Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Hrvatin

Modul za analizo plagiatorstva pri kvizih v sistemu Moodle

diplomsko delo

univerzitetni program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Zoran Bosnić

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Spletna platforma Moodle predstavlja popularno orodje za podporo pedagoškemu procesu in omogoča objavo učnih vsebin kot tudi avtomatizirano preverjanje znanja. Ker se preverjanje znanja s pomočjo kvizov lahko izvaja v nenadzorovanem okolju, je v tovrstnih sistemih prisoten problem plagiatorstva.

Kandidat naj v svojem diplomskem delu razvije vtičnik za Moodle, ki izračuna hevristične ocene stopnje plagiatorstva posameznih uporabnikov. Izračune naj opravi tako na nivoju posameznih kvizov, kot tudi agregirano na nivoju celega predmeta. Kandidat naj sam predlaga različne hevristike za analizo suma plagiatorstva, rezultate pa naj v sistemu Moodle tabelarično in grafično prikaže.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Peter Hrvatin, z vpisno številko **63050045**, sem avtor diplomskega dela z naslovom:

Modul za analizo plagiatorstva pri kvizih v sistemu Moodle

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Zoran Bosnić,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 27. avgusta 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LMS	Learning management system	Sistem za upravljanje učenja
DDL	Data definition language	Jezik za definicijo podatkov
DML	Data manipulation language	Jezik za manipulacijo s podatki
CSV	Comma-separated values	Z vejico ločene vrednosti
PCA	Principal component analysis	Metoda poglavitnih komponent
PCoA	Principal coordinates analysis	Metoda poglavitnih koordinat

Povzetek

Moodle je danes ena najpopularnejših spletnih aplikacij na področju e-izobraževanja. Uporablja se za izdelavo in upravljanje s spletnimi učilnicami. Pomemben del spletnih učilnic je preverjanje znanja, ki v sistemu Moodle poteka preko kvizov. Pogost problem pri izvajanju kvizov pa je plagiatorstvo, saj poteka reševanje v nenadzorovanem okolju. Cilj diplomske naloge je bil razviti vtičnik za Moodle, ki bi na podlagi rezultatov respondentov ugotovil, kateri izmed njih so plagiatorji. To se ugotavlja tako, da se na podlagi ujemanja odgovorov med respondenti in časa reševanja kviza izračunajo različne hevristike, ki ocenijo verjetnost plagiatorstva za vsakega respondenta. Vtičnik omogoča analizo na nivoju posameznega kviza in tudi na nivoju več kvizov. Na tak način lahko izvajalec kviza oceni, kdo je najverjetneje plagiator in postane pozoren na veljavnost rezultatov.

Ključne besede: plagiatorstvo, Moodle, kviz, modul.

Abstract

Moodle is today one of the most popular web applications in the field of e-learning. It is used for creation and administration of online classrooms. An important part of online classrooms is assessment of knowledge, which is implemented via Moodle quizzes. A common problem of online quizzes is plagiarism, as they are being solved in an unsupervised environment. The aim of the thesis was to develop a plug-in for Moodle, which can identify plagiarism based on the results of the respondents. Plug-in detects plagiators by matching responses and sumbission times between respondents. It calculates different heuristics that estimate the likelihood of plagiarism for each respondent. The plug-in enables analysis on the level of each quiz and on the level of multiple quizzes. In this way, a tutor can determine who is most likely a plagiarist and be cautious about the validity of achieved results.

Keywords: plagiarism, Moodle, quiz, module.

Poglavje 1

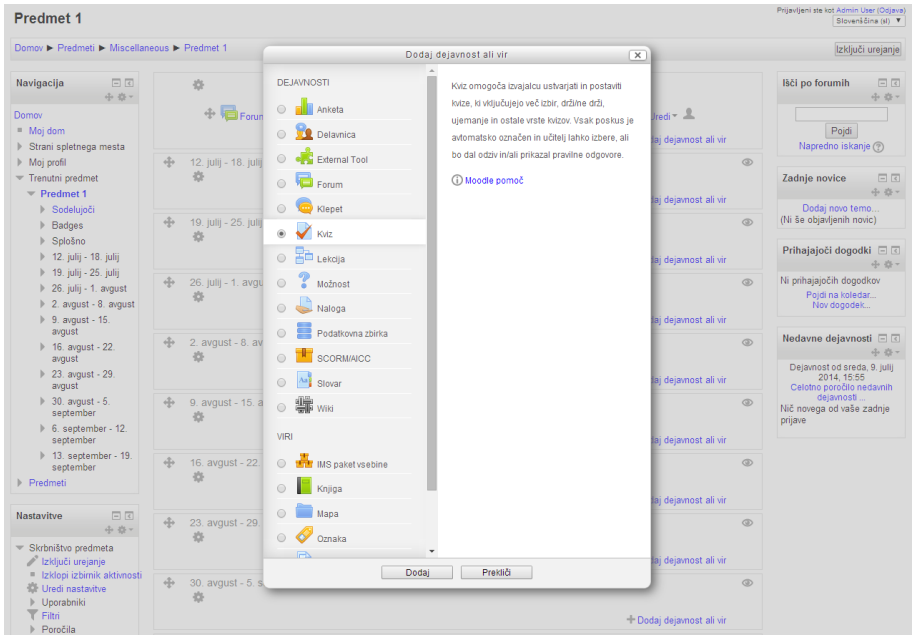
Uvod

1.1 E-izobraževanje in Moodle

E-izobraževanje je način prenosa znanja, ki je v današnjem času čedalje bolj popularen. Pri e-izobraževanju poteka učenje preko osebnega računalnika, pri tem pa ni potrebna neposredna prisotnost učitelja. S takim načinom izobraževanja se znižajo stroški učenja, izboljša se dostopnost, učenci niso časovno omejeni in tudi ocenjevanje je ponavadi dosti hitrejše, saj je vsaj delno avtomatizirano [1].

Velik del e-izobraževanja predstavljajo sistemi za upravljanje z učenjem (Learning management systems), ki omogočajo administracijo in izvajanje e-učenja. Med njih spada tudi Moodle, ki velja danes za eno najpopularnejših orodij za postavitev in upravljanje s spletnimi učilnicami. Omogoča ustvarjanje in urejanje spletnih tečajev, objavo gradiv, izvajanje preizkusov znanja, ocenjevanje udeležencev, ... Trenutno ga uporablja več kot 70 milijonov uporabnikov, z njim pa upravlja več kot 1 milijon učiteljev iz 235 držav. Kot pomoč pri izobraževanju ga pogosto uporabljajo šole in izobraževalna podjetja. Zato je danes zelo pomemben del e-izobraževanja.

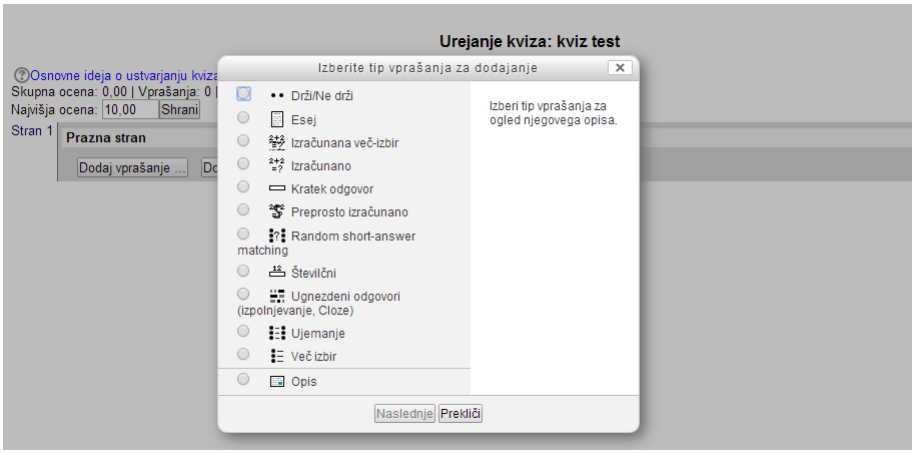
Ena ključnih komponent kakršnegakoli izobraževanja je preverjanje znanja. Pri e-izobraževanju se to običajno izvaja s pomočjo kvizov. Kvizi so lahko for-



Slika 1.1: Izdelava kviza v sistemu Moodle

malni ali neformalni. Neformalni služijo kot pomoč učencu, saj s tem dobi boljšo predstavo o svojem trenutnem znanju. Pri tem je pomembno, da je ocenjevanje popolnoma avtomatizirano in je respondent sproti obveščen o pravilnosti odgovorov. Kvizi pa se lahko izvajajo tudi formalno, kjer se njihova ocena upošteva pri končni oceni učenca.

Tudi Moodle omogoča izdelavo kvizov, ki so namenjeni preverjanju znanja znotraj posameznega predmeta. Upravljalec s predmetom lahko v urejevalnem načinu ustvari in objavi kviz, kar je prikazano na sliki 1.1. V kviz lahko vstavlja različne tipe vprašanj, ki so prikazani na sliki 1.2. Ta kviz lahko potem rešujejo vsi udeleženci tega predmeta, na koncu pa se odgovori ocenijo (avtomatsko ali ročno).



Slika 1.2: Vstavljanje vprašanj v kviz v sistemu Moodle

1.2 Plagiatorstvo

E-izobraževanje ima tudi slabe plati. Glavna slabost je v tem, da ni osebnega stika med učiteljem in učencem. To predstavlja težavo pri ocenjevanju, saj učitelj ne more neposredno oceniti težav učenca in tako pomagati pri razumevanju. Pri preverjanju znanja pa nastopi zaradi odsotnosti učitelja še en problem, in sicer problem plagiatorstva. Respondenti, ki rešujejo kviz, namreč nimajo nadzora, saj ga ponavadi rešujejo doma oz. v okolju, ki se ga ne da nadzorovati. To pomeni, da lahko odgovore dobijo od koga drugega in jih prepišejo, ali pa rešujejo v skupinah.

Aplikacij za avtomatsko ugotavljanje plagiatorstva je danes na voljo kar nekaj, vendar pa jih večina temelji na ugotavljanju podobnosti besedil. Tudi Moodle ima na voljo kar nekaj vtičnikov za zaznavanje podobnosti besedil (običajno med besedilnimi datotekami, ki jih naložijo respondenti). Tak način zaznavanja plagiatorstva pa ne pride v poštev pri samem reševanju kviza. Večina vprašanj v kvizu namreč ni besedilnih, ampak gre ponavadi za izbirne oziroma številske tipe vprašanj.

1.3 Namen diplomske naloge

Cilj diplomske naloge je bil razviti vtičnik za sistem Moodle, ki bi na podlagi analize odgovorov vseh respondentov kviza ugotovil, kateri izmed njih so prepisovali oz. reševali na nedovoljen način. Vtičnik deluje tako, da analizira odgovore posameznih respondentov in na podlagi ujemanja ugotovi, kateri respondenti so najverjetneje plagiatorji.

1.4 Zgradba diplomske naloge

Diplomsko delo je razdeljeno na 2 dela. V prvem delu je opisan postopek razvoja vtičnikov za sistem Moodle. V tem delu so opisani: struktura vtičnikov, tehnologije, ki se uporabljajo za njihovo izdelavo, in knjižnice, ki se pogosto uporabljajo pri samem razvoju.

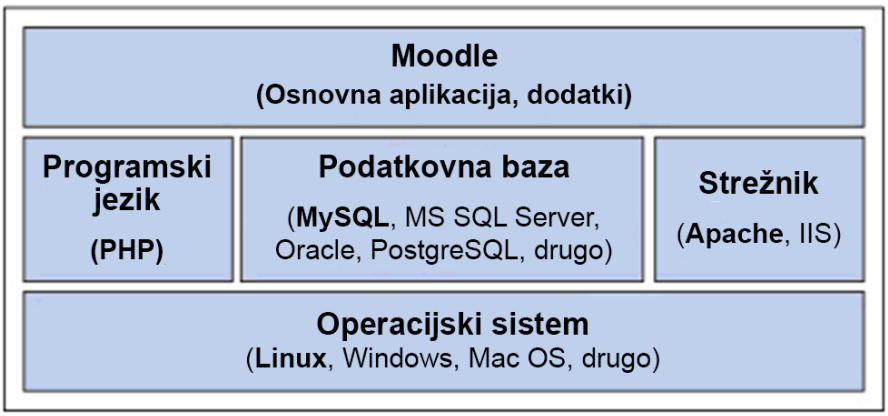
Drugi del je namenjen predstavitvi vtičnika za plagiatorstvo. Razvit vtičnik je sestavljen iz dveh delov. Prvi del vtičnika skrbi za analizo plagiatorstva na nivoju posameznega kviza, drugi del pa za analizo plagiatorstva na nivoju posameznega predmeta.

Poglavje 2

Opis razvoja vtičnikov za Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) je odprtokodna aplikacija, kar pomeni, da lahko kdorkoli izdela njene razširitve oziroma izboljšave. Kot pove prva beseda v imenu, je osnovno načelo modularnost [2]. To najlažje dosežemo z vtičniki, ki jih uporabnik sistema dodaja in odstranjuje po potrebi. Na ta način je omogočeno razvijalcem, da neodvisno implementirajo posamezne izboljšave, uporabniki pa uporabijo samo tiste, ki jih res potrebujejo.

Moodle je napisan v programskem jeziku PHP, podatke pa hrani v relacijski bazi. Zato je za razvoj potrebno poznavanje teh dveh tehnologij. Arhitektura Moodla je prikazana na sliki 2.1. Razvoj vtičnika lahko razdelimo na več korakov in je prikazan na sliki 2.2. Prvi korak je pridobitev predloge vtičnika, ki je dostopna na spletni strani Moodla in vsebuje vse potrebne datoteke za vtičnik. Temu sledi določitev lokacije našega vtičnika v datotečnem sistemu, kar je odvisno od področja, na katerem razvijamo vtičnik (v našem primeru sta bila to področja poročila kvizov in poročila predmetov). Nato je potrebno v vseh datotekah spremeniti privzeto ime (*newmodule*) v ime našega vtičnika. S tem je pripravljena podlaga za začetek izdelave.

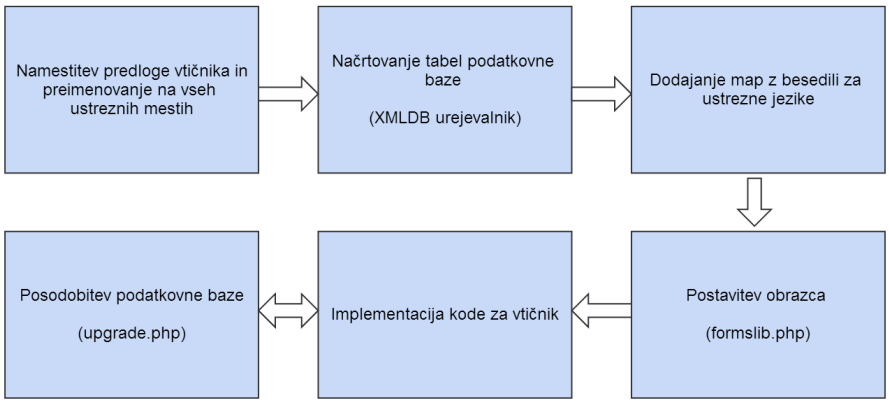


Slika 2.1: Arhitektura sistema Moodle

Drugi korak je načrtovanje tabel podatkovne baze, ki jih bo naš vtičnik uporabljal. To storimo tako, da definiramo vse tabele v datoteki *install.xml*. Lahko pa tudi uporabimo XMLDB urejevalnik, ki se nahaja v vmesniku Moodle.

Tretji korak je dodajanje ustreznih jezikov našemu vtičniku. Vso besedilo, ki bo v našem vtičniku, se definira v jezikovnih datotekah znotraj mape *lang*. Za vsak jezik, ki želimo da je podprt v našem vtičniku, ustvarimo novo mapo, ki vsebuje datoteko z ustreznimi prevodi vseh besedil.

Četrti korak obsega postavitev obrazca. Obrazce uporabljamo za kakršnokoli spreminjanje vsebine našega vtičnika (spreminjanje nastavitev, vnos besedila, ...). Pri tem lahko spremembe shranimo v podatkovno bazo ali pa jih samo uporabimo za drugačen prikaz podatkov. V našem vtičniku se obrazec uporabi za spreminjanje nastavitev, ki se uporabljajo pri izračunu določenih ocen plagiatorstva. Če vtičnik ne omogoča take funkcionalnosti, potem ta korak ni potreben. Obrazec se definira v novi datoteki (običajno $ime\_vticnika\_form.php$), kjer je definiran nov razred, ki je razširitev obstoječega razreda $moodleform$. V tem razredu se mora nahajati funkcija $definition()$, znotraj katere definiramo parametre obrazca in dodajamo poljubne elemente (nastavitve tipa



Slika 2.2: Postopek razvoja vtičnika za sistem Moodle

potrditveno polje, izbirni gumb, ...).

Zadnji korak razvoja je dodajanje kode za delovanje novega vtičnika. Običajno definiramo za to nov razred, ki je razširitev že obstoječega razreda s področja, na katerem ustvarjamo vtičnik. Na primer, če razvijamo vtičnik na področju poročil kvizov, lahko uporabimo že obstoječ razred *quiz_default_report*. Na ta način imamo na voljo vrsto uporabnih funkcij, ki služijo za pridobitev rezultatov kviza, vprašanj v kvizu, vseh respondentov, ki so odgovarjali na kviz, ...

Dodaten korak, h kateremu se vračamo vsakič, ko razširjamo funkcionalnosti našega vtičnika, je posodabljanje tabel podatkovne baze, ki jih potrebujemo za novo funkcionalnost. To je koristno zlasti pri vtičnikih, ki so že v uporabi, saj nam ni potrebno brisati stare verzije vtičnika in ga nato na novo namestiti. Moodle namreč avtomatsko preveri, katera verzija vtičnika je nameščena in izvede potrebne popravke. Za pravilno delovanje je potrebno zabeležiti vsako novo verzijo vtičnika znotraj datoteke *version.php*. Iz te datoteke se potem kliče ustrezne funkcije v datoteki *db/upgrade.php*, ki poskrbijo za spremembe v podatkovni bazi s pomočjo objekta razreda *moodle_database*, ki vsebuje funkcije DDL.

2.1 Zgradba vtičnika

Vtičnik, ki ga želimo izdelati, vedno pripada določeni skupini razširitev (npr. poročilo predmeta, tip vprašanja za kviz, ...), kar določi tudi, kje se bo nahajal v datotečni strukturi. Vsak vtičnik mora vsebovati določene datoteke, ki so nujno potrebne za njegovo delovanje:

- **index.php** - Vsebuje osnovni razred za implementacijo vtičnika. Običajno je ta razred razširitev že obstoječega razreda (odvisno od tega, kateri kategoriji pripada vtičnik). Na ta način so nam na voljo funkcije, ki se pogosto uporabljajo na področju, kjer razvijamo vtičnik.

- **version.php** - Datoteka, ki vsebuje podatke o vseh verzijah vtičnika. S pomočjo te datoteke povemo sistemu Moodle, kaj vse mora izvesti ob posamezni posodobitvi vtičnika.

- **ime_vtičnika_form.php** - Datoteka, v kateri je definiran obrazec, ki ga uporablja naš vtičnik. V njem je definiran nov razred, ki je razširitev obstoječega razreda *moodleform*. V tem razredu je že definirana večina funkcij, potrebnih za upravljanje z obrazci. Če vtičnik nima funkcionalnosti, ki bi zahtevala obrazec, ta datoteka ni potrebna.

- **lang/en/ime_vtičnika.php** - Datoteka, ki vsebuje vsa besedila, ki se pojavljajo v vtičniku. Posamezne nize nato prikličemo znotraj vtičnika z ukazom *get_string*(). Če želimo imeti vtičnik, preveden tudi v kakšen drug jezik, se tukaj lahko doda poljubno število jezikovnih datotek.

- **db/install.xml** - Vsebuje nove tabele podatkovne baze, ki bodo uporabljene v vtičniku. Tabele se definirajo v obliki XML. Ob naložitvi vtičnika se potem ta tabela avtomatsko ustvari s pomočjo knjižnice Moodle DDL.

- **db/upgrade.php** - Vsebuje vse naknadne spremembe podatkovne baze, ki se morajo izvesti ob posodobitvah vtičnika.

2.2 Podatkovna baza

Moodle uporablja za svoje delovanje relacijsko podatkovno bazo. Trenutno podpira podatkovne baze MySQL, PostgreSQL, MS SQL 2005 in Oracle [3].

Celotno upravljanje s podatkovno bazo poteka preko več plasti, ki predstavljo abstrakcijo podatkovne baze. To omogoča fleksibilnost sistema, kar pomeni, da se ga lahko prenese na različne strežnike, ki podpirajo različne podatkovne baze brez kakršnih koli sprememb oz. popravkov. Slika 2.3 prikazuje interakcijo sistema Moodle z relacijsko podatkovno bazo.

- Opisne datoteke XMLDB služijo za ustvarjanje tabel v podatkovni bazi. S pomočjo teh datotek se definira vse nove tabele, ki jih naš vtičnik potrebuje. Te se potem ustvarijo avtomatsko ob namestitvi vtičnika. Izdelava te datoteke je ponavadi prvi korak pri izdelavi vtičnika. Pri tem je priporočljiva uporaba urejevalnika XML, ki je bil izdelan posebej za Moodle in precej olajša delo razvijalcu, hkrati pa zagotavlja pravilnost sintakse datoteke XML. Urejevalnik je dostopen znotraj administratorskega vmesnika za Moodle pod navigacijo Administracija strani->Razvoj.

- Stavki SQL v Moodle skrbijo za dodajanje, popravljanje in brisanje podatkov v podatkovni bazi. Ti stavki niso odvisni od tipa podatkovne baze.

- Knjižnica DDL v Moodle (Data Definition Language) vsebuje funkcije, ki poskrbijo za abstrakcijo upravljanja s strukturo podatkovne baze. Te funkcije se uporabljajo izključno pri procesu namestitve in posodabljanja vtičnika. Pri razvoju vtičnika se uporabljajo znotraj datoteke *upgrade.php*, kjer se definirajo naknadne spremembe na tabelah podatkovne baze, ki jih naš vtičnik uporablja.

  Vse funkcije v tej knjižnici so javne in se kličejo preko globalnega objekta $\$DB$, ki pripada razredu *moodle_database*. Obstajajo 3 skupine teh

funkcij, pri čemer vse upravljajo z objekti XMLDB. Prva skupina funkcij služi za ustvarjanje, spreminjanje in brisanje tabel. Druga skupina se uporablja za urejanje polj v posameznih tabelah. Tretji tip pa za dodajanje, spreminjanje in odstranjevanje indeksov in ključev iz obstoječih tabel. Koda prikazuje primer uporabe vseh treh tipov funkcij.

```
function xmldb_verzijaXX_upgrade {
    global $DB;

    // Iniciacija upravljalca ddl
    $dbman = $DB->get_manager();

    // Izdelava nove tabele
    $dbman->create_table(
        $table, $continue=true, $feedback=true
    );

    // Dodajanje polja tabeli
    $dbman->add_field(
        $table, $field, $continue=true,
        $feedback=true
    );

    // Odstranjevanje indeksa iz tabele
    $dbman->drop_index(
        $table, $index, $continue=true,
        $feedback=true
    );
}
```

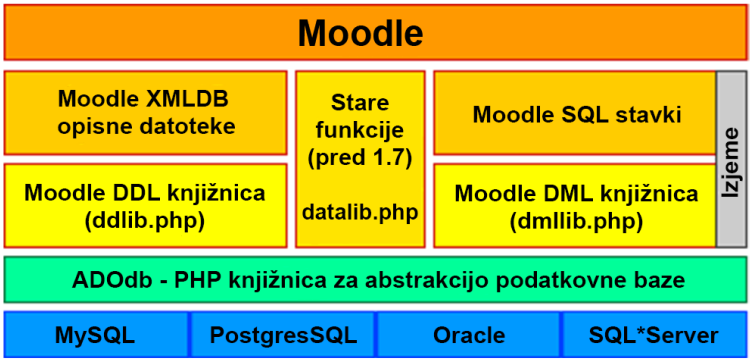
- Knjižnica DML v Moodle (Data Manipulation Language) vsebuje funk-

cije, ki poskrbijo za abstrakcijo upravljanja s podatki v podatkovni bazi. Vso upravljanje s podatki v podatkovni bazi se mora izvajati izključno s temi funkcijami. S tem je zagotovljeno, da bo naš vtičnik deloval na vseh različnih tipih podatkovnih baz, ki jih Moodle podpira.

Podobno kot pri uporabi DDL knjižnice, se tudi te funkcije kličejo preko globalnega objekta $\$DB$. Tukaj imamo dve skupini funkcij. Prva so poizvedovalne funkcije, ki lahko vračajo en vnos (npr. *get_record()*) ali pa več vnosov v obliki tabele (npr. *get_records_select()*). Če uporabljamo kompleksnejše poizvedbe (kjer na primer povezujemo več tabel preko operacije JOIN), ponavadi niso dovolj običajne funkcije, ki so na voljo. V tem primeru lahko uprabimo funkcijo *get_records_sql()*, kjer ji za argument podamo kar stavek SQL. Ta funkcija potem prevede ta stavek v ustrezno obliko glede na tip podatkovne baze, ki ga strežnik uporablja. Druga skupina pa so funkcije, ki služijo za dodajanje, brisanje in spreminjanje vnosov v tabeli.

- Datoteka *datalib.php* vsebuje stare funkcije, ki so skrbele za upravljanje s podatkovno bazo pred Moodle 1.7. Ta knjižnica je zapuščina prejšnjih verzij Moodla in se bo opustila, ko bodo vse funkcije premeščene na ustrezno mesto.

- Funkcije za dostop PHP do podatkovne baze niso standardizirane. Zato Moodle uporablja objektno orientirano knjižnico ADOdb, ki služi za abstrakcijo operacij s podatkovno bazo [4]. S tem je omogočena prenosljivost med sistemi z različnimi podatkovnimi bazami. Trenutno knjižnica podpira podatkovne baze MySQL, Oracle, Microsoft SQL Server, Sybase, Sybase SQL Anywhere, Informix, PostgreSQL, FrontBase, SQLite, Interbase, Foxpro, Access, ADO, DB2, SAP DB in ODBC. Poleg Moodla jo uporablja še veliko popularnih spletnih aplikacij, kot sta na primer phpLens in PostNuke.



Slika 2.3: Interakcija sistema Moodle s podatkovno bazo

2.3 Moodle knjižnice

Pri izdelavi vtičnika se lahko uporabi vrsta knjižnic, ki nam lahko olajšajo razvoj. V našem vtičniku so bile na primer uporabljene knjižnica za izris tabel (*tablelib.php*), knjižnica za izris grafov (*graphlib.php*) in knjižnica za izris obrazcev ($formslib.php$).

Vsako knjižnico je potrebno vključiti v kodo z ukazom *require*, nato pa lahko definiramo objekt razreda, ki ga uporabljamo, in z njim upravljamo s pomočjo že definiranih funkcij.

Knjižnica *tablelib* vsebuje vse potrebne funkcije za izris tabel. Zelo pomembna funkcionalnost je izris tabele s poizvedbami SQL (funkcija $\$table\_sql()$). To je zlasti uporabno pri testiranju pravilnosti poizvedb, saj dobimo rezultat, izpisan v pregledni obliki, ki ga lahko sortiramo po katerikoli celici, tabeli pa moramo podati samo poizvedbo, ki jo želimo izvesti. Poleg tega nam knjižnica omogoča tudi dodajanje funkcionalnosti za prenos tabele v obliki HTML ($optional\_param('download', '', PARAM\_ALPHA)$). Tabelo izrišemo tako, da ji definiramo stolpce ($define\_columns()$), naslovno vrstico ($define\_headers()$) in podatkovne vrstice ($add\_data()$). Naknadno lahko tabeli definiramo tudi celo vrsto atributov (kateri stolpci se sortirajo, kateri stolpci se lahko zaprejo, okvir, ...).

Knjižnica *graphlib* omogoča izdelavo grafov. Izriše se lahko linijski, točkovni, stolpični in ploščinski graf. V našem primeru je izdelan stolpični graf, ki je uporabljen za prikaz histograma ocen plagiatorstva.

Knjižnica *formslib* omogoča uporabo vseh osnovnih elementov HTML (izbirni gumb, potrditveno polje, tekstovno polje, ...). Poleg osnovnih je na voljo tudi dodajanje zahtevnejših elementov, kot so na primer urejevalnik besedila, polje za geslo in izbira datoteke. Vsi elementi se dodajajo preko funkcije *addElement()*, pri čemer se običajno elemente povezuje v skupine znotraj polja *fieldset*. Fieldset ustvarimo z ukazom *addElement('header', 'ime_fieldseta', 'besedilo_legende')*, ki ga postavimo pred skupino elementov, ki jih združujemo. Elementom v obrazcu lahko dodamo celo vrsto atributov, kot so privzeta vrednost (*setDefault()*), onemogočenost (*disabledIf()*) in pravila (*addRule()*), ki se uporabljajo za postavljanje omejitev na posamezne elemente obrazca. Pravila vključujejo preverjanja, kot so obveznost, maksimalna dolžina, preverjanje e-pošte, ... Ta preverjanja se lahko izvajajo na strani strežnika ali pa na strani obiskovalca. V primeru, da jih želimo izvajati na strani obiskovalca (client-side), se to izvaja v obliki kode JavaScript, kjer je ob zahtevnejših pravilih potrebno napisati svojo funkcijo. Vsem elementom je omogočeno tudi dodajanje pomoči s funkcijo *addHelpButton()*.

Poglavje 3

Razvoj vtičnika za plagiatorstvo

Cilj praktičnega dela diplomske naloge je bil razširitev funkcionalnosti sistema Moodle, da bi lahko zaznaval plagiatorstvo med respondenti kvizov.

3.1 Zahteve

Prvi del razširitve naj bi skrbel za analizo plagiatorstva na nivoju posameznega kviza. Za vsakega respondenta je bilo potrebno izračunati vrednost (hevristično oceno plagiatorstva), ki bi bila indikator verjetnosti prepisovanja za tega respondenta pri tem kvizu. Ker so kvizi lahko precej različni (glede količine in tipa vprašanj), je bilo to oceno potrebno izračunati na podlagi več indikatorjev. Prvi indikator bi primerjal respondente med seboj na podlagi ujemanja odgovorov, drugi na podlagi ujemanja napačnih odgovorov in tretji na podlagi časa, ki ga je respondent porabil pri reševanju. Prvi indikator bi temeljil na največji podobnosti odgovorov med dvema respondentoma ali pa na podlagi podobnosti odgovorov respondenta z najbolj pogostimi odgovori. Drugi indikator bi izpostavil respondente, ki so v manjšini odgovarjali enako napačno, pri tem pa so ti odgovori zelo redki v primerjavi z drugimi. Tretji indikator pa bi meril odstopanje respondenta od povprečnega časa reševanja kviza s ciljem, da zaznamo respondente, ki so kviz rešili v zelo kratkem času

(ter s tem verjetno prepisali odgovore). Vsi ti indikatorji bi bili nastavljivi, kar pomeni, da bi lahko vsakemu določili, kako velik vpliv ima na končno oceno (mu nastavili utež). Pri prvem indikatorju pa bi lahko tudi nastavili, katero podobnost uporabljamo pri končni oceni (največjo podobnost ali pa podobnost najbolj pogostim odgovorom). Vse indikatorje je bilo potrebno izpisati v tabeli vseh respondentov kviza, pri čemer bi bilo mogoče sortirati po katerikoli vrednosti. Kot pomoč pri iskanju optimalnih nastavitev za plagiatorstvo je bilo na dnu potrebno izrisati tudi histogram, ki bi prikazoval porazdelitev končne ocene plagiatorstva med vsemi respondenti posameznega kviza.

Drugi del razširitve pa bi služil analizi plagiatorstva na nivoju posameznega predmeta. Torej bi upošteval vse končne ocene plagiatorstva respondenta, ki so bile izračunane za vse kvize, ki jih je reševal. Iz vseh ocen je bilo potrebno izračunati povprečno vrednost za vsakega respondenta in potem tudi ta rezultat izpisati v tabeli. Tudi tu je morala tabela omogočati sortiranje po vrednosti. Ta del pa je moral omogočati tudi ugotavljanje gruč med študenti. Na ta način bi se lahko ugotovilo, kateri respondenti pogosto sodelujejo med seboj. Potrebno je bilo uporabiti enega od algoritmov za zaznavanje gruč in potem gruče izpisati v tabeli ter izrisati na grafu. Ta del vtičnika je moral omogočati tudi nastavljanje števila gruč, ki bi jih želeli izračunati.

3.2 Načrt

Prvi del razširitve smo implementirali v obliki poročila kviza. Ta se nahaja v direktoriju mod->quiz->report->plagiarism. Vse funkcionalnosti so definirane v razredu, ki je razširitev razreda $quiz\_default\_report$. Na ta način so na razpolago funkcije, ki se pogosto uporabljajo pri prikazu poročil (na primer funkcija za pridobivanje vseh vprašanj v kvizu).

V tem delu vtičnika smo za vsakega respondenta, ki je odgovarjal na kviz, izračunali štiri delne ocene (indikatorje plagiatorstva), iz njih pa smo na podlagi nastavitev izračunali tudi končno oceno plagiatorstva.

Prva delna ocena je ocena najvišje podobnosti. Ta se izračuna tako, da se primerja vse odgovore med respondenti. Za vsako vprašanje pri paru respondentov, kjer imamo popolno ujemanje, se števec poveča za 1. Na koncu se za vsakega respondenta upošteva najvišja ocena. Na ta način izpostavimo respondente, ki imajo med seboj veliko identičnih odgovorov. Težava tega indikatorja je, da visoko oceno dobijo tudi respondenti, ki so na večino vprašanj odgovorili pravilno (to se lahko zgodi v primeru, ko je kviz zelo lahek). V tem primeru to ne bo več dober indikator plagiatorstva in mu bo potrebno v skladu s tem ustrezno nastaviti utež za izračun končne ocene.

Druga delna ocena je podobnost povprečnemu študentu. To pomeni, da na začetku izračunamo najpogostejši odgovor za vsako vprašanje v kvizu. Nato preštejemo ujemanje teh odgovorov z vsemi odgovori posameznega respondenta.

Tretja ocena je ocena odstopanja dejanskega časa reševanja od povprečnega časa reševanja kviza. Enota ocene so sekunde, pri čemer je lahko vrednost negativna ali pozitivna (odvisno od tega, ali je respondent rešil kviz nadpovprečno ali podpovprečno hitro). Pri izračunu končne ocene se upoštevajo samo negativne vrednosti (torej respondenti, ki so rešili kviz nadpovprečno hitro), saj prepočasno reševanje ni indikator plagiatorstva. Želimo namreč izpostaviti respondente, ki so rešili kviz prehitro, da bi do odgovorov lahko prišli sami.

Zadnja delna ocena pa izpostavlja respondente, ki so odgovorili na določena vprašanja napačno, pri čemer je tudi še peščica ostalih respondentov naredila isto napako. To izmerimo tako, da za vsakega respondenta preštejemo vse njegove napačne odgovore, nato pa preštejemo odgovore respondentov, ki so odgovarjali enako na ta vprašanja. To število se na koncu deli s številom vseh respondentov. Ocena bo blizu vrednosti 1 v primeru, ko bo respondent odgovarjal pravilno ali pa napačno in podobno kot večina. Več kot bo imel unikatnih napačnih odgovorov, bolj se bo približal oceni 0. Ker želimo, da z verjetnostjo plagiatorstva indikator narašča, bomo na koncu 1 delili s pridobljeno vrednostjo. Ta ocena bo služila kot dopolnilo prvi oceni (oceni najvišje podobnosti),

saj bo odstranila respondente, ki bodo dobili visoko oceno podobnosti zaradi velikega števila pravilnih odgovorov.

Na koncu vse delne ocene uporabimo za izračun končne ocene plagiatorstva. Pri tem se delne ocene normalizirajo (da bodo vse vrednosti vsake ocene med 0 in 100) ter množijo z nastavljenimi utežmi. Zato je tudi vrednost končne ocene vedno med 0 in 100. Vse ocene se hranijo v tabeli podatkovne baze, saj jih tako ni potrebno vsakič računati na novo (če ne bo novih vnosov respondentov). To je potrebno tudi zato, ker se vse te vrednosti uporabljajo v drugem delu vtičnika (plagiatorstvo za cel predmet). Vsaka vrstica v tabeli vsebuje identifikacijo respondenta, kviza in tečaja ter vse izračunane ocene plagiatorstva.

Prvi element na strani je obrazec z nastavitvami za izračun končne ocene plagiatorstva. Za implementacijo smo uporabili razred *formslib.php*, ki je namenjen izdelavi obrazcev. Sestavljen je iz dveh nastavitev. Prva ponuja izbiro osnove za oceno podobnosti. Med možnostmi izbire sta: najvišja podobnost ali pa podobnost povprečnemu študentu. Ta nastavitev pove, katera od dveh ocen se upošteva pri izračunu končne ocene plagiatorstva. Druga nastavitev je sestavljena iz treh vnosnih polj. Vsako od vnosnih polj sprejme vrednost med 0 in 1 in določa utež, s katero se posamezna nastavitev upošteva pri končni oceni. Utež se lahko določi za oceno podobnosti, oceno časa in oceno ujemanja napačnih odgovorov. Ker mora biti vsota vseh treh uteži 1, je možen vnos samo za dve polji, tretje pa bo onemogočeno, pri čemer se avtomatsko izračuna na podlagi prvih dveh s pomočjo programa v jeziku JavaScript.

Drugi element na strani je tabela, ki vključuje vse respondente, ki so reševali izbrani kviz. Ta se izriše s pomočjo knjižnice *tablelib.php*, ki je namenjena izrisovanju tabel. Za vsakega respondenta izpišemo ime, priimek in e-pošto. Izpišemo tudi vse delne ocene plagiatorstva (ocena najvišje podobnosti, ocena podobnosti povprečnemu študentu, ocena časa in ocena ujemanja napačnih odgovorov). Na koncu za vsakega respondenta izračunamo končno oceno na podlagi nastavitev na vrhu. Tabela omogoča sortiranje po kateremkoli stolpcu

ter tudi izključevanje kateregakoli stolpca zaradi večje preglednosti.

Zadnji element tega dela vtičnika je histogram. Izrisan je s pomočjo knjižnice *graphlib.php*. Prikazuje porazdelitev končnih ocen plagiatorstva za vse respondente kviza.

Drugi del razširitve smo implementirali v obliki poročila predmeta. Nahaja se v datoteki report->globalplagiarism. Osnova, na podlagi katere se izvajajo vsi izračuni v tem delu razširitve, so končne ocene plagiatorstva, izračunane v prvem delu razširitve (plagiatorstvo znotraj kviza).

Prvi element poročila je tabela, ki vsebuje respondente, ki so odgovarjali na kvize znotraj izbranega predmeta. Tudi to tabelo izrišemo s pomočjo knjižnice *tablelib.php*. Vsebuje ime, priimek in e-pošto respondenta ter povprečno oceno plagiatorstva. To oceno izračunamo kot povprečje vseh končnih ocen plagiatorstva za vse kvize, ki jih je posamezen respondent reševal. Tabelo lahko sortiramo po vseh poljih, ravno tako pa je mogoče tudi izključevanje posameznih stolpcev zaradi večje preglednosti.

Drugi element na strani je obrazec, ki omogoča nastavljanje števila gruč respondentov. Omogoča izbiro števila gruč med 2 in 9. Za implementacijo smo uporabili knjižnico *formslib.php*.

Tretji element je tabela, ki prikazuje vse respondente znotraj posameznih gruč. Število gruč bo odvisno od zgornje nastavitve. Vsaka vrstica v tabeli vsebuje ime in priimek respondenta, vsak stolpec pa posamezno gručo, ki ji respondenti pripadajo. Gruče izračunamo na podlagi matrike razdalj med posameznimi respondenti. Ta se izračuna tako, da se za vsak par respondentov izračuna število identičnih odgovorov (za vsako ujemanje se bo števec povečal za 1). Pri tem se enako obravnava vse tipe vprašanj. S tem se izračuna matrika podobnosti (večje število pomeni večjo podobnost med respondentoma), ki pa se jo pretvori v matriko razdalj z množenjem z -1. Na matriki razdalj smo pognali algoritem k-medoids, ki je primeren za iskanje gruč na podlagi matrike razdalj.

Zadnji element v tem delu razširitve vsebuje graf, ki prikazuje gruče rese-

pondentov. Za izris grafa gruč je potreben algoritem, ki preslika večdimenzionalne vektorje v 2-dimenzionalni prostor. Ker taki algoritmi izvajajo zahtevne računske operacije na matrikah, smo za izračun uporabili program R, ki je primeren za take statistične izračune.

3.3 Orodja

Razvoj je potekal na razvojni platformi WampServer (Windows, Apache, MySQL, PHP). Gre za okolje, ki omogoča razvoj aplikacij na strežniku Apache s podporo PHP ter podatkovno bazo MySQL. WAMP vsebuje tudi aplikacijo PHPMyAdmin, ki omogoča lažji pregled in urejanje podatkovne baze.

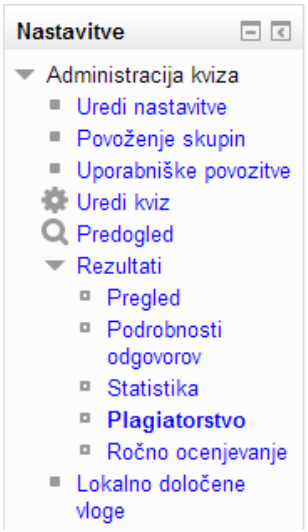
Izdelava vtičnika je potekala v skriptnem programskem jeziku PHP [5], saj je celoten sistem Moodle napisan v tem jeziku. PHP je skriptni jezik, ki se uporablja predvsem za razvoj spletnih aplikacij. Deluje tako, da se koda izvede na spletnem strežniku, rezultat pa se pošlje odjemalcu (običajno v obliki HTML kode spletne strani).

Za izris grafa gruč se je uporabilo okolje R, ki je namenjeno predvsem statistični obdelavi podatkov.

3.4 Plagiatorstvo znotraj kviza

Poročilo plagiatorstva se prikaže v podmeniju *Rezultati* znotraj posameznega kviza, kar je prikazano na sliki 3.1. Vsebuje 3 elemente:

- tabelo z vsemi respondenti in njihovimi ocenami plagiatorstva, ki so odgovarjali na kviz in bili tudi ocenjeni,
- nastavitve tabele, ki določajo, na kakšen način se izračuna končna ocena plagiatorstva,
- histogram, ki prikazuje porazdelitev končnih ocen plagiatorstva za vse respondente.



Slika 3.1: Navigacija vtičnika za plagiatorstvo

3.4.1 Podatkovna baza

Podatke, ki so vsebovani v tabeli, pridobimo iz nove tabele v podatkovni bazi z imenom *quiz_plagiarism*. Predpona *quiz* je obvezna vsem tabelam, ki so del vtičnikov, ki ponujajo razširitev funkcionalnosti kvizov. Tabela se zgradi ob naložitvi vtičnika iz datoteke install.xml, v kateri je definirana:

```
<TABLE NAME="quiz_plagiarism">
 <FIELDS>
  <FIELD NAME="id" TYPE="int" LENGTH="10"
   NOTNULL="true" SEQUENCE="true"/>
  <FIELD NAME="quiz_id" TYPE="int" LENGTH="10"
   NOTNULL="true" SEQUENCE="false"/>
  <FIELD NAME="user_id" TYPE="int" LENGTH="10"
   NOTNULL="true" SEQUENCE="false"/>
  <FIELD NAME="question_score" TYPE="number"
   LENGTH="15" NOTNULL="false" SEQUENCE="false"
   DECIMALS="5"/>
```

```
  <FIELD NAME="question_score_avg" TYPE="number"
   LENGTH="15" NOTNULL="false" SEQUENCE="false"
   DECIMALS="5"/>
  <FIELD NAME="question_score_wrong" TYPE="number"
   LENGTH="15" NOTNULL="false" SEQUENCE="false"
   DECIMALS="5"/>
  <FIELD NAME="time_score" TYPE="number"
   LENGTH="15" NOTNULL="false" SEQUENCE="false"
   DECIMALS="5"/>
  <FIELD NAME="score" TYPE="number" LENGTH="15"
   NOTNULL="false" SEQUENCE="false" DECIMALS="5"/>
 </FIELDS>
 <KEYS>
  <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
  <KEY NAME="quiz_id" TYPE="foreign" FIELDS="quiz_id"
   REFTABLE="quiz" REFFIELDS="id"/>
  <KEY NAME="user_id" TYPE="foreign" FIELDS="user_id"
   REFTABLE="user" REFFIELDS="id"/>
 </KEYS>
 <INDEXES>
  <INDEX NAME="quiz_id-user_id" UNIQUE="true"
   FIELDS="quiz_id, user_id"/>
 </INDEXES>
</TABLE>
```

- polje *id* je primarni ključ tabele, ki unikatno določa vsako posamezno vrstico v tabeli,
- polji *quiz_id* in *user_id* določata kviz in respondenta, na katerega se vrstica nanaša. Obe polji sta tuja ključa, pri čemer se prvo navezuje na polje *id* v tabeli *quiz* in drugo na *id* v tabeli *user*,

- polji *quiz_id* in *user_id* skupaj tvorita unikatni indeks tabele, saj se ista kombinacija vrednosti lahko pojavi v tabeli samo enkrat (vsak par vrednosti je unikaten),
- polja *question_score*, *question_score_avg*, *question_score_wrong*, *time_score* in *score* vsebujejo številske vrednosti posameznih ocen plagiatorstva za respondenta v določenem kvizu. Polja so omejena na 15 mestne številske vrednosti z največ 5 decimalnimi mesti.

3.4.2 Tabela plagiatorstva

V tabeli predstavlja vsaka vrstica enega respondenta, ki je odgovoril na kviz. Poleg imena, priimka in e-pošte respondenta se zraven izpišejo tudi 4 delne ocene in končna ocena plagiatorstva, ki je izračunana iz njih. Tabela se izriše s pomočjo knjižnice *tablelib.php*, s katero ustvarimo objekt *flexible_table*. To je tabela, ki omogoča sortiranje in izključevanje posameznih stolpcev zaradi boljše preglednosti.

Sortiranje deluje tako, da se na začetku izrisovanja tabele uporabi funkcijo *get_sql_sort()*, ki vrne ime stolpca, po katerem želimo sortirati, ter tip sortiranja (naraščajoče ali padajoče). Nato se ta podatek posreduje funkciji, ki izvede poizvedbo ter vrne podatke, ki jih izrišemo v tabeli. Izključevanje stolpcev se doseže s pomočjo funkcije *collapsible*(*true*).

Primer tabele za kviz, ki vsebuje 5 respondentov, je prikazan na sliki 3.2.

3.4.3 Ocena najvišje podobnosti

Prva ocena, ki se izračuna, je ocena najvišje podobnosti. Izračuna se tako, da se primerja odgovore na posamezna vprašanja v kvizu za vsak par respondentov. Vsi uporabniki so shranjeni v tabeli *mdl_user*. Pri tem se v primerjavi upoštevajo samo tisti, ki so izbrani kviz reševali in je bil njihov poskus tudi ocenjen. Vsi ti respondenti dobijo status *finished*, ki se zabeleži v tabeli *mdl_quiz_attempts*.

ID	Ime	Priimek	E-pošta	Najvišja podobnost	Podobnost povprečnemu študentu	Podobnost napačnih odgovorov	Razlika časa od povprečja	Ocena
7	Student	Pet	student.pet@xx.com	11	11	0.68	15.6	67.2
6	Student	Stiri	student.stiri@xx.com	6	3	0	41.6	21.8
5	Student	Tri	student.tri@xx.com	11	11	0.68	-5.4	68.7
4	Student	Dva	student.dva@xx.com	3	1	0.87	-22.4	51.9
3	Student	Ena	student.ena@xx.com	6	1	0	-29.4	30.1

Slika 3.2: Tabela plagiatorstva na nivoju posameznega kviza

Moodle ima na voljo več različnih tipov vprašanj, ki se lahko vstavijo v kviz, vendar pa se vsi odgovori shranijo v bazo na enak način - v obliki niza. Pri tem se podvprašanja ločijo med seboj s podpičjem. Vsi ti odgovori se shranjujejo v tabeli *mdl_question_attempts* v polju *responsesummary*.

Primer odgovora v bazi na vprašanje s 3 podvprašanji:

```
Prvo podvprasanje -> Izbran odgovor 1;Drugo
podvprasanje -> Izbran odgovor 2; Tretje
podvprasanje -> Izbran odgovor 3
```

Zato je primerjava posameznih odgovorov tudi precej lažja. Za vsako vprašanje oz. podvprašanje se primerja odgovor iz baze in ob popolnem ujemanju se števec ocene poveča za 1. Na koncu se za vsakega respondenta izpiše najvišja pridobljena ocena.

3.4.4 Podobnost povprečnemu respondentu

Druga ocena je variacija prve podobnosti. V tem primeru ne iščemo najvišje podobnosti za vsakega respondenta, ampak podobnost povprečnemu respondentu (ki je določen z najbolj pogostimi odgovori). Najprej se za vsako vprašanje pridobi odgovor, ki je bil največkrat odgovorjen. Ti odgovori predstavljajo povprečnega respondenta. Nato pa vsakega respondenta primerjamo s povprečnim

respondentom na enak način kot pri oceni najvišje podobnosti.

3.4.5 Podobnost napačnih odgovorov

Tretja ocena temelji na napačnh odgovorih posameznega respondenta. Tukaj nas zanimajo odgovori respondenta, ki so napačni in hkrati zelo redki glede na ostale respondente.

Prvi korak pri izračunu te ocene je pridobitev vseh pravilnih odgovorov v kvizu. Nato primerjamo vse odgovore posameznega respondenta s pravilnimi in kjer ni ujemanja, zabeležimo napačen odgovor. Ti napačni odgovori nam služijo kot osnova za izračun ocene pri posameznem respondentu. Sledi iteracija čez vse ostale respondente. Pri tem za vsak par izvedemo primerjavo pridobljenih napačnih odgovorov ter za vsako ujemanje povečamo števec za 1. To število potem delimo s številom vseh napačnih odgovorov prvotnega respondenta (na ta način dobimo delež napačnih odgovorov, ki se ujemajo). Postopek ponavljamo za vsak par, kjer to število prištejemo skupni oceni napačnih odgovorov. Po zaključku iteracije delimo oceno s številom vseh respondentov, ki so odgovarjali na kviz.

Vrednost končne ocene je med 0 in 1, pri čemer velja, da je verjetnost prepisovanja med majhnim številom respondentov večja, bolj kot se vrednost približuje 0. Zato na koncu oceno obrnemo (1-ocena), tako da je višja vrednost ocene indikator plagiatorstva.

3.4.6 Odstopanje od povprečnega časa

Zadnja ocena, s katero ugotavljamo verjetnost prepisovanja posameznega respondenta, temelji na času reševanja.

Pridobimo jo tako, da na začetku izračunamo povprečen čas, ki je bil porabljen za reševanje kviza. V tabeli *mdl_quiz_attempts* so zapisani časi začetka in konca reševanja kviza za vsakega respondenta. Zato se povprečen čas izračuna enostavno z ukazom *AVG(timefinish-timestart)*.

Nato za vsakega respondenta izračunamo razliko med povprečnim časom in dejanskim časom. Ta vrednost je merjena v sekundah in je lahko negativna ali pozitivna (odvisno od tega, ali je respondent rešil kviz hitreje ali počasneje od povprečnega časa).

3.4.7 Končna ocena plagiatorstva

Končna ocena plagiatorstva (3.1) posameznega respondenta se izračuna na koncu iz delnih ocen. Gre za vsoto normaliziranih delnih ocen (vrednosti se nahajajo med 0 in 100), pri čemer se upoštevajo nastavljene uteži na vrhu tabele. Pri izračunu se uporabi ocena časa, ocena napake in ocena podobnosti.

Pri oceni časa se upošteva samo negativne vrednosti (torej respondente, ki so odgovorili hitreje od povprečnega časa reševanja), saj predolgo reševanje ni pokazatelj plagiatorstva.

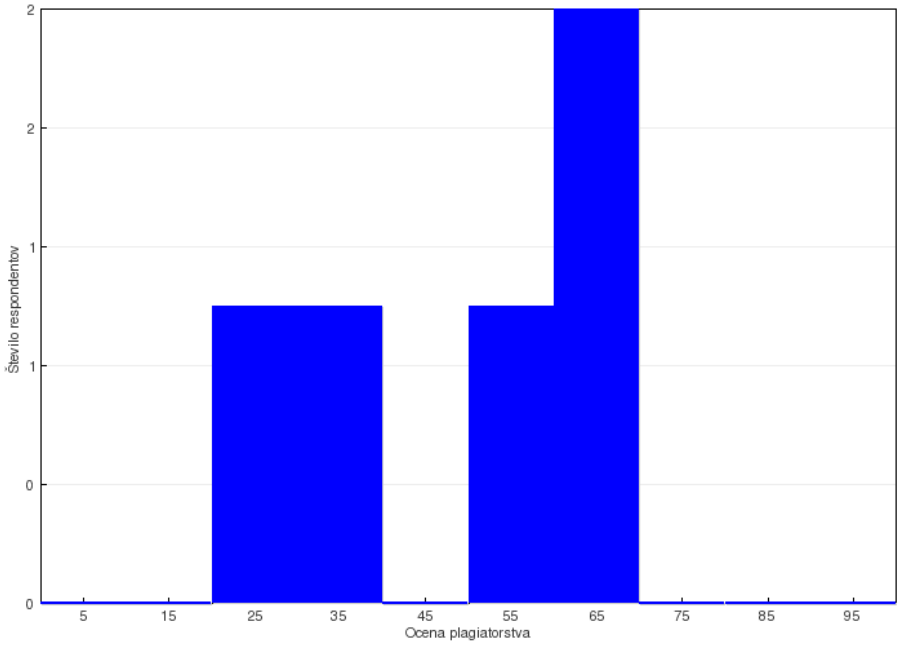
Za oceno podobnosti se lahko izbere ocena najvišje podobnosti (privzeto), lahko pa se nastavi, da se namesto nje uporabi podobnost povprečnemu študentu:

$$\begin{aligned} ocenaPlagiatorstva = &(utezNapake * normaliziranaOcenaNapake) \\ &+ (utezCasa * normaliziranaOcenaCasa) \\ &+ (utezPodobnosti * normaliziranaOcenaPodobnosti) \end{aligned} \quad (3.1)$$

3.4.8 Histogram ocen plagiatorstva

Pod tabelo se izriše histogram izračunanih končnih ocen plagiatorstva za posamezen kviz, ki je prikazan na sliki 3.3. Histogram prikazuje porazdelitev končnih ocen, kar nam izboljša celotno predstavo o prepisovanju za celoten kviz. Pomemben je tudi zato, ker nam v primeru, da je porazdelitev zelo neuravnotežena, pove, da moramo prilagoditi posamezne nastavitve izračuna, saj trenutne ocene niso realen prikaz plagiatorstva.

Izris histograma se izvede s pomočjo knjižnice *graphlib.php*. Primer uporabe knjižnice za izris histograma:



Slika 3.3: Histogram končnih ocen plagiatorstva za izbran kviz

```
// Ustvarimo nov objekt razreda graph
$graph = new graph(704, 500);

// Določimo naslov grafa
$graph->parameter['title'] = '';

// Določimo naslov osi X
$graph->parameter['x_label'] = get_string(
        'plagiarismgraph_x', 'quiz_plagiarism');

// Določimo naslov osi Y
$graph->parameter['y_label_left'] = get_string(
        'plagiarismgraph_y', 'quiz_plagiarism');

// Določimo naklon besedil obeh osi
$graph->parameter['y_label_angle'] = 90;
$graph->parameter['x_label_angle'] = 0;

// Podamo intervale na osi X
$graph->x_data = array(5,15,25,35,45,55,65,75,85,95);

// Podamo podatke, ki jih želimo izrisati na grafu
$graph->y_data['data'] = $histogram_data;

// Določimo tip grafa, barvo stolpcev in senčenje
$graph->y_format['data'] = array(
        'colour' => 'blue',
        'bar' => 'fill',
        'shadow_offset' => 1
);
```

```
$graph->y_order = array('data');

// Določimo debelino stolpcev, razmak in mrežo
$graph->parameter['bar_size'] = 1;
$graph->parameter['bar_spacing'] = 1;
$graph->parameter['x_grid'] = 'none';

// Določimo meje na osi Y (med 0 in največjo vrednostjo)
$hist_max = max($histogram_data);
$graph->parameter['y_min_left'] = 0;
$graph->parameter['y_max_left'] = $hist_max;

// Izrišemo graf
$graph->draw();
```

3.5 Plagiatorstvo na ravni celega predmeta

Drugi del vtičnika za plagiatorstvo deluje na nivoju celega predmeta (vseh kvizov, ki so del predmeta). Nahaja se v poročilih posameznega predmeta. Podobno kot prvi del vtičnika tudi ta vsebuje 3 elemente:

- tabelo z vsemi respondenti, ki so odgovorili in bili ocenjeni za vsaj 1 kviz znotraj predmeta,
- tabelo gruč, v kateri se prikažejo skupine študentov, ki so reševali podobno. Število gruč se nastavi na vrhu strani,
- graf gruč, ki prikaže skupine študentov, ki so reševali podobno. Ta element se izriše s pomočjo programa R. Če R ni naložen na strežniku, kjer se nahaja inštalacija sistema Moodle, se graf ne prikaže.

3.5.1 Tabela plagiatorstva za vse kvize

V tabeli vsaka vrstica predstavlja enega respondenta, ki je odgovoril na vsaj 1 kviz znotraj predmeta. Poleg imena, priimka in e-pošte respondenta se zraven izpiše še povprečna ocena plagiatorstva. Izračuna se s funkcijo AVG znotraj poizvedbe za podatkovno bazo:

```
SELECT AVG(score) as avgscore
FROM mdl_quiz_plagiarism
```

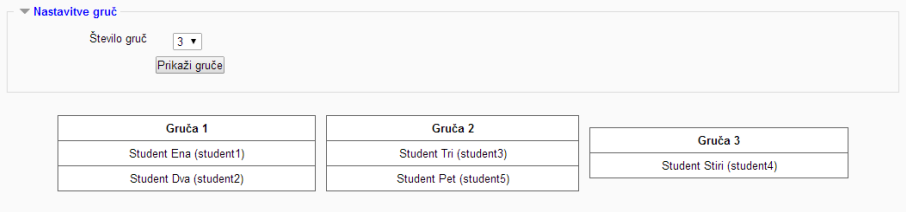
Tabela se izriše na enak način kot tabela plagiatorstva na nivoju posameznega kviza, s pomočjo knjižnice *tablelib.php*. Tudi ta tabela omogoča tako sortiranje po stolpcih, kot tudi izključevanje posameznih stolpcev.

3.5.2 Gruče respondentov

Gručenje vključuje metodologije, ki avtomatsko razvrščajo člane nekega vzorca v določeno število gruč [8]. To se izvede na podlagi mere podobnosti, kar pomeni, da so člani znotraj ene gruče med seboj podobni, člani med posameznimi gručami pa si niso. V našem primeru to pomeni iskanje skupin respondentov, kjer isti skupini pripadajo respondenti, ki imajo pogosto podobne odgovore. Algoritmov za gručenje obstaja veliko, zato je bilo potrebno izbrati takega, ki bi bil najbolj učinkovit v našem primeru.

Elementi, ki jih primerjamo med sabo za gručenje, so vektorji. Vsak vektor predstavlja enega respondenta in vsebuje vse odgovore na kvize, ki jih je rešil znotraj izbranega predmeta. Vektor je dimenzije n, pri čemer n pomeni število vprašanj v vseh kvizih znotraj predmeta. V našem primeru je težava v tem, da so ti odgovori tekstovne in ne številske vrednosti. Zaradi tega ne moremo uporabiti algoritmov za gručenje, ki zahtevajo številske vrednosti za posamezne dimenzije vektorja.

Najprej se iz vektorjev, ki predstavljajo respondente izračuna matrika razdalj. To naredimo tako, da primerjamo vektorje med sabo po vsaki dimenziji, in če gre za popolno ujemanje niza (enak odgovor na vprašanje) povečamo



Slika 3.4: Izračun 3 gruč za 5 respondentov

razdaljo med tema dvema respondentoma za 1. S tem dobimo matriko podobnosti, ki jo na koncu množimo z -1, da dobimo matriko razdalj. Matrika je dimenzije $n \times n$, pri čemer n predstavlja število vseh respondentov, ki so odgovarjali na kvize.

V našem primeru je primeren algoritem za gručenje k-medoids [6]. K-medoids je zelo soroden algoritmu k-means s to razliko, da ko naključno izbere element (medoid) gruče, izračuna skupno napako razdalje na podlagi vsote vseh oddaljenosti od izbranega elementa in ne na podlagi oddaljenosti od centroida. Deluje tako, da najprej naključno izbere k elementov (v našem primeru so to respondenti). Nato na podlagi tega določi gruče, kjer je pripadnost elementa določena z najmanjšo razdaljo do na začetku izbranih elementov. Na koncu se izračuna celotna napaka gruč, ki jo predstavlja vsota razdalj elementov znotraj gruče do centralnega elementa. Nato spremenimo enega od na začetku izbranih elementov in postopek ponovimo. Ta postopek ponavljamo, dokler ne ugotovimo najmanjše možne napake, kar pomeni da smo dobili končni nabor gruč.

Slika 3.4 prikazuje primer izpisa gruč.

3.5.3 Graf gruč

Zadnji element plagiatorstva za cel predmet je izris grafa gruč. Vse respondente, za katere so izračunane matrika razdalj in gruče, je bilo potrebno izrisati v 2-dimenzionalnem grafu. V ta namen je bilo potrebno uporabiti enega od

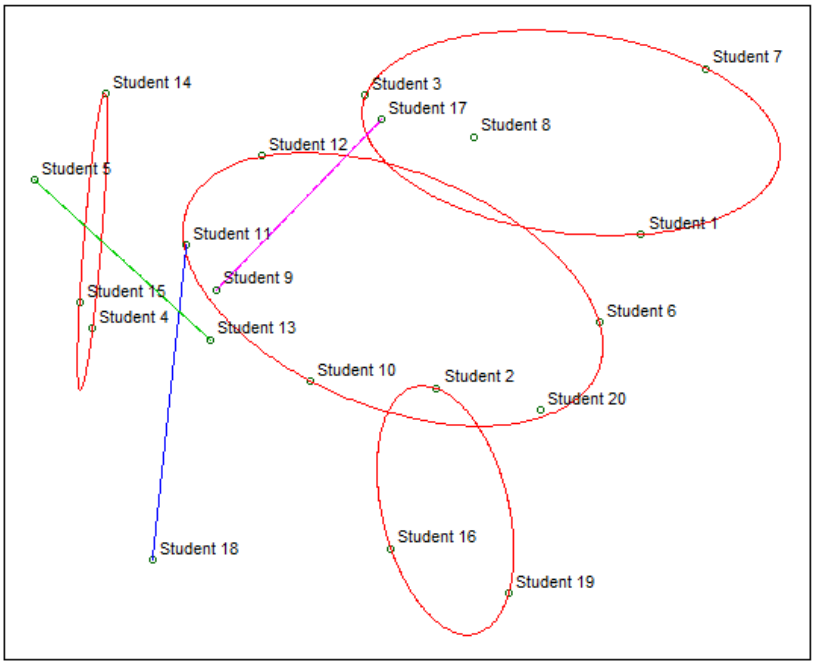
algoritmov, ki izvedejo redukcijo dimenzij [7].

Težava se pojavi v okolju PHP, ki ni optimiziran za delo z matrikami. Zato bi bila implementacija kakršnekoli metode za izris gruč zelo počasna. S tem razlogom smo se odločili, da se za grafični prikaz gruč uporabi okolje R [9]. R je optimiziran za obdelavo podatkov za statistične namene in vsebuje kar nekaj vgrajenih algoritmov za izrisovanje gruč. Če želimo imeti to funkcionalnost vtičnika, mora seveda biti R naložen na strežniku, kjer se nahaja tudi Moodle. Če R ni naložen, se prikaže samo tabela, v kateri so prikazane vse gruče respondentov, ter opozorilo, da je za grafičen prikaz gruč potrebno naložiti R.

R se iz okolja PHP pokliče z ukazom *shell_exec(RScript clustering.R)*, ki izvede ukaz RScript preko ukazne lupine. Ta ukaz požene R skripto *clustering.R*, v kateri se izvede izris slike grafa. Matrika razdalj se poda v R preko začasne datoteke CSV, ki jo zgradimo pred zagonom skripte. Ob klicu skripte R podamo tudi parametre, ki definirajo lokacijo začasne datoteke CSV z matriko razdalj, lokacijo kamor se bo shranil zgrajen graf in spremenljivko k, ki določa število gruč.

Izris gruč se izvede znotraj skripte R z ukazom *clusplot* [10]. *Clusplot* izriše dvodimenzionalen graf na objektu, ki vsebuje podatke o gručah. V našem primeru so to gruče respondentov izračunane s k-medoids (partitioning around medoids) algoritmom in razdalje med posameznimi elementi, ki gradijo gruče (matrika razdalj). Za pripravo podatkov v ustrezni obliki (v obliki 2-dimenzionalnih točk) uporabi *clusplot* algoritem PCoA (metoda poglavitnih koordinat) [11].

Algoritem PCoA najprej postavi vse elemente v prostor na podlagi matrike razdalj. To poteka tako, da najprej postavi prvi element v izhodišče, nato doda drugega na prvo os, da je razdalja do prvega enaka njuni medsebojni oddaljenosti. Tretji element se doda na mesto, kjer oddaljenosti do prvih dveh ustrezata vrednostim v matriki razdalj. Da je to mogoče, je običajno potrebno dodati novo os, torej povečati dimenzijo prostora za 1. Postopek se ponovi za vsak naslednji element, dokler niso v prostor dodani vsi elementi. Na ta način dobimo



Slika 3.5: Graf 7 gruč v primeru 20 respondentov

$n-1$ dimenzionalni prostor, v katerem je n elementov. Drugi del algoritma pa je redukcija dimenzij, za kar se uporabi algoritem PCA (metoda poglavitnih komponent) [12]. Algoritem PCA izvede v našem primeru redukcijo dimenzij na 2, pri čemer želimo doseči čim manjšo izgubo informacij. Nove komponente se izračunajo tako, da se spremenljivkam, iz katerih je sestavljena prva komponenta, priredijo takšne uteži, da bo komponenta pojasnila kar največji del razpršenosti spremenljivk. Nato določimo drugo komponento, pri čemer ne sme korelirati s prvo in hkrati pojasni čim več preostale razpršenosti.

Slika 3.5 prikazuje izris grafa 7 gruč za 20 respondentov.

Poglavje 4

Zaključek

Rezultat diplomskega dela je vtičnik za sistem Moodle, ki omogoča analizo odgovorov respondentov na kvize. Ti se v praksi ponavadi uporabljajo za preverjanje znanja. Pri tem pa se pogosto pojavlja problem plagiatorstva.

Vtičnik nam z analizo odgovorov prikaže verjetnost plagiatorstva posameznega respondenta. Pri tem se lahko izvede analiza na nivoju posameznega kviza ali pa na nivoju vseh kvizov znotraj določenega predmeta. Pri analizi je uporabljenih več algoritmov, ki izračunajo delne ocene plagiatorstva na podlagi različnih kriterijev, kot so ujemajoči odgovori, napačni odgovori, čas reševanja kviza itd. Na ta način je mogoče natančneje odkriti respondente, ki kvizov ne rešujejo samostojno. Analiza nam omogoča tudi odkrivanje gruč respondentov, ki pogosto podobno odgovarjajo. Ta rezultat je prikazan tudi v obliki grafa.

Možnosti za razširitev vtičnika je precej. Precej prostora za izboljšave je še na področju izračuna ocen plagiatorstva. Algoritme bi se dalo prilagoditi, da drugače obravnavajo različne tipe vprašanj. Na primer pri daljših besedilnih odgovorih se zaenkrat preverja samo popolno ujemanje, lahko pa bi se uporabil kakšen algoritem za ugotavljanje podobnosti besedila.

Možnost razširitve je tudi na področju gručenja. Zaenkrat se uporablja samo en algoritem za izračun gruč (k-medoids) in samo en tip grafa za izris

gruč. Če bi bilo na voljo več algoritmov, bi lahko uporabnik izbral algoritem, ki bi bil najbolj primeren v posameznem primeru (glede na tipe vprašanj, ki se pojavljajo v kvizu in število respondentov). Tudi graf, ki prikazuje gruče respondentov, bi bilo mogoče prikazati v drugačnih oblikah (npr. drevesni strukturi, če bi uporabili hierarhično gručenje).

Navsezadnje pa je v prihodnje možno razviti tudi nov tip bloka v sistemu Moodle, ki bi prikazal respondente, ki imajo najvišjo oceno prepisovanja. Ti respondenti bi lahko dobili tudi avtomatsko opozorilo preko e-pošte.

Vtičnik je razvit z vsemi načeli pravilne izdelave, ki so zahtevani pri razvoju vtičnikov za Moodle. Zato ga je mogoče v prihodnje tudi objaviti na spletu in s tem omogočiti javno uporabo.

Literatura

[1] A. Clarke. *E-learning skills*. Palgrave Macmillan, 2008.

[2] Moodle plugins dostopna na:
http://docs.moodle.org/dev/Plugins, 2.4.2014

[3] XMLDB introduction dostopna na:
http://docs.moodle.org/dev/XMLDB_introduction, 3.4.2014

[4] ADOdb dostopna na:
http://adodb.sourceforge.net/, 10.4.2014

[5] L. Atkinson. *Core PHP programming: using PHP to build dynamic Web sites*. Prentice Hall PTR, 2000.

[6] M. Kantardzic. *Data mining: concepts, models, methods, and algorithms*. Wiley-Interscience: IEEE Press, 2003.

[7] Pang-Ning Tan, M. Steinbach, V. Kumar. *Introduction to data mining*. Pearson Addison Wesley, 2006.

[8] I. Kononenko. *Strojno učenje*. Fakulteta za računalništvo in informatiko, 2005.

[9] R dostopna na:
http://www.r-project.org/, 15.5.2014

[10] Pison, Greet & Struyf, Anja & Rousseeuw, Peter J. "Displaying a clustering with CLUSPLOT", Computational Statistics & Data Analysis, št. 30, str. 381-392.

[11] PCoA dostopna na:
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/cmdscale.html, 16.5.2014

[12] PCA dostopna na:
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/princomp.html, 16.5.2014