the use of compression models in this application domain, and demonstrate that compression models perform well for this task, while requiring

fewer resources than competing methods.

7.1 Future Work

We have already discussed possible directions for future improvements of methods in specific application domains, namely for instance selection and active learning (Section 4.8), spam filtering (Section 5.7) and lexical stress assignment (Section 6.6). We now take a broader view on prospective future work which transcends the text mining applications that are the topic of this thesis.

Application of the developed methods for other types of sequential data. A natural direction for future work is the application of methods developed in this thesis, most notably representative sampling and, to a lesser extent, also the use of adaptation in classification models, for mining other types of *sequential* data. Such data includes biological sequences like DNA and protein, music, as well as discrete or continuous time series data after suitable transformation into symbolic form[1]. Finding patterns in such data is currently a topic of active research, and compression-based methods are known to be suitable for mining these types of sequences (see, for example, reviews by Keogh et al., 2007; Giancarlo et al., 2009).

Combining Compression Models and Discriminative Learning. The use of compression models in supervised machine learning implies a generative approach. Compression models are well-suited for accurate probabilistic modeling of certain types of data, which may otherwise be difficult to describe using standard machine learning techniques. However, compression models suffer from the standard limitations of generative models in classification settings. One such limitation is the difficulty of extending generative models with additional features, except under unrealistic assumptions of independence. Discriminative methods, on the other hand, often provide for greater flexibility in terms of feature engineering due to the implicit feature selection and feature scaling they perform. A further advantage of many discriminative methods is that such methods include some form of regularization, which allows for tuning the tradeoff between training models that minimize training error and more regular models which are less prone to overfitting in the presence of noise.

One principled way of using generative models in discriminative classification frameworks is via the kernel trick. Fisher kernels (Jaakkola and Haussler, 1999) and TOP kernels (Tsuda et al., 2002) map sequences into a feature space defined by the param-

[1] For example, by using symbolic aggregate approximation (SAX) of Lin et al. (2003).

eters of a generative model. Both of these kernels require probability estimates of the generative models to be differentiable with respect to the model parameters, which appears to be feasible for CTW and DMC compression models (assuming a fixed state machine structure for DMC). Another possible approach would be to use predictions of compression models in different prediction contexts (or FSM states in DMC) as features for discriminative models. In this way, the discriminative classifier could be used to produce a weighting over the implicit feature space used by the compression model, similar to the weighted naive Bayes method (Gärtner and Flach, 2001).

Generalizations of information-based criteria to other domains. Although the CERS algorithm for representative instance selection is designed for sampling sequential data, the basic principle of cross-entropy reduction sampling is more general. The same approach could generally be used to identify representative data of any type if combined with generative models suited to the data of interest. It remains to be seen whether the approach can be generalized to other types of generative models and data in a way that would be both efficient and effective.

Adaptation of sequential prediction models when such models are applied for classification results in selecting the classification hypothesis which yields the shortest description of both the training and the test data combined, i.e. the hypothesis which permits the inference of "better" models form the combined data according to the MDL principle. This approach penalizes redundant evidence pointing towards a particular classification outcome. We would like to further explore this idea for classification of sequences and for other types of data, for which dependencies between features could produce similar intrinsic redundancies which might affect classification based on probabilistic models.

Bibliography

P. H. Algoet and T. M. Cover. A sandwich proof of the Shannon-McMillan-Breiman theorem. *Annals of Probability*, 16:899–909, 1988.

J. Allan, editor. *Topic detection and tracking: event-based information organization.* Kluwer Academic Publishers, 2002.

J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, Lansdowne, Virginia, USA, 1998.

O. Amayri and N. Bouguila. A study of spam filtering using support vector machines. *Artificial Intelligence Review*, 34:73–108, June 2010.

B. Anderson and A. Moore. Active learning for hidden Markov models: objective functions and algorithms. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 9–16, Bonn, Germany, 2005.

I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive Bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, volume 1810 of *Lecture Notes in Computer Science*, pages 9–17, Barcelona, Spain, 2000. Springer.

I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. Technical Report 2004/2, NCSR "Demokritos", Athens, Greece, October 2004.

D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, New Orleans, Louisiana, USA, 2007.

F. Assis. OSBF-Lua – A text classification module for Lua: The importance of the training method. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, 7 pages, Gaithersburg, Maryland, USA, November 2006. NIST.

F. Assis, W. Yerazunis, C. Siefkes, and S. Chhabra. CRM114 versus Mr. X: CRM114 notes for the TREC 2005 Spam Track. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 9 pages, Gaithersburg, Maryland, USA, November 2005. NIST.

F. Assis, W. S. Yerazunis, C. Siefkes, and S. Chhabra. Exponential differential document count: A feature selection factor for improving Bayesian filters accuracy. In *MIT Spam Conference 2006*, 6 pages, Cambridge, Massachusetts, USA, 2006.

M. M. Astrahan. Speech analysis by clustering, or the hyperphome method. Technical Report Artificial Intelligence Project Memorandum AIM-124, Stanford University, California, USA, 1970.

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

A. R. Barron, J. Rissanen, and B. Yu. The Minimum Description Length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

R. A. Baxter and J. J. Oliver. MDL and MML: Similarities and differences. Technical Report TR 207, Dept. of Computer Science, Monash University, Clayton, Victoria, Austalia, 1994.

R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22(1):385–421, 2004.

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3:1183–1208, 2003.

D. Benedetto, E. Caglioti, and Loreto V. Language trees and zipping. *Physical Review Letters*, 88(4):048702, 2002.

C. H. Bennett, P. Gacs, M. Li, P. M. B. Vitanyi, and W. H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, July 1998.

C. H. Bennett, M. Li, and B. Ma. Chain letters and evolutionary histories. *Scientific American*, 288(6):76–81, 2003.

P. Berkhin. *A Survey of Clustering Data Mining Techniques*, pages 25–71. Springer, 2006.

M. Bisani and H. Ney. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *7th International Conference on Spoken Language Processing*, INTERSPEECH 2002, pages 105–108, Denver, Colorado, USA, 2002.

M. Bisani and H. Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, 2008.

A. W. Black, K. Lenzo, and V. Pagel. Issues in building general letter to sound rules. In *Proceedings of the ESCA Workshop on Speech Synthesis*, pages 77–80, Blue Mountains, Australia, 1998.

E. Blanzieri and A. Bryl. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92, March 2008.

D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

A. Bratko and B. Filipič. Spam filtering using character-level Markov models: Experiments for the TREC 2005 spam track. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 11 pages, Gaithersburg, Maryland, USA, November 2005.

A. Bratko and B. Filipič. Exploiting structural information for semi-structured document categorization. *Information Processing and Management*, 42(3):679–694, 2006.

A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7(Dec): 2673–2698, 2006a.

A. Bratko, B. Filipič, and B. Zupan. Towards practical PPM spam filtering: Experiments for the TREC 2006 spam track. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, 8 pages, Gaithersburg, Maryland, USA, November 2006b.

L. Breyer. DBACL at the TREC 2005. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 14 pages, Gaithersburg, Maryland, USA, November 2005.

P. F. Brown, J. Cocke, S. A. Della Pietra, Della, V. J. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June 1990.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, 1992.

M. Burrows and D. Wheeler. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.

X. Carreras and L. Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, Bulgaria, 2001.

O. Catoni. *Statistical learning theory and stochastic optimization*, volume 1851 of *Lecture Notes in Mathematics*. Springer, 2004.

G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, 1966.

S. F. Chen. Conditional and joint models for grapheme-to-phoneme conversion. In *8th European Conference on Speech Communication and Technology*, EUROSPEECH 2003, pages 2033–2036, Geneva, Switzerland, 2003.

X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *Proceedings of the 10th Workshop on Genome Informatics*, pages 51–61, Tokyo, Japan, 1999.

X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50(7): 1545–1551, 2004.

R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3):67–75, 1997.

J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

G. Cormack and T. Lynam. Spam corpus creation for TREC. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS 2005)*, 2 pages, Paolo Alto, California, USA, 2005a.

G. V. Cormack. TREC 2006 spam track overview. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, 11 pages, Gaithersburg, Maryland, USA, November 2006.

G. V. Cormack. University of Waterloo participation in the TREC 2007 spam track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 1 page, Gaithersburg, Maryland, USA, November 2007a.

G. V. Cormack. TREC 2007 spam track overview. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 8 pages, Gaithersburg, Maryland, USA, November 2007b.

G. V. Cormack. Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval*, 1:335–455, April 2008.

G. V. Cormack and A. Bratko. Batch and online spam filter comparison. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, 9 pages, Mountain View, California, USA, 2006.

G. V. Cormack and R. N. S. Horspool. Data compression using dynamic Markov modelling. *The Computer Journal*, 30(6):541–550, 1987.

G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 17 pages, Gaithersburg, Maryland, USA, November 2005b.

G. V. Cormack and T. R. Lynam. Statistical precision of information retrieval evaluation. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 533–540, Seattle, Washington, USA, 2006.

G. V. Cormack and T. R. Lynam. Online supervised spam filter evaluation. *ACM Transactions on Information Systems*, 25(3), July 2007.

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley-Interscience, August 1991.

M. Cuturi and J. P. Vert. The context-tree kernel for strings. *Neural Networks*, 18(8): 1111–1123, 2005.

I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *The Second Conference on Empirical Methods in Natural Language Processing (EMNLP 1997)*, pages 55–63, Providence, Rhode Island, USA, 1997.

M. M. Dalkilic, W. T. Clark, J. C. Costello, and P. Radivojac. Using compression to identify classes of inauthentic texts. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, Bethesda, Maryland, USA, 2006.

Z. Dawy, J. Hagenauer, and A. Hoffmann. Implementing the context tree weighting method for content recognition. In *2004 Data Compression Conference (DCC 2004)*, page 536, Snowbird, Utah, USA, 2004.

G. M. Del Corso, A. Gullí, and F. Romani. Ranking a stream of news. In *WWW '05: Proceedings of the 14th International Conference on World Wide Web*, pages 97–106, Chiba, Japan, 2005.

S. Delany and D. Bridge. Textual case-based reasoning for spam filtering: A comparison of feature-based and feature-free approaches. *Artificial Intelligence Review*, 26(1):75–87, 2006.

S. Deligne, F. Yvon, and F. Bimbot. Variable-length sequence matching for phonetic transcription using joint multigrams. In *4th European Conference on Speech Communication and Technology, EUROSPEECH 1995*, pages 2243–2246, Madrid, Spain, 1995.

V. Demberg, H. Schmid, and G. Möhler. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 96–103, Prague, Czech Republic, 2007. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rdin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39: 1–38, 1977.

B. E. Dom. An information-theoretic external cluster-validity measure. Technical Report RJ 10219, IBM Research, 2001.

P. Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

Q. Dou, S. Bergsma, S. Jiampojamarn, and G. Kondrak. A ranking approach to stress prediction for letter-to-phoneme conversion. In *Proceedings of the 47th Annual Meeting of the Association of Computational Linguistics*, pages 118–126, Suntec, Singapore, 2009.

T. Erjavec and N. Ide. The MULTEXT-East corpus. In *First International Conference on Language Resources and Evaluation (LREC'98)*, pages 28–3, Granada, Spain, 1998.

G. E. Farr and C. S. Wallace. The complexity of strict minimum message length inference. *The Computer Journal*, 45(3):285–292, 2002.

P. Ferragina, R. Giancarlo, V. Greco, G. Manzini, and G. Valiente. Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. *BMC Bioinformatics*, 8(1):252, 2007.

E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

G. Forman and I. Cohen. Learning from little: Comparison of classifiers given little training. In *Proceedings of PKDD-04, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 161–172, Pisa, Italy, 2004.

E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 144–151, Madison, Wisconsin, USA, 1998.

E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Data Compression Conference, DCC 2000*, pages 200–209, Snowbird, Utah, USA, 2000.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, ICML '96, pages 148–1556, Bari, Italy, 1996.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

B. J. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, January 2007.

E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: Providing personalized newsfeeds via analysis of information novelty. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 482–490, New York City, New York, USA, 2004.

L. Galescu and J. Allen. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, 6 pages, Perthshire, Scotland, 2001.

T. Gärtner and P. A. Flach. WBCSVM: Weighted Bayesian classification based on support vector machines. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 154–161, Williamstown, Massachusetts, USA, 2001.

R. Giancarlo, D. Scaturro, and F. Utro. Textual data compression in computational biology: A synopsis. *Bioinformatics*, 25(13):1575–1586, 2009.

T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, (38):293–306, 1985.

J. Goodman and W. Yih. Online discriminative spam filter training. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, 3 pages, Mountain View, California, USA, July 2006.

J. Goodman, D. Heckerman, and R. Rounthwaite. Stopping spam. *Scientific American*, 292(4):42–88, April 2005.

J. Goodman, G. V. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Commununications of the ACM*, 50(2):24–33, 2007.

M. Grochowski and N. Jankowski. Comparison of instance selection algorithms I. algorithms survey. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Soft Computing, ICAISC 2004*, volume 3070 of *Lecture Notes in Computer Science*, pages 598–603, Zakopane, Poland, 2004. Springer.

J. Gros, A. Miheličč, N. Pavešičč, M. Žganec, and S. Gruden. Slovenian text-to-speech synthesis for speech user interfaces. In *Proceedings of the Third World Enformatika Conference*, pages 216–220, Istanbul, Turkey, 2005.

P. Grünwald. A tutorial introduction to the Minimum Description Length principle. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 3–81. MIT Press, 2005.

P. D. Grünwald and P. M. B. Vitányi. Kolmogorov complexity and information theory. with an interpretation in terms of questions and answers. *Journal of Logic, Language and Information*, 12(4):497–529, 2003.

T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, September 2009.

B. Hayes. How many ways can you spell V1@gra? *American Scientist*, 4(95):298–302, 2007.

T. J. Hazen and A. Margolis. Discriminative feature weighting using MCE training for topic identification of spoken audio recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008*, pages 4965–4968, Las Vegas, Nevada, USA, 2008.

J. He, M. Lan, C. Tan, S. Sung, and H. Low. Initialization of cluster refinement algorithms: A review and comparative study. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)*, pages 297–302, Budapest, Hungary, July 2004.

D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

J. M. G. Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 615–620, Madrid, Spain, March 2002.

J. Hovold. Naive Bayes spam filtering using word-position-based attributes. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS 2005)*, 8 pages, Palo Alto, California, USA, July 2005.

P. G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems.* PhD thesis, Brown University, Providence, Rhode Island, 1993.

D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, 1952.

M. Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability.* Texts in theoretical computer science. Springer, 2005.

M. Hutter. Prize for compressing human knowledge, 2006. URL `http://prize.hutter1.net/`.

M. Hutter. Universal learning theory. In C. Sammut and G. Webb, editors, *Encyclopedia of Machine Learning*, pages 1001–1008. Springer, 2011.

K. Y. Itakura and C. L. A. Clarke. Using Dynamic Markov Compression to detect vandalism in the Wikipedia. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 822–823, Boston, Massachusetts, USA, 2009.

T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11 (NIPS 1998)*, pages 487–493, Denver, Colorado, USA, 1999.

H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT Press, 1998a.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142, Chemnitz, Germany, 1998b. Springer.

G. H. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 367–370, Portland, Oregon, USA, 1996.

Z. Jorgensen, Y. Zhou, and M. Inge. A multiple instance learning strategy for combating good word attacks on spam filters. *Journal of Machine Learning Research*, 9(Jun): 1115–1146, 2008.

M. Kato, J. Langeway, Y. Wu, and W. S. Yerazunis. Three non-Bayesian methods of spam filtration: CRM114 at TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 7 pages, Gaithersburg, Maryland, USA, November 2007.

I. Katsavounidis, C. Jay, and Z. Zhang. A new initialization technique for generalized lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, 1994.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An introduction to cluster analysis*. Wiley, 1990.

E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 206–215, Seattle, Washington, USA, 2004.

E. Keogh, S. Lonardi, C. A. Ratanamahatana, L. Wei, S. Lee, and J. Handley. Compression-based data mining of sequential data. *Data Mining and Knowledge Discovery*, 14(1):99–129, 2007.

D. V. Khmelev and W. J. Teahan. A repetition based measure for verification of text collections and for text categorization. pages 104–110, 2003.

A. Kilgarriff. Comparing corpora. *International Journal of Corpus Linguistics*, 6(1): 97–133, 2001.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52:2165–2176, August 2004.

J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, Edmonton, Alberta, Canada, 2002.

J. Kleinberg. Temporal dynamics of on-line information streams. In M. Garofalakis, J. Gehrke, and R. Rastogi, editors, *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2006.

A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems in Information Transmission*, 1(1):1–7, 1965.

P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri. An MDL framework for data clustering. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 323–354. MIT Press, 2005.

L. G. Kraft. A device for quantizing, grouping, and coding amplitude modulated pulses. Master's thesis, Massachusetts Institute of Technology, 1949.

N. Krasnogor and D. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 7(20):1015–1021, 2004.

W. Krauth and M. Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752, 1987.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 282–289, Williamstown, Massachusetts, USA, 2001.

M. Lambers and C. J. Veenman. Forensic authorship attribution using compression distances to prototypes. In *Third International Workshop on Computational Forensics, IWCF 2009*, volume 5718 of *Lecture Notes in Computer Science*, pages 13–24, The Hague, The Netherlands, 2009. Springer.

B. Langmead, C. Trapnell, M. Pop, and S.L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3): R25, 2009.

H. Lee and A. Y. Ng. Spam deobfuscation using a hidden Markov model. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS 2005)*, 8 pages, Paolo Alto, California, USA, 2005.

S. Lee, I. Jeong, and S. Choi. Dynamically weighted hidden Markov model for spam deobfuscation. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, pages 2523–2529, Hyderabad, India, 2007.

L. A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10:206–210, 1974.

D. D. Lewis. Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15, Chemnitz, Germany, 1998. Springer.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, Dublin, Ireland, 1994.

H. Li and R. Durbin. Fast and accurate short read alignment with Burrows's wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009.

M. Li and P. Vitány. Inductive reasoning and Kolmogorov complexity. *Journal of Computer and System Sciences*, 44:343–384, 1992.

M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 3 edition, 2008.

M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *SODA '03: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 863–872, Baltimore, Maryland, USA, 2003.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, December 2004.

R. Liebscher and R. K. Belew. Lexical dynamics and conceptual change: Analyses and implications for information retrieval. *Cognitive Science Online*, 1(2):46–57, 2003.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, San Diego, California, USA, 2003.

H. Liu and H. Motoda. Data reduction via instance selection. In H. Liu and H. Motoda, editors, *Instance Selection and Construction for Data Mining*, pages 3–20. Kluwer Academic Publishers, 2001.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

T. R. Lynam. *Spam Filter Improvement Through Measurement*. PhD thesis, University of Waterloo, 2009.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

M. Mahoney. Large text compression benchmark, 2009. URL `http://www.mattmahoney.net/`.

D. Marinčič, T. Tušar, M. Gams, and T. Šef. Analysis of automatic stress assignment in Slovene. *Informatica, Lith. Acad. Sci.*, 20(1):35–50, 2009.

Y. Marton, N. Wu, and L. Hellerstein. On compression-based text classification. In *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005*, pages 300–314, Santiago de Compostela, Spain, 2005.

A. Mccallum and D. Freitag. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, ICML '00, pages 591–598, Stanford, California, USA, 2000.

A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Fifteenth National Conference on Artificial Intelligence (AAAI-98) Workshop on Learning for Text Categorization*, pages 137–142, Madison, Wisconsin, USA, 1998a.

A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 350–358, Madison, Wisconsin, USA, 1998b.

K. R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. L. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *HLT '02: Proceedings of the Second International Conference on Human Language Technology Research*, pages 280–285, San Diego, California, USA, 2002.

B. McMillan. Two inequalities implied by unique decipherability. *IEEE Transactions on Information Theory*, 2(4):115–116, 1956.

M. Meilă and D. Heckerman. An experimental comparison of several clustering and initialization methods. Technical Report MSR-TR-98-06, Microsoft Research, Redmond, Washington, USA, 1998.

V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive Bayes – which naive Bayes? In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, 9 pages, Mountain View, California, USA, July 2006.

T. A. Meyer. A TREC along the spam track with SpamBayes. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 10 pages, Gaithersburg, Maryland, USA, November 2005.

E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004)*, 8 pages, Mountain View, California, USA, July 2004.

G. W. Milligan. The validation of four ultrametric clustering algorithms. *Pattern Recognition*, 12(2):41–50, 1980.

B. Mirkin. *Clustering For Data Mining: A Data Recovery Approach.* Chapman & Hall/CRC, 2005.

A. Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, COM-38:1917–1921, 1990.

S. Needham and D. Dowe. Message length as an effective Ockham's razor in decision tree induction. In *Proceedings of the 8th International Workshop on AI and Statistics*, pages 253–260, Key West, Florida, USA, 2001.

H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–28, 1994.

H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings fo the 21st International Conference on Machine learning*, ICML '04, pages 79–86, Banff, Alberta, Canada, 2004.

K. Nishida, R. Banno, K. Fujimura, and T. Hoshide. Tweet classification by data compression. In *Proceedings of the 2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the social web, DETECT '11*, pages 29–34, Glasgow, United Kingdom, 2011.

R. Ostrovsky, Y. Rabani, Schulman. L. J., and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 165–176, Washington, District of Columbia, USA, 2006.

R. Pampapathi, B. Mirkin, and M. Levene. A suffix tree approach to anti-spam email filtering. *Machine Learning*, 65(1):309–338, 2006.

L. A. F. Park. Bootstrap confidence intervals for mean average precision. In *Proceedings of the 4th Applied Statistics Education and Research Collaboration (ASEARC) Conference*, pages 51–54, Parramatta, Australia, 2011.

D. Pavelec, L. S. Oliveira, E. Justino, F. D. N. Neto, and L. V. Batista. Author identification using compression models. In *10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 936–940, Barcelona, Spain, 2009.

S. Pearson, R. Kuhn, S. Fincke, and N. Kibre. Automatic methods for lexical stress assignment and syllabification. In *Sixth International Conference on Spoken Language Processing*, ICSLP 2000, pages 423–426, Beijing, China, 2000.

F. Peng, X. Huang, D. Schuurmans, and S. Wang. Text classification in Asian languages without word segmentation. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages*, IRAL 2003, pages 41–48, Sappro, Japan, 2003.

F. Peng, D. Schuurmans, and S. Wang. Augmenting naive Bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345, 2004.

J. M. Peña, J. A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10): 1027–1040, 1999.

R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

D. Radev, J. Otterbacher, A. Winkel, and S. Blair-Goldensohn. Newsinessence: Summarizing online news topics. *Communications of the ACM*, 48(10):95–98, 2005.

T. Reinartz. A unifying view on instance selection. *Data Mining and Knowledge Discovery*, 6(2):191–210, 2002.

I. Rigoutsos and T. Huynh. Chung-kwei: A pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004)*, 8 pages, Mountain View, California, USA, July 2004.

J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, 1984.

J. Rissanen. Complexity of strings in the class of Markov sources. *IEEE Transactions on Information Theory*, 32(4):526–532, 1986.

J. Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., 1989.

J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, 1996.

G. Robinson. A statistical approach to the spam problem. *Linux Journal*, 107:3, March 2003.

M. Rojc and Z. Kačič. Time and space-efficient architecture for a corpus-based text-to-speech synthesis system. *Speech Communication*, 49(3):230–249, 2007.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 441–448, Williamstown, Massachusetts, USA, 2001.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 44–50, Pittsburgh, Pennsylvania, USA, 2001.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6(1):49–73, 2003.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

K. Sayood. *Introduction to Data Compression*. Elsevier, 2005.

SAZU. *Slovene Orthography – 1. Rules (Slovenski pravopis – 1. Pravila)*. Državna založba Slovenije, Ljubljana, 1990.

K. M. Schneider. A comparison of event models for naive Bayes anti-spam e-mail filtering. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '03, Budapest, Hungary, 2003.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning*, ICML '00, pages 839–846, Stanford, California, USA, 2000.

D. Sculley. *Advances in Online Learning-Based Spam Filtering.* PhD thesis, Tufts University, 2008.

D. Sculley and C. E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *2006 Data Compression Conference (DCC 2006)*, pages 332–332, Snowbird, Utah, USA, 2006.

D. Sculley and G. V. Cormack. Filtering email spam in the presence of noisy user feedback. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008)*, 10 pages, Mountain View, California, USA, 2008.

D. Sculley and G. M. Wachman. Relaxed online SVMs for spam filtering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 415–422, Amsterdam, The Netherlands, 2007a.

D. Sculley and G. M. Wachman. Relaxed online SVMs in the TREC spam filtering track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 10 pages, Gaithersburg, Maryland, USA, November 2007b.

D. Sculley, G. M. Wachman, and C. E. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, 10 pages, Gaithersburg, Maryland, USA, November 2006.

L. Seaward and S. Matwin. Intrinsic plagiarism detection using complexity analysis. In *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*, pages 56–61, San Sebastian, Spain, 2009.

R. Segal. IBM SpamGuru on the TREC 2005 spam track. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, 6 pages, Gaithersburg, Maryland, USA, November 2005.

B. Settles. *Active Learning.* Morgan & Claypool, 2012.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory (COLT 1992)*, pages 287–294, Pittsburgh, Pennsylvania, USA, 1992.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.

X. Shen and C. Zhai. Active feedback - UIUC TREC-2003 HARD experiments. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, pages 662–666, Gaithersburg, Maryland, USA, November 2003.

D. Shkarin. PPM: One step to practicality. In *2002 Data Compression Conference (DCC 2002)*, pages 202–212, Snowbird, Utah, USA, 2002.

Y. M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23(3):175–186, 1987.

K. Smets, B. Goethals, and B. Verdonk. Automatic vandalism detection in wikipedia: Towards a machine learning approach. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI08)*, pages 43–48, Chicago, Illinois, USA, 2008.

R. Solomonoff. A preliminary report on a general theory of inductive inference. Technical Report ZTB-138, Zator Company, Cambridge, Massachusetts, USA, 1960.

R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 1(7):1–22 and 224–254, 1964.

R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, 24(4):422–432, 1978.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000) Workshop on Text Mining*, 20 pages, Boston, Massachusetts, USA, 2000.

D. Steinley and M. J. Brusco. Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification*, 24(1):99–121, June 2007.

T. Su and J. G. Dy. In search of deterministic methods for initializing k-means and Gaussian mixture clustering. *Intelligent Data Analysis*, 11(4):319–338, 2007.

R. Swan and J. Allan. Extracting significant time varying features from text. In *CIKM '99: Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 38–45, Kansas City, Missouri, USA, 1999.

T. Šef. *Analiza besedila v postopku sinteze slovenskega govora.* PhD thesis, Faculty of Computer and Information Science, University of Ljubljana, 2001.

T. Šef, M. Škrjanc, and M. Gams. Automatic lexical stress assignment of unknown words for highly inflected Slovenian language. In *Proceedings of the 5th International Conference on Text, Speech and Dialogue*, TSD '02, pages 165–172, London, United Kingdom, 2002.

T. Šef and M. Gams. Data mining for creating accentuation rules. *Applied Artificial Intelligence*, 18(5):395–410, 2004.

P. Taylor. Hidden Markov models for grapheme to phoneme conversion. In *Eurospeech, 9th European Conference on Speech Communication and Technology*, INTERSPEECH 2005, pages 1973–1976, Lisbon, Portugal, 2005.

W. Teahan. Probability estimation for PPM. In *Proceedings of the New Zealand Computer Science Research Students' Conference*, Univ. of Waikato, Hamilton, New Zealand, 1995.

W. J. Teahan. Text classification and segmentation using minimum cross-entropy. In *Proceeding of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 943–961, Paris, France, 2000.

W. J. Teahan and J. G. Cleary. The entropy of English using PPM-based models. In *Proceedings of the 6th Data Compression Conference (DCC 1996)*, pages 53–62, Snowbird, Utah, USA, 1996.

W. J. Teahan and D. J. Harper. Using compression-based language models for text categorization. In W. B. Croft and J. Lafferty, editors, *Language Modeling for Information Retrieval*, pages 141–166. Kluwer Academic Publishers, 2003.

W. J. Teahan, S. Inglis, J. G. Cleary, and G. Holmes. Correcting English text using PPM models. In *Data Compression Conference, DCC 1998*, pages 289–298, Snowbird, Utah, USA, 1998.

W. J. Teahan, R. McNab, Y. Wen, and I. H. Witten. A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, 26(3):375–393, 2000.

Tj. J. Tjalkens, Yu. M. Shtarkov, and F. M. J. Willems. Context tree weighting: Multi-alphabet sources. In *Proc. of the 14th Symp. on Inform. Theory in the Benelux*, pages 128–135, Veldhoven, The Netherlands, 1993.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

J. Toporišič. *Slovene Grammar (Slovenska slovnica)*. Založba Obzorja, Maribor, 1984.

K. Tretyakov. Machine learning techniques in spam filtering. Technical report, Institute of Computer Science, University of Tartu, 2004.

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K. R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, Oct 2002.

T. Tušar, A. Bratko, M. Gams, and T. Šef. Comparison between humans and machines on the task of accentuation of Slovene words. In *Proc. 8th International Multiconference Information Society, IS'2005*, pages 353–356, Ljubljana, Slovenia, 2005.

C. Ungurean, D. Burileanu, and A. Dervis. A statistical approach to lexical stress assignment for TTS synthesis. *International Journal of Speech Technology*, 12(2): 63–73, 2009.

V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., 1995.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

P. Vozila, Y. Adams, J. Lobacheva, and R. Thomas. Grapheme to phoneme conversion and dictionary verification using graphonemes. In *8th European Conference on Speech Communication and Technology*, EUROSPEECH 2003, pages 2469–2472, Geneva, Switzerland, 2003.

C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length.* Springer, 2005.

C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–194, 1968.

C. S. Wallace and D. L. Dowe. Minimum Message Length and Kolmogorov Complexity. *Computer Journal*, 42(4):270–283, 1999.

G. Webster. Improving letter-to-pronunciation accuracy with automatic morphologically-based stress prediction. In *8th International Conference on Spoken Language Processing*, INTERSPEECH 2004, pages 2573–2576, 2004.

P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory*, SWAT 1973, pages 1–11, Iowa City, Iowa, USA, 1973.

T. A. Welch. A technique for high-performance data compression. *The Computer Journal*, 17(6):8–19, 1984.

F. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798, 1998.

F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The Context-tree Weighting Method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004)*, 7 pages, Mountain View, California, USA, 2004.

I. H. Witten. Adaptive text mining: Inferring structure from sequences. *Journal of Discrete Algorithms*, 2(2):137–159, 2004.

I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

I. H. Witten, Z. Bray, M. Mahoui, and B. Teahan. Text mining: A new frontier for lossless compression. In *Data Compression Conference, DCC 1999*, pages 198–207, Snowbird, Utah, USA, 1999a.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999b.

P. Wood, M. Lee, D. Bleaken, M. Nisbet, K. Yuriko, N. Johnston, A. Issa, B. Krishnappa, M. Venugopalan, J. Hurcombe, M. Vicario, and D. Lewis. MessageLabs Intelligence: 2010 annual security report. Technical report, Syantec MessageLabs, 2011.

J. Xu, J. Yao, J. Zheng, Q. Sun, and J. Niu. WIM at TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 11 pages, Gaithersburg, Maryland, USA, November 2007.

Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative sampling for text classification using support vector machines. In *Proceedings of the 25th European Conference on IR Research*, ECIR '03, pages 393–407, Pisa, Italy, 2003.

L. Yang, G. Chang, X. Zhang, and T. Wang. Use of the Burrows-Wheeler similarity distribution to the comparison of the proteins. *Amino Acids*, 39(3):887–898, 2010.

Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49, Berkeley, California, USA, 1999.

Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 28–36, Melbourne, Australia, 1998.

Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 688–693, Edmonton, Alberta, Canada, 2002.

W. S. Yerazunis. Seven hypotheses about spam filtering. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, 10 pages, Gaithersburg, Maryland, USA, November 2006.

Y. Zhou and W. M. Inge. Malware detection using adaptive data compression. In *Proceedings of the 1st ACM Workshop on AISec*, AISec '08, pages 53–60, Alexandria, Virginia, USA, 2008.

Y. Zhou, M. Inge, and M. Kantarcioglu. Compression for anti-adversarial learning. In J. Huang, L. Cao, and J. Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, volume 6635 of *Lecture Notes in Computer Science*, pages 198–209. 2011.

J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

J. Žganec Gros, V. Cvetko Orešnik, and P. Jakopin. SI-PRON pronunciation lexicon: A new language resource for Slovenian. *Informatica (Slovenia)*, 30(4):447–452, 2006.

Dodatek A

Povzetek disertacije

A.1 Uvod

Iskanje zakonitosti v besedilih je področje umetne inteligence, ki se ukvarja z odkrivanjem vzorcev, pridobivanjem strukturiranih podatkov in učenjem statističnih modelov iz besedil v naravnem jeziku. Cilj postopka je, da uporabnikom pomaga pri razumevanju, organizaciji in učinkovitem pregledovanju zbirk besedil, kadar so le-te zaradi obsežnosti neobvladljive, ali da omogoči strojno obdelavo podatkov, izluščenih iz besedila. Področje združuje metode strojnega učenja in statistike, odkrivanja zakonitosti v podatkih, informacijskega poizvedovanja in jezikovnih tehnologij.

Besedilo je v osnovni obliki predstavljeno kot zaporedje znakov, torej zaporedje črk, številk in ločil poljubne dolžine. V takšni obliki besedila niso primerna za obdelavo s standardnimi metodami strojnega učenja, ki so običajno prilagojene za delo s podatki v atributnem opisu, v katerem je vsak primer predstavljen s kombinacijo vrednosti fiksnega števila vnaprej izbranih atributov. Zaradi tega je običajen prvi korak pri odkrivanju zakonitosti v besedilih pretvorba izvornega besedila v atributni zapis z razčlembo besedila na posamezne besede. Rezultat pretvorbe je tipično atributni opis znan pod imenom "vreča besed", v katerem je število pojavitev vsake besede predstavljeno z ločenim atributom. Za tiste probleme na področju iskanja zakonitosti v besedilih, pri katerih je ključen tudi vrstni red besed, se najpogosteje uporabljajo razni markovski modeli nad razčlenjenim zaporedjem besed.

V delu smo ubrali drugačen, manj pogost pristop k modeliranju besedila. Besedilo neposredno modeliramo v osnovni obliki, torej kot zaporedje znakov, brez razčlembe po besedah in brez eksplicitnega atributnega zapisa primerov. Osnovna prednost modeliranja besedila v izvorni obliki je, da na ta način v podatkih ohranimo vse prisotne vzorce, ki bi lahko bili za ciljni problem relevantni, a se pri razčlembi v atributni zapis

na podlagi besed deloma izgubijo. Za modeliranje besedila uporabljamo verjetnostne modele, ki so bili prvotno razviti za kompresijo podatkov. Ti modeli so zasnovani za napovedovanje zaporednih znakov v besedilu, pri čemer je pri modeliranju običajna poenostavitev, da je vsak naslednji znak odvisen le od omejenega števila predhodnih znakov. Primernost kompresijskih algoritmov za modeliranje zaporedij se neposredno odraža v njihovi učinkovitosti pri kompresiji: čim bolj so točne napovedi algoritma, tem boljša je končna kompresija. Kompresijski modeli so verjetnostni modeli nad prostorom diskretnih zaporedij, kot take pa jih je mogoče neposredno uporabiti v številnih problemih s področja odkrivanja zakonitosti v besedilih.

A.1.1 Ožja področja disertacije

Delo se uvršča med številne obstoječe raziskave na področju iskanja zakonitosti v besedilih, ki preučujejo uporabo kompresijskih metod. V disertaciji preučujemo uporabo kompresijskih algoritmov v nekaterih novih aplikacijah, na osnovi kompresijskih modelov pa razvijemo tudi nove metode odkrivanja zakonitosti v besedilih. Posvečamo se različnim problemom s področja, in sicer izbiri informativnih besedil, klasifikaciji besedil in označevanju besedil. Pri izbiri informativnih besedil nadalje preučimo uporabo razvite metode v kontekstu aktivnega učenja in razvrščanja v skupine. Čeprav se v delu omejimo na strojno učenje z besedili, je večina obravnavanih metod v splošnem primerna za strojno učenje s poljubnimi diskretnimi zaporedji.

Izbira primerov (odkrivanje reprezentativnih podmnožic dokumentov). Pri inteligentni izbiri primerov gre za problem optimalne izbire manjše podmnožice podatkov izmed vseh razpoložljivih primerov v podatkovni zbirki, na primer za namen ročnega označevanja ali pregleda s strani uporabnika. V disertaciji se posvečamo problemu izbire podmnožice raznovrstnih in reprezentativnih dokumentov iz večjega korpusa besedil. V ta namen razvijemo hevristiko, ki reprezentativnost podmnožice besedil oceni tako, da na osnovi izbranih besedil zgradi kompresijski model in preveri, kakšna je kompresivnost tako dobljenega modela na referenčnem korpusu besedil, ki naj bi jih podmnožica predstavljala. Uporabnost metode za izbor besedil na osnovi predlagane hevristike smo ovrednotili na problemu aktivnega učenja klasifikatorjev in za inicializacijo algoritma za razvrščanje v skupine k-povprečij. Delovanje metode smo preučili tudi na primeru iskanja pomembnejših zgodb v zgodovinskem arhivu novic.

Klasifikacija (filtriranje nezaželene elektronske pošte). Cilj klasifikacije je na podlagi predhodno označenih primerov besedil zgraditi hipotezo, s katero je možno napovedati razred novih, neoznačenih dokumentov. Za napoved razreda pri klasifikaciji

ocenimo verjetnost vhodnega besedila z različnimi kompresijskimi modeli, naučenimi iz učnih primerov za vsak razred, pri čemer je končna napoved določena glede na model, ki besedilu pripiše največjo verjetnost. V delu smo predlagali in ovrednotili uporabo kompresijskih modelov za filtriranje nezaželene elektronske pošte. Gre za aplikacijo, v kateri je uporaba metod strojnega učenja zelo razširjena, ustaljena in široko raziskana. Z obsežnim ovrednotenjem smo preverili uporabnost kompresijskih modelov za filtriranje nezaželene elektronske pošte v primerjavi z najuspešnejšimi znanimi metodami na tem področju, še posebej pa v primerjavi z dotlej najbolj razširjenim pristopom, ki temelji na pretvorbi sporočila v atributni zapis tipa "vreča besed". V okviru problema filtriranja elektronske pošte smo preučili in ovrednotili tudi pomen sprotnega prilagajanja kompresijskih modelov pri uporabi le-teh za klasifikacijo.

Označevanje besedil (avtomatsko naglaševanje slovenskih besed). Problem označevanja besedil je soroden problemu klasifikacije, le da je v tem primeru potrebno razredne oznake pripisati posameznim delom besedila. Pri tem se tipično pojavljajo odvisnosti med sosednjimi oznakami v besedilu. Za reševanje problemov označevanja lahko uporabimo kompresijski model, naučen na vnaprej označenih besedilih, s katerim ocenimo verjetnosti različnih dovoljenih kombinacij oznak za novo besedilo. Kot rezultat izberemo najbolj verjetno rešitev. V disertaciji predlagamo in ovrednotimo uporabo kompresijskih modelov za problem avtomatskega naglaševanja slovenskih besed, ki je pomemben korak v postopku sinteze slovenskega govora. Preučili smo nekatere specifične tehnike pri uporabi kompresijskih modelov za naglaševanje, ki močno vplivajo na točnost metode, pristop na osnovi kompresijskih modelov pa neposredno primerjamo tudi z drugimi obstoječimi metodami, razvitimi posebej za ta problem.

A.2 Kompresijski modeli

Pri kompresiji navadno obravnavamo informacijski vir, ki oddaja sporočila v obliki zaporedij $\mathbf{x} = \mathbf{x}_1^n = x_1 \dots x_n \in \Sigma^*$ simbolov nad končno abecedo Σ. V primeru, da poznamo statistično porazdelitev nad sporočili, je z uporabo ustreznega kodiranja možna optimalna kompresija sporočil (Cover and Thomas, 1991, glej tudi poglavje 2.2). Statistične lastnosti vira so tipično neznane, tako da je za kompresijo pomembno učenje modelov informacijskih virov na podlagi primerov sporočil oz. podatkov, ki jih vir generira.

Pri kompresiji na podlagi modelov, ki napovedujejo verjetnosti zaporednih simbolov, za bolj verjetne simbole uporabimo krajši zapis kot za simbole, ki so manj verjetni, npr. z uporabo aritmetičnega kodiranja (Rissanen, 1976). Čim bolj je verjetnost različnih simbolov neenakomerna, tem večja je teoretična možnost za kompresijo, de-

jansko dosežena stopnja kompresije pa je neposredno povezana s točnostjo verjetnostnih napovedi modela.

Kontekstni kompresijski modeli delujejo ob predpostavki, da je vsak naslednji simbol x_i v zaporedju $\mathbf{x}$ odvisen izključno od omejenega dela zaporedja $\mathbf{x}_{i-k}^{i-1}$, ki se nahaja tik pred simbolom (t.i. *kontekst* simbola), torej $p(\mathbf{x}) = \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_1^{i-1}) \approx \prod_{i=1}^{\overline{\mathbf{x}}} p(x_i|\mathbf{x}_{i-k}^{i-1})$. Število parametrov kontekstnega modela je $(|\Sigma|-1)\Sigma^k$, torej raste eksponentno z dolžino konteksta k. Napovedi na podlagi daljših kontekstov so potencialno bolj precizne, medtem ko so napovedi na podlagi krajših kontekstov zaradi manjšega števila parametrov bolj zanesljive pri omejeni količini učnih podatkov. Zaradi tega kompresijski modeli sproti prilagajajo dolžino konteksta, torej pri napovedovanju upoštevajo različno število predhodnih znakov k glede na razpoložljivo količino učnih podatkov. Gradnja kompresijskih modelov je tipično postopna, tako da se model za napovedovanje uči sproti s kompresijo vhodnega zaporedja, pri čemer vsaka naslednja napoved $p(x_i|\mathbf{x}_1^{i-1})$ upošteva celoten predhodni del zaporedja $\mathbf{x}_1^{i-1}$ kot učne podatke. Opisan postopek imenujemo *kompresija s prilagajanjem* (angl. adaptive compression).

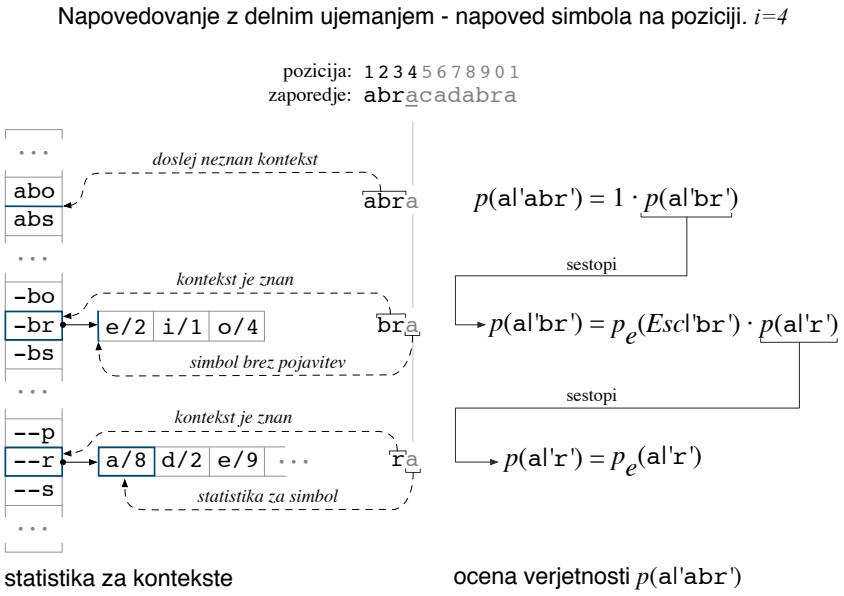
Najbolj znani kontekstni kompresijski modeli so napovedovanje z delnim ujemanjem (PPM; angl. *prediction by partial matching*; Cleary and Witten, 1984), povprečenje kontekstnih dreves (CTW; angl. *context tree weighting*; Willems et al., 1995) in dinamična kompresija Markova (DMC; angl. *dynamic Markov compression*; Cormack and Horspool, 1987). V disertaciji največ uporabljamo prvi metodi, ki jih bomo na kratko opisali v naslednjih podpoglavjih.

A.2.1 Napovedovanje z delnim ujemanjem

Napovedovanje z delnim ujemanjem (PPM; angl. *prediction by partial matching*; Cleary and Witten, 1984) in njegove izpeljanke veljajo za eno najuspešejših kompresijskih metod zadnjih dvajsetih let (Sayood, 2005). Algoritem temelji na glajenju oziroma interpolaciji napovedi na podlagi daljših kontekstov z stabilnejšimi napovedmi na podlagi krajših kontekstov. Podoben postopek glajenja napovedi uporabljajo nekateri jezikovni modeli (npr. Ney et al., 1994), ki pa so v primerjavi s kompresijskimi modeli optimizirani za večje abecede možnih simbolov (za napovedovanje besed namesto znakov).

Predstavljamo si, da PPM model reda k hrani tabelo vseh podnizov dolžine k ali manj, ki se pojavijo v učnih podatkih. Za vsak tak niz beležimo število pojavitev različnih simbolov, ki temu *kontekstu* neposredno sledijo. Ko algoritem napoveduje naslednji simbol x_i, v omenjeni tabeli poišče najdaljši kontekst $\mathbf{x}_{i-l}^{i-1}$, $l \leq k$. Če se v učnih podatkih simbol x_i pojavi v kontekstu $\mathbf{x}_{i-l}^{i-1}$ vsaj enkrat, za napoved verjetnosti uporabimo relativno frekvenco simbola v tem kontekstu. Verjetnosti simbolov, ki se v določenem kontekstu v učnih podatkih pojavijo, je potrebno nekoliko zmanjšati v raz-

merju z verjetnostjo pojavitve kakšnega novega simbola, ki se dotlej v tem kontekstu še ni pojavil. To verjetnost imenujemo *verjetnost sestopa* (angl. escape probability). Verjetnost sestopa porazdelimo med simbole z ničelnim številom pojavitev v kontekstu $\mathbf{x}_{i-l}^{i-1}$, in sicer glede na statistiko krajšega konteksta $\mathbf{x}_{i-l+1}^{i-1}$ dolžine $l-1$. Postopek sestopanja ponavljamo, dokler ne dobimo neničelne verjetnosti za vse možne simbole. Po potrebi uporabimo privzet kontekst reda -1, v katerem so verjetnosti vseh simbolov enake. Postopek ocenjevanja verjetnosti naslednjega znaka na izbranem primeru prikazuje slika A.1.



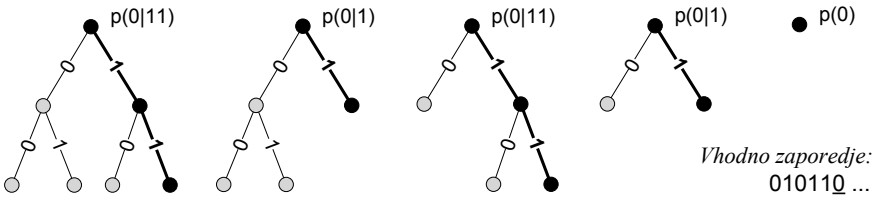
Slika A.1: Shema delovanja algoritma PPM na konkretnem primeru. Postopek ocenjevanja verjetnosti je prikazan na desni in vključuje dva sestopa.

Različne implementacije algoritma se razlikujejo predvsem po modeliranju verjetnosti sestopa. Med bolj popularnimi in uspešnimi metodami je algoritem PPMD (Howard, 1993), ki zmanjša število pojavitev vsakega simbola za $\frac{1}{2}$ in za ravno toliko poveča "število pojavitev" navideznega sestopnega simbola.

A.2.2 Povprečenje kontekstnih dreves

Povprečenje kontekstnih dreves (CTW; angl. *context tree weighting*; Willems et al., 1995) za napovedovanje uporabi bayesovsko povprečje napovedi velikega števila različnih modelov iz razreda *kontekstnih dreves* (angl. context trees, tree source models). Osnovni modeli predstavljajo vsa polna poddrevesa kontekstov z omejeno globino k, kjer je k red CTW modela, kot prikazano na sliki A.2. Vsak osnovni model hrani ločen nabor parametrov za vsak kontekst v drevesu (verjetnosti naslednjih simbolov se za vsak kontekst

v drevesu razlikujejo).



Slika A.2: Vsa kontekstna drevesa, ki jih upošteva algoritem CTW reda 2 nad binarno abecedo. V vsakem modelu je poudarjen kontekst, ki se uporabi za napovedovanje zadnjega simbola zaporedja na sliki.

Verjetnost celotnega zaporedja $\mathbf{x}$ je torej $p(\mathbf{x}) = \sum_{\mathcal{K}} \pi(\mathcal{K}) p(\mathbf{x}|\mathcal{K})$, kjer je $\pi(\mathcal{K})$ apriorna verjetnost določenega kontekstnega drevesa $\mathcal{K}$. Čeprav vsota vključuje napovedi več kot $2^{|\Sigma|^{k-1}}$ različnih modelov, je kompleksnost algoritma linearna zaradi učinkovite rekurzivne dekompozicije izračuna, ki izkorišča dejstvo, da imajo različna kontekstna drevesa med seboj veliko skupnih parametrov. Algoritem omogoča tudi učinkovit izračun verjetnosti zaporednih simbolov $p(x_i|\mathbf{x}_1^{i-1})$, in sicer v času $O(k)$.

Pri uporabi algoritma CTW za modeliranje zaporedij nad abecedami z več kot dvema simboloma je za napovedi naslednjega simbola v posameznih kontekstih osnovnih modelov smiselna uporaba PPM metod sestopanja (Tjalkens et al., 1993), npr. PPM metoda D (Howard, 1993).

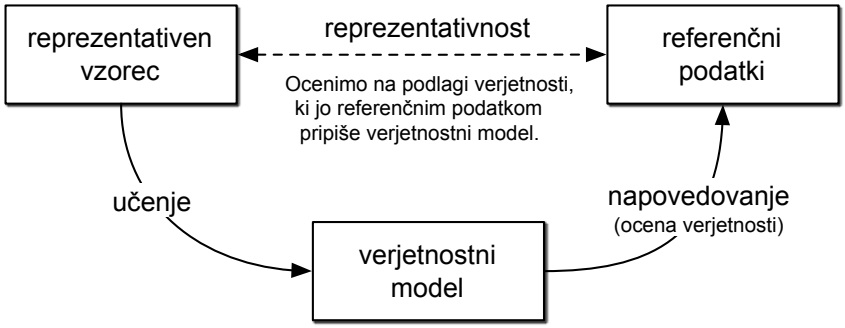
A.3 Iskanje reprezentativnih podmnožic dokumentov

Pogosto se srečujemo s problemi, pri katerih je količina podatkov prevelika, da bi lahko vse razpoložljive podatke ročno pregledali. Nemalokrat je količina podatkov tolikšna, da je otežena tudi strojna obdelava z računsko zahtevnejšimi metodami strojnega učenja. V takšnih situacijah je potrebno število primerov zmanjšati z vzorčenjem.

V disertaciji smo razvili metodo za ciljno iskanje majhnih vzorcev dokumentov, ki čim bolje opisujejo večjo zbirko besedil. Zanima nas, katera besedila iz nepoznanega korpusa besedil naj pregledamo, da bi dobili čim boljšo predstavo o vsebini celotne zbirke ob omejitvi, da lahko izberemo le majhen delež vseh besedil. V primeru, da je del zbirke "poznan", na primer, če znan korpus besedil razširimo z novimi besedili, nas zanima, kateri dokumenti najbolje predstavljajo vsebino, ki je nova ali drugačna v "neznanem" delu podatkov glede na "znano" vsebino.

A.3.1 Metoda na podlagi zmanjševanja križne entropije

Osnovna ideja metode vzorčenja z zmanjševanjem križne entropije (CERS; angl. cross-entropy reduction sampling) je, da reprezentativnost podmnožice primerov (vzorca) merimo z uspešnostjo verjetnostnega modela, zgrajenega na podlagi vzorca, pri napovedovanju referenčnih podatkov (slika A.3). Cilj je poiskati tiste podmnožice primerov, ki porodijo modele, ki preostalim podatkom pripišejo veliko verjetnost. Intuitivno porojeni verjetnostni model predstavlja "znanje", ki ga pridobimo iz vzorca. Čim večja je verjetnost referenčnih podatkov glede na ta model, tem manj ostaja "neznanega" v ostalih podatkih, če poznamo vzorec.



Slika A.3: Reprezentativnost podmnožice oz. vzorca glede na referenčne podatke ocenimo tako, da vzorec uporabimo za učenje verjetnostnega modela in izračunamo verjetnost, ki jo naučeni model pripiše vsem podatkom v referenčni podatkovni zbirki.

Križno entropijo $H(\mathbf{x}, \mathbf{y})$ med dvema zaporedjema definiramo kot povprečno število bitov na simbol, ki jih potrebujemo za kompresijo zaporedja $\mathbf{y}$ pri uporabi kompresijskega modela $\theta_{\mathbf{x}}$, zgrajenega iz $\mathbf{x}$. Če je $p(\mathbf{y}|\mathbf{x})$ verjetnost $\mathbf{y}$ glede na model $\theta_{\mathbf{x}}$, velja $H(\mathbf{x}, \mathbf{y}) = -\frac{1}{|\mathbf{y}|} \log_2 p(\mathbf{y}|\mathbf{x})$. Križna entropija $H(\mathbf{x}, \mathbf{y})$ je majhna v primeru, da je verjetnost $p(\mathbf{y}|\mathbf{x})$ velika, torej da model $\theta_{\mathbf{x}}$, naučen iz zaporedja $\mathbf{x}$, dobro napoveduje referenčno zaporedje $\mathbf{y}$. V disertaciji predstavimo algoritem za izračun križne entropije $H(\mathbf{x}, \mathbf{y})$ med dvema zaporedjema, ki temelji na kompresijskem algoritmu CTW. Algoritem pogojno verjetnost zaporedja $\mathbf{y}$ ob poznavanju zaporedja $\mathbf{x}$ izračuna kot bayesovsko povprečje napovedi kontekstnih dreves $p(\mathbf{y}|\mathbf{x}) = \sum_{\mathcal{K}} p(\mathcal{K}|\mathbf{x}) p(\mathbf{y}|\mathcal{K}, \mathbf{x})$. Aposteriorno verjetnost $p(\mathcal{K}|\mathbf{x})$ kontekstnega drevesa $\mathcal{K}$ po videnju $\mathbf{x}$ pridobimo iz algoritma CTW, medtem ko $p(\mathbf{y}|\mathcal{K}, \mathbf{x})$ predstavlja verjetnost, ki jo zaporedju $\mathbf{y}$ pripiše kontekstno drevo s strukturo $\mathcal{K}$ in parametri, naučenimi iz $\mathbf{x}$.

Iščemo vzorec, torej množico dokumentov S, ki je reprezentativna za večji korpus besedil D. Reprezentativnost vzorca merimo s križno entropijo med množicama $H(\mathrm{S}, \mathrm{D})$, ki je v grobem enaka križni entropiji med spojenimi besedili v S in v D. Algoritem za

iskanje v vsaki iteraciji reprezentativnemu vzorcu doda dokument $\mathbf{s}$, ki najbolj zmanjša križno entropijo med vzorcem in referenčnim korpusom $\Delta H = H(\mathrm{S}, \mathrm{D}) - H(\mathrm{S} \cup \{\mathbf{s}\}, \mathrm{D})$. Z metodo na osnovi algoritma CTW lahko spremembo križne entropije ocenimo v času, ki je odvisen le od dolžine dokumenta $\mathbf{s}$, ne pa tudi od velikosti množic S in D.

A.3.2 Aplikacije

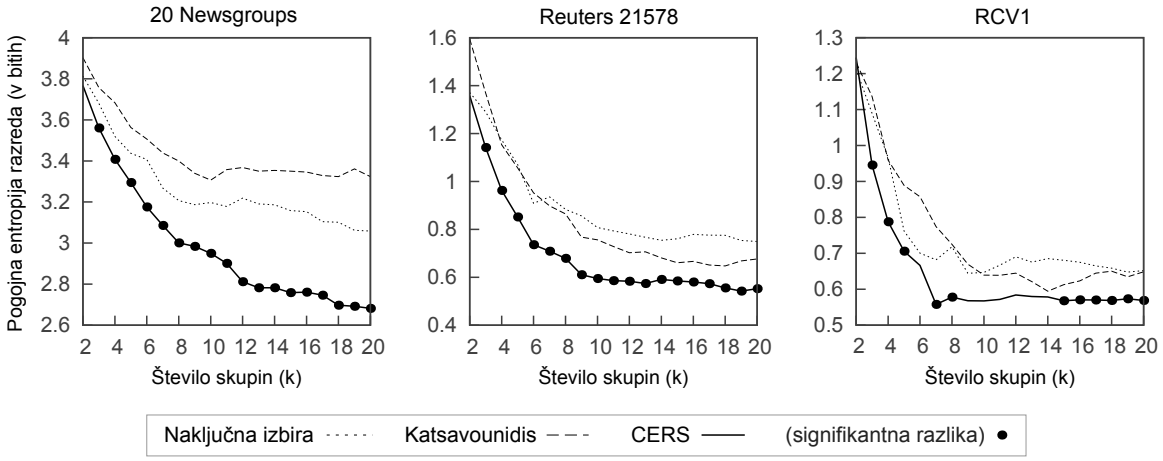
Preizkusili smo različne načine uporabe metode CERS za iskanje reprezentativnih podmnožic dokumentov, in sicer za inicializacijo algoritma za razvrščanje v skupine k-povprečij, za aktivno učenje klasifikatorjev besedil in za iskanje reprezentativnih zgodb v zgodovinskem arhivu novic. Poizkusi temeljijo na javno dostopnih korpusih besedil `20 Newsgroups`, `Reuters21578` in `RCV1`, ki se v literaturi pogosto uporabljajo za vrednotenje algoritmov za odkrivanje zakonitosti v besedilih. V vseh treh korpusih so dokumenti uvrščeni v razrede.

V poizkusih smo za razvrščanje v skupine in za klasifikacijo uporabili standardne algoritme, ki temeljijo na atributnem zapisu dokumentov. Za potrebe teh metod so dokumenti predstavljeni kot normalizirani vektorji tipa "vreča besed" z uteževanjem po metodi TF*IDF (angl. *term frequency invese document frequency*; Salton and Buckley, 1988), kot razdalja med dokumenti pa služi kosinus kota med dvema vektorjema. V vseh poizkusih je red CERS modela 5.

Inicializacija algoritma k-povprečij

Algoritem k-povprečij (angl. k-means) je iterativna metoda za nenadzorovano razvrščanje podatkov v k skupin med seboj podobnih primerov. Algoritem v vsaki iteraciji izračuna centroid (povprečje) vsake od skupin primerov, nato pa vsak primer v zbirki ponovno uvrsti v skupino glede na primeru najbližji centroid. Rezultat metode k-povprečij je močno odvisen od začetne rešitve, ki je tipično izbrana naključno. Cilj metode CERS je poiskati vzorec reprezentativnih in raznolikih dokumentov v danem korpusu besedil. Reprezentativnost si običajno predstavljamo tako, da dokumenti ležijo v središčih večjih skupin podobnih dokumentov. Če so dokumenti raznoliki, pričakujemo tudi, da pripadajo različnim skupinam. Predpostavljamo torej, da so dokumenti, izbrani z metodo CERS, uporabni kot centroidi začetne rešitve pri razvrščanju besedil v skupine z metodo k-povprečij.

Na sliki A.4 je prikazana kakovost končne razvrstitve dokumentov z algoritmom k-povprečij z različnim številom skupin k pri uporabi različnih metod za inicializacijo. Mera kakovosti rešitve je stopnja ujemanja skupin pri razvrčanju z razredi dokumentov, izražena kot pogojna entropija vrednosti razreda dokumenta ob poznavanju uvrstitve v



Slika A.4: Uspešnost razvrščanja v skupine pri $k = 2$ do $k = 20$, ocenjena kot pogojna entropija vrednosti razreda dokumenta ob poznavanju uvrstitve v skupino. Ocena je povprečje desetih poizkusov na različnih naključnih podmnožicah podatkov. Simbol '•' označuje meritve, v katerih je razlika med najboljšo metodo in ostalimi značilna ($p < 0,05$; enosmerni test).

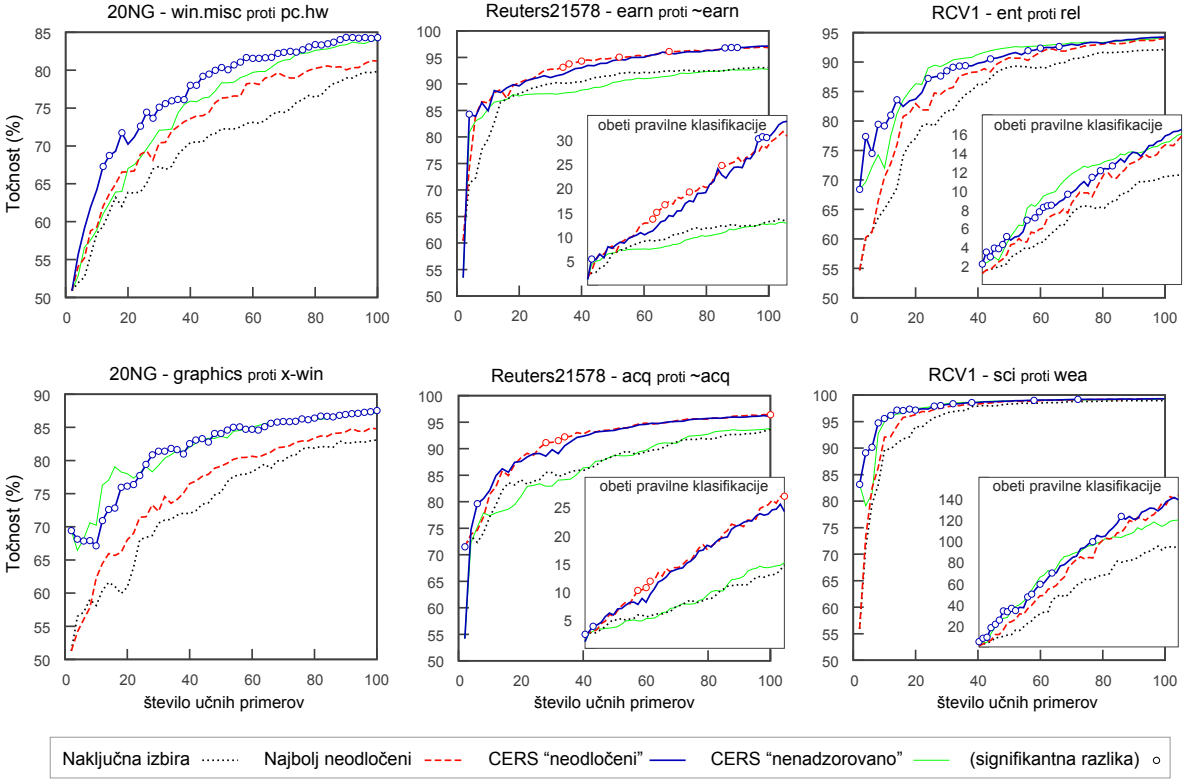
skupino. Inicializacijo z metodo CERS smo primerjali s pogosto uporabljenim postopkom, po katerem za začetne centroide izberemo med seboj oddaljene primere Katsavounidis et al. (1994). Predlagan postopek inicializacije algoritma k-povprečij se je v primerjavi z drugimi metodami izkazal kot uspešen. V disertaciji smo pokazali tudi, da so začetni centroidi, izbrani z metodo CERS, zelo stabilni in v končni razvrstitvi skupin po izteku algoritma k-povprečij ostanejo blizu središčem skupin, ki jih porodijo.

Aktivna izbira učnih primerov za klasifikacijo

Naloga metod za aktivno učenje klasifikatorjev je, da iz zbirke neoznačenih primerov izberejo tiste primere, katerih označba bi predvidoma največ pripomogla k izboljšanju točnosti danega klasifikacijskega modela. S čim manj označenimi učnimi primeri želimo doseči čim boljšo napovedno točnost. Učni primeri se običajno izbirajo zaporedno, pri čemer se predpostavlja, da so oznake predhodno izbranih primerov na voljo pred izbiro naslednjega. Na osnovi metode CERS smo v delu preizkusili nekoliko nekonvencionalen, "nenadzorovan" pristop k aktivnemu učenju besedil, pri katerem za izbiro primerov ne upoštevamo označb predhodno izbranih primerov ali dotedanje klasifikacijske hipoteze. V kolikor so dokumenti, izbrani z metodo CERS, reprezentativni za določene skupine besedil, predpostavljamo, da bomo lahko oznake teh dokumentov uspešno posplošili na podobne dokumente, ne glede na konkreten klasifikacijski problem.

Metode za aktivno učenje pogosto temeljijo na izbiri primerov, za katere je napoved razreda na podlagi do tedaj označenih podatkov najbolj neodločena (npr. Lewis and

Gale, 1994). Pri izbiri najbolj neodločenega primera obstaja nevarnost, da bomo za označevanje izbrali neznačilne primere (angl. outliers), katerih razredne oznake je težko posplošiti na druge primere. Preverili smo hipotezo, da je moč to nevarnost zmanjšati tako, da v vsakem koraku z metodo CERS izberemo najbolj reprezentativen primer izmed majhnega števila najbolj neodločenih primerov[1].



Slika A.5: Povprečna točnost (%) SVM klasifikatorja glede na število učnih primerov za različne metode aktivnega učenja. Rezultat je povprečje 10 poizkusov na naključnih podmnožicah podatkov. Za grafe, kjer je najboljše metode po točnosti težko razločiti, so v manjšem grafu prikazani tudi obeti pravilne klasifikacije, torej $\frac{x}{1-x}$, kjer x predstavlja točnost klasifikatorja. Simbol '$\circ$' označuje meritve, v katerih je razlika med najboljšo metodo in ostalima značilna ($p < 0,05$; enosmerni test, brez upoštevanja nenadzorovane različice metode CERS).

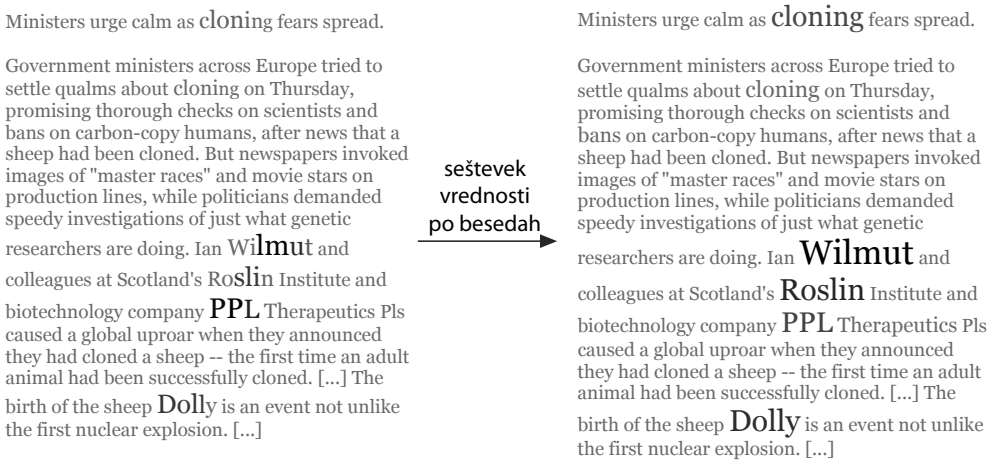
Na sliki A.5 je prikazana točnost klasifikatorja na osnovi metode podpornih vektorjev (SVM) na šestih binarnih klasifikacijskih problemih z različnimi metodami aktivnega učenja. Kombinacija metode CERS z načelom izbire najbolj neodločenih primerov kaže najboljše rezultate in pogosto močno izboljša točnost v primerjavi z izbiro najbolj neodločenega primera. Za nekatere klasifikacijske probleme je zelo uspešna tudi osnovna, nenadzorovana metoda CERS, katere prednost je, da ne potrebuje interakcije z uporabnikom in omogoča paralelno označevanje učnih primerov s strani več označevalcev

[1] V vseh poizkusih smo izbirali med 10 najbolj neodločenimi primeri.

hkrati. Podobne rezultate dobimo pri aktivnem učenju naivnega Bayesovega klasifikatorja.

Identifikacija pomembnih zgodb v zgodovinskem arhivu novic

Metodo CERS smo preizkusili tudi v aplikaciji za iskanje pomembnih zgodb v zgodovinskem arhivu novic. V aplikaciji smo kronološko zaporedje novic s področja znanosti in tehnologije v korpusu RCV1 razdelili na mesečne intervale in iz vsakega intervala z metodo CERS izbrali novice, ki najbolje opisujejo dogodke tistega časa. Poleg časovnice reprezentativnih novic smo iz izbranih novic izluščili tudi ključne besede, ki najbolj prispevajo k oceni pomembnosti novic. Kot primer je na sliki A.6 prikazan izvleček iz članka, ki ga metoda oceni kot najbolj reprezentativnega za februar '97.



Slika A.6: Prispevek posameznih znakov (levo) in skupen prispevek besed (desno) k oceni informativnosti na podlagi znižanja križne entropije za najbolj reprezentativen članek v februarju '97. Nerelevantni deli članka so izpuščeni.

A.4 Filtriranje nezaželene elektronske pošte

Elektronska pošta je v današnjem času nepogrešljiva osnovna komunikacijska storitev, ki jo dnevno uporablja stotine milijonov ljudi. Zaradi pomanjkanja varnostnih mehanizmov v temeljnih protokolih za izmenjavo elektronske pošte se je v zadnjih dveh desetletjih močno razširil problem nezaželene komercialne ali zlonamerne e-pošte (angl. *spam*). V letu 2010 je bilo tovrstnih okoli 89% vseh elektronskih sporočil (Wood et al., 2011). Med ukrepi proti nezaželeni pošti so v široki uporabi različne metode filtriranja na osnovi strojnega učenja (za pregled glej npr. Cormack, 2008; Blanzieri and Bryl, 2008). Med bistvene posebnosti tega problema s stališča strojnega učenja šteje

okoliščina, da filtri delujejo proti aktivnemu nasprotniku, kar zahteva primerno robustne metode.

A.4.1 Metode za filtriranje s kompresijskimi modeli

Kompresijske modele lahko uporabimo za klasifikacijo tako, da za vsak razred zgradimo model, ki povzema značilnosti dotičnega razreda. Pri določanju razreda ocenimo verjetnost vhodnega besedila z vsemi tako zgrajenimi modeli in izberemo razred, katerega model doseže največjo stopnjo kompresije testnega dokumenta. Kompresijski modeli so bili že večkrat uporabljeni za različne probleme razvrščanja besedil (npr. Frank et al., 2000; Marton et al., 2005). Do sedaj se je izkazalo, da so kompresijski modeli še posebej primerni za določanje avtorstva (Teahan, 2000; Lambers and Veenman, 2009), kjer so oblikovne lastnosti besedila pomembnejše od "ključnih besed" v vsebinskem smislu.

V disertaciji smo preučili uporabo kompresijskih modelov za filtriranje nezaželene elektronske pošte. Kompresijski modeli imajo v tej aplikaciji nekaj izrazitih prednosti. Za učenje kompresijskih modelov ni potrebna razčlemba na besede, ki je šibka točka tradicionalnih metod za filtriranje e-pošte (Wittel and Wu, 2004). Kompresijske modele je možno graditi postopno, kar je v tej aplikativni domeni pomembno. Kompresijski modeli omogočajo tudi modeliranje vzorcev, ki niso neposredno povezani z vsebino besedila, kot na primer razni meta-podatki v glavi sporočil, relevantni vzorci ločil ipd. Nekaj takšnih vzorcev je prikazanih na primeru vizualizacije klasifikacije s kompresijskim modelom PPM na sliki A.7.

Pri ocenjevanju pogojne verjetnosti testnega dokumenta s kompresijskimi modeli verjetnost besedila ocenjujemo po znakih. V okviru dela smo preučili alternativen način uporabe kompresijskih modelov za klasifikacijo, pri katerem se (že naučeni) klasifikacijski model sproti prilagaja besedilu, ki ga klasificiramo. S tem dosežemo enak učinek, kot da bi bilo besedilo pred trenutno pozicijo del učnih podatkov. Kompresijske modele s prilagajanjem na ta način uporabimo za izbor klasifikacijske hipoteze, ki omogoča najkrajši opis učnih podatkov *in* testnega primera, kar je skladno z zaporedno različico načela najkrajšega opisa (angl. *prequential MDL*; Rissanen, 1984; Grünwald, 2005). Prilagajanje modela pri klasifikaciji zmanjša vpliv redundantnih podzaporedij, saj vsaka nadaljnja ponovitev določenega podzaporedja vedno manj prispeva h končni izbiri razreda, ker je zaradi prilagajanja vsaka nadaljnja ponovitev bolj verjetna.

A.4.2 Eksperimentalno ovrednotenje

Uporabo kompresijskih modelov za filtriranje elektronske pošte smo eksperimentalno ovrednotili na tekmovanjih sistemov za filtriranje e-pošte pod okriljem konference Text

Slika A.7: Vizualizacija klasifikacije sporočila s kompresijskim modelom PPM-D reda 6. Vsak znak je pobarvan glede na razmerje verjetnosti dotičnega znaka na podlagi dveh kompresijskih modelov – modela nezaželene pošte in modela ostalih sporočil. Vijolična barva kaže na nezaželeno pošto, zelena kaže na legitimno sporočilo.

REtrieval Conference (TREC) v letih od 2005 do 2007 (Cormack and Lynam, 2005b; Cormack, 2006, 2007b). Ovrednotenje metod temelji na sprotnem učenju (angl. online learning), torej se sporočila filtrirajo v kronološkem vrstnem redu, takoj po napovedi pa je metodi na voljo povratna informacija o pravem razredu dokumenta. Glavna mera za ocenjevanje filtrov elektronske pošte je AUC, t.j. površina pod ROC krivuljo (angl. area under the receiver-operating characteristic curve). Dodatni preizkusi, ki smo jih poleg ovrednotenja na konferenci TREC opravili v okviru disertacije, temeljijo na javno dostopnih podatkovnih zbirkah `trec05p`, `trec06p`, in `trec07p`, ki so bile sestavljene za potrebe testiranja na konferenci TREC, in na korpusu elektronske pošte `spamassassin`.

Vpliv prilagajanja modelov pri klasifikaciji

Primerjava točnosti filtriranja z uporabo kompresijskih modelov s prilagajanjem in brez prilagajanja je podana v tabeli A.1. Iz tabele je razvidno, da prilagajanje dosledno izboljša rezultate v vseh poizkusih. V disertaciji smo pokazali, da prilagajanje modelov še posebej pozitivno vpliva na mero AUC. Izboljšanje v primerjavi z napovedovanjem brez prilagajanja je največje na skrajnih koncih ROC krivulje, ki ustrezajo scenarijem, ko je cena napačne razvrstitve zelo različna glede na dejanski razred primera, kar je tipično za filtriranje e-pošte. Na nekaterih primerih pokažemo tudi, da prilagajanje modelov pri klasifikaciji zmanjša vpliv redundantnih vzorcev, poveča pa vpliv raznovrstnih značilk

na končno klasifikacijo dokumenta.

Korpus sporočil	Brez prilagajanja	Prilagajanje (MDL)
`trec05p`	0.0220 (0.0154 - 0.0314)	**0.0142*** (0.0104 - 0.0193)
`trec06p`	0.0408 (0.0299 - 0.0555)	**0.0282*** (0.0205 - 0.0389)
`trec07p`	0.0147 (0.0053 - 0.0411)	**0.0082** (0.0030 - 0.0224)
`spamassassin`	0.3191 (0.1878 - 0.5416)	**0.1794*** (0.1047 - 0.3075)

Tabela A.1: Primerjava PPM kompresijskih modelov reda 6 brez prilagajanja in s prilagajanjem na podatkovni zbirki SpamAssassin in prosto dostopnih korpusih elektronske pošte TREC. Rezultati so podani v odstotkih površine nad ROC krivuljo 1–AUC(%) s 95% intervali zaupanja. Značilne razlike so označene z znakom '$\star$' ($p < 0.01$; enosmerni test).

Primerjava z drugimi metodami

Uporabo kompresijskih modelov za filtriranje elektronske pošte smo prvič preizkusili v okviru tekmovanja na konferenci TREC leta 2005 (Bratko and Filipič, 2005), kjer je sistem na podlagi prilagodljivega kompresijskega modela PPM dosegel povprečno najboljši rezultat med 53 filtri elektronske pošte s strani 17 avtorjev (Cormack and Lynam, 2005b).

Za primerjavo z drugimi dotedanjimi metodami smo izvedli dodatne poizkuse s standardnim prečnim preverjanjem na podatkovnih zbirkah `Ling-Spam`, `PU1` in `PU2`, ki so se dotlej največkrat uporabljale za vrednotenje metod filtriranja e-pošte. Pokazali smo, da kompresijski modeli PPM in DMC dosegajo primerljive ali boljše rezultate v primerjavi s številnimi drugimi metodami, objavljenimi v dotedanjih študijah, ki poročajo rezultate za omenjene podatkovne zbirke (cf. Bratko et al., 2006a; Cormack and Bratko, 2006). V istem delu smo pokazali, da so kompresijski modeli v primerjavi z metodami na osnovi razčlembe po besedah dosti odpornejši na šum, ki v besedilu nastane zaradi prikrivanja vsebine s strani avtorjev nezaželenih sporočil.

V okviru kasnejših tekmovanj TREC v letih 2006 in 2007 je bilo preizkušenih več sistemov na osnovi kompresijskih modelov. Čeprav so kompresijske metode tudi v teh primerjavah zelo konkurenčne, pa so v kasnejših tekmovanjih prišli v ospredje razločevalni modeli, kot sta metoda podpornih vektorjev (SVM) in logistična regresija. Kot zanimivost omenimo, da najuspešnejše kasnejše metode za predstavitev sporočil namesto razčlembe po besedah uporabljajo podnize znakov. Z uporabo kompresijskih modelov za filtriranje e-pošte smo med prvimi prepričljivo pokazali prednosti modeliranja na nivoju znakov v primerjavi z dotlej močno prevladujočo razčlembo po besedah, kar je, sodeč po nekaterih referencah, pripomoglo tudi k poizkusom drugih avtorjev v tej smeri (cf. Sculley et al., 2006; Xu et al., 2007).

V tabeli A.2 je podana primerjava izbranih filtrov nezaželene elektronske pošte na

Klasifikator / Korpus	`trec05p`	`trec06p`	`trec07p`
Najboljši uraden rezultat TREC			
	.019 (.015-.023)[a]	.054 (.034-.085)[b]	.006 (.002-.016)[c]
Popularni odprtokodni filtri			
SpamAssassin	.055 (.042-.072)[⋆]	.131 (.097-.176)[⋆]	.082 (.058-.115)[⋆]
Bogofilter-b	.048 (.038-.062)[⋆,a]	.087 (.066-.114)[⋆,f]	.027 (.017-.043)[⋆,f]
Izbrani klasifikatorji, besede			
Naivni Bayes (MNB)	.351 (.326-.378)[g]	.145 (.116-.181)[g]	N/A
Perceptron	.105 (.091-.120)[g]	.163 (.130-.205)[g]	.042 (.027-.067)[g]
Perceptron s širokim robom	.037 (.030-.046)[g]	.047 (.034-.064)[g]	.019 (.011-.032)[g]
Logistična regresija (LR)	.046 (.041-.052)[g]	.087 (.073-.104)[g]	.011 (.006-.021)[g]
Podporni vektorji (SVM)	.015 (.011-.022)[e]	.034 (.025-.046)[e]	N/A
Izbrani klasifikatorji, 4-grami			
Naivni Bayes (MNB)	.871 (.831-.913)[g]	.477 (.426-.535)[f,g]	.168 (.151-.185)[f]
Perceptron	.060 (.051-.070)[g]	.229 (.186-.282)[g]	.050 (.033-.073)[g]
Perceptron s širokim robom	.022 (.018-.027)[g]	.049 (.034-.070)[g]	.011 (.006-.019)[g]
Logistična regresija (LR)	.013 (.011-.015)[g]	.032 (.025-.041)[f,g]	.006 (.002-.016)[f,g]
Podporni vektorji (SVM)	.008 (.007-.011)[e]	.023 (.017-.032)[e]	N/A
Kompresijski modeli			
DMC	.013 (.010-.018)[d]	.031 (.024-.041)[f]	.008 (.003-.017)[c,f]
PPM	.014 (.010-.019)[⋆]	.028 (.021-.039)[⋆]	.008 (.003-.022)[⋆]
Najboljši filtri TREC '06 in TREC '07			
OSBF-Lua `(oflS1)`	.011 (.008-.014)[⋆]	.054 (.034-.085)[⋆,b,e]	.029 (.015-.058)[⋆,e]
t-ROSVM `(tftS3)`	.010 (.008-.013)[⋆]	.031 (.021-.044)[f]	.009 (.002-.041)[c,f]
w-LR `(wat1)`	.012 (.010-.015)[⋆]	.036 (.027-.049)[f]	.006 (.002-.017)[c,f]
w-LR+DMC `(wat3)`	.010 (.008-.012)[⋆]	.028 (.021-.037)[⋆]	.006 (.002-.016)[c]

Tabela A.2: Primerjava izbranih filtrov nezaželene elektronske pošte na javno dostopnih korpusih TREC glede na površino nad ROC krivuljo 1–AUC(%). Podan je tudi 95% interval zaupanja. Primerjava vključuje kompresijske modele PPM in DMC, dva popularna odprtokodna sistema proti nezaželeni elektronski pošti, najboljše sisteme s tekmovanj TREC v letih 2006 in 2007 in izbrane implementacije različnih algoritmov iz (Sculley, 2008). Sistemi s tekmovanj TREC imajo v oklepajih pripisane oznake iz uradnih rezultatov.

Pripisani znaki označujejo vir rezultatov, pri čemer več označb pomeni, da je rezultat moč najti v več različnih virih:

⋆ – Pričujoča disertacija
a – TREC 2005 uradni rezultati (Cormack and Lynam, 2005b)
b – TREC 2006 uradni rezultati (Cormack, 2006)
c – TREC 2007 uradni rezultati (Cormack, 2007b)
d – Bratko et al. (2006a)
e – Sculley and Wachman (2007a)
f – Sculley and Cormack (2008)
g – Sculley (2008)

podlagi treh javno dostopnih korpusov, objavljenih v okviru tekmovanj iz serije TREC. Primerjava vključuje kompresijske modele PPM in DMC, dva popularna odportokodna programa SpamAssassin in Bogofilter, najuspešnejše filtre s tekmovanj TREC v letih 2006 in 2007 in izbrane implementacije različnih algoritmov strojnega učenja za filtriranje e-pošte, povzete po (Sculley, 2008). V primerjavi kompresijski metodi dosegata boljše rezultate kot odprtokodni programi in druge metode strojnega učenja, ki temeljijo na razčlembi na besede. Točnost metod strojnega učenja na osnovi razločevalnih modelov se poveča, kadar predstavitev sporočil temelji na podnizih znakov[2], vendar so tudi v tem primeru kompresijske metode konkurenčne, razen v primerjavi z metodo podpornih vektorjev, ki pa zaradi računske zahtevnosti ni primerna za sprotno učenje na večjih množicah podatkov (Sculley and Wachman, 2007a). Kompresijska modela PPM in DMC dosegata točnost, ki je primerljiva z najboljšimi sistemi v kasnejših tekmovanjih iz serije TREC. Končno velja poudariti, da je kombinacija logistične regresije in kompresijskega modela DMC ena izmed med najuspešnejših metod (oznaka w-LR+DMC v tabeli A.2).

A.5 Avtomatsko naglaševanje slovenskih besed

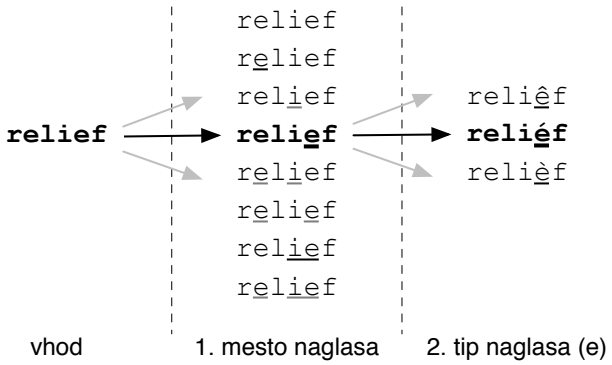
Pretvorba besedila v zapis izgovorjave (fonemski zapis) je ključen korak v postopku avtomatske sinteze govora. V nasprotju z nekaterimi drugimi jeziki je grafemsko-fonemska pretvorba slovenskih besed razmeroma enostavna, če je podana naglašena oblika besede (Šef, 2001; Žganec Gros et al., 2006). Po drugi strani je avtomatsko določanje mesta in tipa naglasa v besedah zahtevnejši problem, saj je naglaševanje različnih besed v slovenščini zelo raznoliko in nepredvidljivo (Toporišič, 1984). Sistemi za sintezo slovenskega govora uporabljajo slovarje izgovorjav, kadar naletijo na besedo, ki je slovar ne pokriva, pa je pred fonemsko pretvorbo običajna uporaba različnih metod za avtomatsko naglaševanje, ki večinoma temeljijo na metodah strojnega učenja (cf. Šef et al., 2002; Šef and Gams, 2004; Tušar et al., 2005; Rojc and Kačič, 2007; Marinčič et al., 2009) ali na uporabi ročno sestavljenih pravil (npr. Šef, 2001; Gros et al., 2005).

A.5.1 Kompresijske metode za naglaševanje

Problem naglaševanja lahko razčlenimo na tri ločene podprobleme, po vzoru ročno sestavljenih pravil za naglaševanje (Toporišič, 1984; Šef, 2001). Najprej določimo kateri samoglasniki v besedi so naglašeni, pri čemer je lahko naglašen kateri izmed samoglasnikov *a*, *e*, *i*, *o*, *u* ali polglasnik pred črko *r*. Nato določimo še tip naglasa, in sicer ločeno

[2]Predstavitev s podnizi štirih zaporednih znakov je najuspešnejša med številnimi predstavitvami v študiji (Sculley et al., 2006).

za naglašene samoglasnike *e* in *o* (široki ali ozki, možen je tudi naglašeni polglasnik namesto samoglasnika *e*). Primer postopka je prikazan na sliki A.8.



Slika A.8: Podproblema napovedovanja mesta naglasa in vrste naglasa za samoglasnik *e* na primeru besede *relief*.

V disertaciji vse tri podprobleme rešujemo z uporabo kompresijskih metod na podoben način. Iz vnaprej označenih učnih primerov zgradimo enega ali več kompresijskih modelov. Pri napovedovanju pregledamo vse možne kombinacije naglasnih mest ali tipov naglasov, verjetnost vsake od možnih rešitev pa ocenimo z naučenimi modeli. Kot rezultat izberemo najverjetnejšo rešitev, torej rešitev, ki daje največjo kompresivnost glede na učne podatke. Uporaba kompresijskih modelov za oceno verjetnosti posamezne rešitve, pridobljene z določeno pretvorbo vhodnega besedila, je poznana tudi v drugih aplikativnih domenah, na primer za popravljanje napak pri optičnem razpoznavanju znakov (Teahan et al., 1998), besedno razčlembo kitajskih besedil (Teahan et al., 2000) in razpoznavanje različnih vrst poimenovalnih izrazov (Witten et al., 1999a).

Verjetnosti rešitev ocenimo s kompresijskimi modeli omejenega reda, kar pomeni, da pri napovedovanju vsake črke v besedi upoštevamo le kratek kontekst nekaj predhodnih črk. Ker ima večina besed le en naglašen zlog, se verjetnost naglašenega zloga po prvem naglasu zmanjša. Tovrstne odvisnosti, ki presegajo dolžino konteksta, modeliramo tako, da pri ocenjevanju verjetnosti možne rešitve po vsakem naglasu uporabimo drug kompresijski model.

Pri določanju mesta naglasa si lahko dostikrat pomagamo z značilnimi priponami in predponami besed (Toporišič, 1984). Zaradi učinkovitejšega modeliranja značilnih pripon in predpon začetek in konec besed označimo s posebnimi simboli, hkrati pa s kompresijskimi modeli napovedujemo spojene nize osnovne in obrnjene besede.

A.5.2 Eksperimentalno ovrednotenje

Uporabo kompresijskih modelov za naglaševanje smo eksperimentalno ovrednotili in primerjali z metodami za naglaševanje na osnovi klasifikacije (Marinčič et al., 2009), ter z ročno sestavljenimi pravili za naglaševanje (Šef, 2001). Metode smo ovrednotili s prečnim preverjanjem na ročno pregledanem slovarju s preko 500.000 naglašenimi besedami (Šef et al., 2002). V poizkusih smo uporabili kompresijski algoritem PPMD.

V tabeli A.3 je podana primerjava točnosti kompresijske metode, metod na osnovi klasifikacije in ročno sestavljenih pravil za naglaševanje. Podani so rezultati za problem določanja naglašenosti zlogov, določanja tipa naglasa za naglašene samoglasnike *e* in *o*, in skupna točnost določanja mesta in vrste naglasa na nivoju celih besed. Kompresijska metoda je še posebej uspešna pri zadnji meri, ki je v praksi tudi najpomembnejša. Pri interpretaciji rezultatov je potrebno upoštevati, da ostale metode zahtevajo tudi oblikoslovne oznake besed in podatke o besedni vrsti, ki jih je v praksi potrebno določiti avtomatsko, kar predstavlja dodatno možnost napak pri teh metodah.

	Naglašenost (zlog)	**Tip nagl. (e)** (naglašen 'e')	**Tip nagl. (o)** (naglašen 'o')	**Mesto in tip** (beseda)
Lingvistična pravila	65.44	63.00	80.80	31.37
Odločitvena pravila	88.98	90.73	84.04	72.05
Odločitvena drevesa	88.83	92.52	85.39	72.21
AdaBoost (drevesa)	**92.99**	93.37	87.17	80.34
PPM, red 3	92.27	**93.54**	**88.29**	81.55
PPM, red 4	92.55	93.14	87.55	**81.93**
PPM, red 5	92.39	92.79	87.04	81.48

Tabela A.3: Točnost (%) kompresijske metode za naglaševanje z uporabo algoritma PPM v primerjavi z ročno sestavljenimi pravili za naglaševanje in metodami na osnovi klasifikacije (Marinčič et al., 2009).

A.6 Zaključek

V disertaciji smo predstavili več prispevkov na področju odkrivanja zakonitosti v besedilih, ki izvirajo iz uporabe kompresijskih modelov za ocenjevanje informacijskih kriterijev za klasifikacijo, aktivno izbiro primerov in označevanje besedil.

A.6.1 Prispevki k znanosti

Prispevke disertacije delimo na metodološke prispevke (prispevki pod točkama I. in II. v nadaljevanju) in aplikativne prispevke v smislu izboljšanja napovednih točnosti dotlej

znanih metod in novih dognanj, ki so relevantna za specifične aplikacije (točki III in IV).

I. Metoda vzorčenja z zmanjševanjem križne entropije. Metoda CERS (angl. cross-entropy reduction sampling), ki temelji na kompresijskih modelih, omogoča identifikacijo podmnožice raznovrstnih dokumentov, ki so reprezentativni za večji korpus besedil (reprezentativen vzorec).

- Uporaba križne entropije kot hevristike pri iskanju reprezentativnih primerov, torej, da reprezentativnost nekega vzorca ocenimo na podlagi verjetnosti, ki jo porojevalni model (angl. generative model), naučen iz vzorca, pripiše referenčni zbirki podatkov.
- Učinkovit algoritem za izračun spremembe križne entropije za zaporedja na osnovi kompresijske metode CTW, ki omogoča iskanje reprezentativnih vzorcev v prostoru diskretnih zaporedij.
- Hevristika za ocenjevanje informativnosti podzaporedij, s katero lahko pri iskanju reprezentativnih dokumentov izluščimo tudi najbolj informativne besede.
- Empirično ovrednotenje, s katerim pokažemo koristnost metode za aktivno izbiro učnih primerov pri klasifikaciji in za inicializacijo algoritma za razvrščanje v skupine k-povprečij.
- Primer uporabe metode za problem avtomatske identifikacije pomembnih zgodb in ključnih besed v zgodovinskem arhivu novic.

II. Analiza prilagajanja verjetnostnih modelov pri klasifikaciji. Na področju klasifikacije s kompresijskimi modeli smo preučili razliko med ocenjevanjem verjetnosti testnega primera s "fiksnimi", vnaprej naučenimi verjetnostnimi modeli (običajen pristop v strojnem učenju) na eni strani in sprotnim prilagajanjem modelov med ocenjevanjem verjetnosti testnega primera, po vzoru uporabe verjetnostnih modelov v kompresiji na drugi strani.

- Utemeljitev prilagajanja modelov pri klasifikaciji z načelom najkrajšega opisa.
- Ovrednotenje vpliva prilagajanja verjetnostnih modelov za filtriranje elektronske pošte, kjer pokažemo, da prilagajanje modelov značilno izboljša rezultate glede na mero AUC.
- Na primerih pokažemo, da prilagajanje verjetnostnih modelov pri klasifikaciji zmanjša vpliv redundantnih vzorcev, ki kažejo na določen razred primera.

III. Prispevki na področju filtriranja nezaželene elektronske pošte.

- Preučili smo uporabo kompresijskih modelov s prilagajanjem za problem filtriranja nezaželene elektronske pošte in pokazali, da kompresijski modeli dosegajo boljše rezultate od dotlej znanih metod za ta problem.
- Pokazali smo, da so v primerjavi s klasičnimi metodami na osnovi razčlembe na besede kompresijski modeli manj občutljivi na prikrivanje vsebine, kar je pogosta taktika avtorjev nezaželenih sporočil.
- Delo, predstavljeno v disertaciji, je prispevalo k splošnejšemu razumevanju, da je modeliranje na osnovi podzaporedij znakov primernejše za problem filtriranja elektronske pošte kot razčlemba na besede.

IV. Prispevki na področju avtomatskega naglaševanja slovenskih besed.

- V disertaciji predlagamo in ovrednotimo metodo na osnovi kompresijskih modelov za problem določanja naglašenih zlogov in vrste naglasa v slovenskih besedah, in pokažemo, da je metoda na osnovi kompresije uspešnejša od drugih metod razvitih posebej za ta problem, pri čemer kompresijska metoda potrebuje tudi manj vhodnih podatkov.

A.6.2 Nadaljnje delo

V prvi vrsti bi veljalo preizkusiti razvito metodo za izbiro reprezentativnih primerov za probleme, kjer se pojavljajo druge vrste diskretnih zaporedij, na primer časovne vrste (z ustrezno diskretizacijo), potencialno pa tudi za biološka zaporedja. V okviru metode iskanja reprezentativnih podmnožic bi bilo zanimivo preučiti možnosti uporabe drugačnih verjetnostnih modelov za oceno zmanjšanja križne entropije, na primer modelov iz razreda Bayesovih mrež. Na ta način bi bilo možno metodo posplošiti še za druge vrste podatkov poleg diskretnih zaporedij.

Na področju klasifikacije in označevanja bi veljalo raziskati možne kombinacije kompresijskih modelov z razločevalnimi modeli (angl. discriminative models) strojnega učenja, na primer z razvojem jedrnih funkcij na osnovi kompresijskih modelov po načelu Fisherjevih jeder (Jaakkola and Haussler, 1999) ali TOP jeder (Tsuda et al., 2002), ali z uporabo delnih napovedi kompresijskih modelov kot atributov, nad katerimi lahko uporabimo razločevalne metode (Gärtner and Flach, 2001).