Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-73915

Michal Hucko

Analýza textov odpovedí študentov na otázky

Bakalárska práca

Vedúci práce: prof. Ing. Mária Bieliková PhD.

Máj, 2017

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-73915

Michal Hucko

Analýza textov odpovedí študentov na otázky

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky
Vedúci práce:	prof. Ing. Mária Bieliková PhD.

Máj, 2017

>>>> ASSIGNMENT <<<<
>>>> ZADANIE <<<<

POĎAKOVANIE

V prvom rade sa chcem poďakovať pani profesorke Bielikovej, ktorá svojimi nápadmi, časom a najmä vynikajúcim prístupom významne dopomohla k vypracovaniu tejto bakalárskej práce.

Taktiež ďakujem doktorantom Matúšovi Pikuliakov a Peťovi Gašparovi, ktorý si každý týždeň našli čas a konzultovali so mnou môj postup. Ďalej ďakujem skupine Pewe, ktorej členovia svojimi nápadmi posunuli kvalitu mojej práce vždy o kus ďalej.

Samozrejme ďakujem aj Vassilisovi Triglianosovi, ktorý ma oboznámil s používaním systému ASQ a vždy pohotovo reagoval na problémy, ktoré nastali pri jeho používaní.

V neposlednom rade ďakujem svojím priateľom a rodine za ich podporu počas vypracovania tejto práce.

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že som bakalársku prácu vypracoval samostatne s využitím uvedených zdrojov literatúry.

V Bratislave, 5.5.2017 .....

Michal Hucko

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informatika
Bakalárska práca: Analýza textov odpovedí študentov na otázky
Autor: Michal Hucko
Vedúci práce: prof. Ing. Mária Bieliková PhD.
Máj, 2017

Kladenie otázok je základný spôsob získavania spätnej väzby vo výučbe. Slúži nie len ako hlavný nastroj pri záverečných testoch, ale je vhodné aj pri overovaní vedomostí počas výučby. V prípade, že sa pracuje s veľkým množstvom opýtaných, následná analýza výsledku môže byť časovo náročná. Klastrovanie korpusu odpovedí by mohlo pomôcť s procesom opravovania, zobrazrením štruktúry. Klastre sú totiž zhluky podobných odpovedí, ktoré môžu odzrkadľovať časté chyby študentov. Keďže sa otázky na školách zvyknú opakovať sezónne, učiteľ mnohokrát disponuje príkladmi správnych a nesprávnych odpovedí. Tento dataset sa môže použiť na vytvorenie klasifikátora, ktorý by umožnil automatickú opravu. V tejto práci sme poskytli riešenie spojené s klasifikáciou a klastrovaním. Na vizualizáciu výsledkov sme použili nami vytvorenú aplikáciu. Výsledky zhlukovania sme overili s prezentačným systémom ASQ, ktorý umožňuje interaktívnu výučbu, skrz kladenie otázok. Pri overovaní klasifikácie používame dataset zozbieraný za posledných 5 rokov. Odpovede sa týkali domény softvérového inžinierstva.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Study program: Informatika
Bachelor thesis: Analysis of students answers to questions
Author: Michal Hucko
Supervisor: prof. Ing. Mária Bieliková PhD.
2017, May

Asking questions is a basic approach when getting feedback while teaching. It is not only a tool in final exams, but it is also appropriate when proving knowledge of class through teaching. Result analysis of hundreds anttendants is very time-consuming. Clustered corpus of answers could help. Clusters are similar answers, which can demonstrate common mistakes of the class. Because tests at schools tend to repeat regularly, teachers often have labeled data from previous years. This data set can be used with training of classifier. Using this classifier we can automatically evaluate new answers. In this work we provided solutions in classification and clustering. For the visualization of results to teacher we developed a web application. We evaluate results of clustering using ASQ, which is an real time presentation software used in teaching with questions and answers. For the classification evaluation we used data set gathered at our school for last 5 years. Answers were all about software engineering domain.

Obsah

1 Úvod

Základnou činnosťou pedagóga na škole je výučba študentov. Táto činnosť je spojená s odovzdávaním nových poznatkov niekedy až stovkám ľudí naraz. Jedným zo spôsobov ako overiť úspech vyučovania je testovanie. Testovanie zahŕňa vypracovanie otázok, na ktoré študenti odpovedajú a učiteľ tak dostane spätnú väzbu. Cieľom učiteľa v tomto procese je hlavne identifikovať stav porozumenia. Výsledkom je zväčša zoznam odpovedí, ktorý tvoria záznamy v neprehľadnom poradí. Štruktúra dokumentu nijako neprezrádza správanie sa triedy študentov. Poradie je zväčša definované abecedou, prípadne iným usporiadaním, ktoré tento jav zapríčiňuje. Po tejto fáze sa prechádza na manuálnu kontrolu odpovedí. Proces je spojený so sekvenčeným prechádzaním zoznamu. Keďže v triede môžu byť aj stovky študentov náročnosť stúpa.

V tomto smere existuje snaha o automatizovanie vymenovaním množiny správnych odpovedí, kedy algoritmus vyhodnotí ako správnu len tú, ktorá sa v množine nachádza. Avšak existujú typy otázok, kde nie je možné vopred určiť všetky správne odpovede. Jedná sa prevažne o otvorené otázky. Napríklad otázka: “Ako je definovaná kvalita softvéru?”. Je jasné, že postup by tu bol neefektívny. S prihliadnutím na tieto predpoklady môžme povedať, že sa naozaj jedná o problém.

Našim cieľom je pomôcť učiteľom práve v tomto smere. Otázky na školách, v rámci overovania znalostí, majú tendenciu opakovať sa sezónne. Zaznamenávaním odpovedí by sme mohli vytvoriť trénovaciu databázu pre klasifikačné algoritmy. Ideálne by sme v ďalších rokoch skúšania nepotrebovali prechádzať odpovede ručne, stačila by predikcia nami natrénovaného klasifikátora. Tá by poslúžila ako odporúčanie, na základe ktorého by sa testy opravovali. Častým opakovaním otázok by sme prispeli k zlepšeniu predikcie.

Nájdením správnych odpovedí však proces nekončí. Pedagóg sa vo všeobecnosti snaží svojím výkladom pomôcť študentom, ktorí majú problém. Ak by sme mu poskytli len množinu správnych odpovedí nemohol by sa v tomto smere zlepšovať. To že vieme povedať, že” n” študentov odpovedalo nesprávne, nám

nedá odpoveď na otázku prečo? Ak by sme však videli, že sa u "k" študentov opakuje konkrétna chyba vedeli by sme svoj výklad na prednáške upraviť. Pri manuálnej kontrole pedagóg dokáže identifikovať problémové oblasti vzorky. Sekvenčné prehľadávanie však zabraňuje identifikácii výrazných vzorov správania študentov. Našim ďalším cieľom je ukázať, že klastrovanie textov môže výrazne dopomôcť aj v tejto oblasti. Implementovali sme riešenie, kde vyučujúci použije korpus odpovedí ako vstup. Výstup je vizualizácia klastrov získaná aplikovaním nášho modelu. Spojenie takejto vizualizácie s real-time prezentačnými systémami by mohli výrazne pedagógom pomôcť. Ak by totiž vedeli počas prednášky zadať test a následne jasne vidieť zhluky odpovedí, vedeli by flexibilne meniť obsah výkladu v záujme pomôcť študentom, ktorí majú problémy s daným učivom.

V našej práci sa venujeme krátkym odpovediam v slovenskom jazyku. Študenti majú mnohokrát obmedzený čas a pracujú pod stresom, čo sa môže výrazne podpísať na ich úprave odpovedí. Konkrétne v našom prípade pracujeme s odpoveďami, ktoré boli zozbierané real-time prezentačným systémom ASQ [18]. V ňom bol priestor na premýšľanie naozaj obmedzený. Nesprávna gramatika, prípadne štylistika sa môžu značne podpísať pod naše výsledky. Túto potenciálnu hrozbu však vieme vyriešiť vhodným predspracovaním textov. Získali sme korpus, ktorý je tvorený viac ako 900 odpoveďami študentov na otázky z domény softvérového inžinierstva. Na prácu s klasifikátormi sme použili dataset zozbieraný počas posledných 4 rokov na tomto predmete. Študenti mali počas testovania priestor zostaviť odpoveď ľubovoľnej dĺžky. Práca so systémom ASQ prebehla v rámci medzinárodné projektu SCOPES s Fakultou informatiky Univerzity v Lugane s názvom Innovative teaching curricula, methods and infrastructures for computer science and software engineering.

Práca pozostáva zo siedmich častí. V druhej kapitole sa venujeme spracovaniu textu. Poskytujeme v nej opis rôznych spôsobov, ktoré sa využívajú nie len v slovenčine. Tretia kapitola sa venuje klasifikácii a klastrovaniu. Je prehľadom postupov v tejto doméne. Okrem ich opisov je v kapitole zahrnutá aj časť metrík, ktorá sa používa pri overovaní. V štvrtej kapitole poskytujeme nami navrhnutú metódu štruktúrovania odpovedí. Na konci opisujeme rozhranie, ktoré sme vyvinuli za účelom vizualizácie výsledkov učiteľovi. Piata kapitola

dokumentuje postupy pri overovaní spojené s experimentom na expertoch. Ním ukazujeme, že použitie metódy sa zhoduje s očakávaním vyučujúcich.

2 Predspracovanie textu

Cieľom predspracovania textu je zjednotiť ho po stránke štruktúry do formy, s ktorou sme ďalej schopný pracovať v klastrovaní a klasifikácii. Korpus ako taký je v našom prípade tvorený odpoveďami študentov. Môžme v ňom vidieť značnú nekonzistenciu. Ľudia sa často ponáhľajú a denne napíšu mnoho krátkych správ, kde na gramatiku a štylistiku nedbajú. Príkladom môžu byť dopyty do googlu alebo sms-ky. Alexander Clark so svojím tímom identifikovali tento problém v práci [1] a vypracovali návrh modelu na spracovanie podobných textov. Ich myšlienka spočíva najmä v tom, že chyby sú často krát obklopené slovami (frázami) s korektnou úpravou. Študenti majú obmedzený čas pri vypracovaní otázok, čo ich núti odpovedať a písať rýchlo. Samozrejme korekcia nesprávnej gramatiky a štylistiky je len drobná časť predspracovania, ktorú ďalej rozoberieme.

Ďalším cieľom je transformovať korpus do takej podoby, ktorá by mohla podporiť kvalitu výsledkov klasifikačných a klastrovacích algoritmov. Tento proces zahŕňa tvorbu n-gramov a výber čŕt textu (angl. feature selection). Je potrebné dodať, že pracujeme s kratšími odpoveďami.

2.1 Lematizácia a gramatické spracovanie

Úlohou predspracovania je vyriešiť problém s gramaticky nekonzistentnými textami spomenutom v úvode kapitoly. Výber vhodnej techniky, k splneniu tohto cieľa, je značne ovplyvnený jazykom, v ktorom je korpus napísaný [9]. V kapitole uvádzame príklady riešení vhodných pre Slovenský jazyk. Sú nimi oprava gramatiky, lematizácia a náhrada problémových písmen.

V rámci opravy gramatiky je vhodné najskôr doplniť diakritiku. Jedným z riešení v anglickom jazyku je systém na kontrolu gramatiky [12]. Systém na vstupe identifikuje chyby a vyberie zo zoznamu korektných slov ich náhradu. Tento výber prebieha na základe hodnotenia, ktoré sa vyráta z kontextu, v ktorom bolo chybné slovo použité. Identický postup sa využíva aj v slovenskom

jazyku [13].

Lematizácia slúži na určenie základného tvaru slova tzv. lemy. Napr. pre slovo „rybníkom“ je lemou „rybník“, pre slovo „klesajúci“ je lemou „klesať“, pre slovo „vznešenou“ je lemou „vznešený“ [13]. Po aplikácii lematizácie na vetu máme prístup k čistým slovám, z ktorých boli odobrané predpony a prípony. Ako príklad uvádzame vetu: “Zaoberá sa procesmi pri vývoji softvéru, jeho údržbe a rozhoduje o jeho zániku.”. Výsledkom bude lematizovaná veta: “Zaoberať sa proces pri vývoj softvér jeho údržba a rozhodovať o jeho zánik”. Rovnaký postup sa využíva aj pri predspracovaní anglických textov [9]. Úspešnú aplikáciu vidíme aj v práci Koreniusa a Laurikaly [6], ktorí porovnávali vplyv vhodného predspracovania textu na klastrovanie fínskych textov. Fnínština splu so slovenčinou patria medzi gramaticky zložitejšie jazyky. Ich práca dokazuje, že práve aplikácia lematizácie má na klastrovanie najlepší vplyv.

Posledný krok je spojený s gramatikou. Po prezretí získaných odpovedí sme identifikovali problém s písaním tvrdého “y” a mäkkého “i”. V rámci slovenčiny poznáme niekoľko pravidiel spojených s písaním týchto dvoch písmen. Jedná sa napríklad o vybrané slová. Sú to slová, po ktorých sa píše vždy tvrdé “y” nech sa vyskytujú v akomkoľvek páde. Keďže sa chystáme v našej metóde využiť porovnávanie slov, nekonzistencia pri týchto písmenách by mohla ovplyvniť výsledky. Ako vhodné riešenie pri predspracovaní je možnosť aplikovania nasledujúcich pravidiel. V prípade mäkčeňov a dĺžňov sme špeciálne znaky jednoducho odstránili. Pri gramatike písania “y” a “i” nahradzujeme všetky písmená “y” písmenom “i”. Po aplikácii týchto pravidiel sa naša veta zmení na: “zaoberat sa proces pri vivoj softver jeho udrzba a rozhodovať o jeho zanik”.

2.2 N-gramy

Jedným z pohľadov na vetu je prístup “batoh slov” (ang. bag of words). Myšlienka spočíva v tom, že celý korpus pozostáva z jednotiek (slov), na ktoré sa dá pozerať individuálne. Každej jednotke potom vieme priradiť početnosť vrámci záznamu. Jeho aplikáciou však môžme stratiť cennú informáciu kontextu, v ktorom bolo slovo použité. Každé môže byť totiž súčasťou fráz, ktoré majú vo

vete väčšiu informačnú hodnotu. Strata kontextu môže spôsobovať problém pri celkovom výsledku klastrovania a klasifikácie. Ako vhodné riešenie sa ukazuje technika n-gramov.

N-gram je sekvencia po sebe idúcich elementov. Tieto elementy môžu byť písmená, slová, znaky, alebo iné. Zvyčajne bývajú zozbierané zo vzorových textov a neskôr bývajú aplikované na menšie dokumenty. Ak sú tvorené jedným elementom voláme ich unigramy, ak dvomi tak bigrami a podobne. Táto technika sa používa najmä pri spracovaní textu. Aplikujú sa pri tom pravdepodobnosti výskytu slova v závislosti od jeho predchodcu [4].

Jedným zo základných problémov, ktoré sa pri n-gramoch riešia je ich veľkosť. Čím viac elementov použijeme, tým náročnejší je výpočet. Keďže n-gramy sú vlastne ďalším parametrom pre náš model, pri veľkej hodnote premennej "n" výrazne rastie dimenzionalita, čo sa podpisuje pod zložitosť riešenia. Zvyčajne sa používajú dĺžky veľkosti $n < 5$ [4, 10]. Na príklade demonštrujeme použitie n-gramov dĺžky 2 vo vete: "Večer čítam dobrú knihu". Výsledky sú: "Večer čítam", "Čítam dobrú", "dobrú knihu". Tento, krok vieme aplikovať na akúkoľvek dĺžku "n" menšiu ako dĺžka vety.

Jedným z príkladov využitia n-gramov je BLEU metrika. Detailnejší pohľad na ňu nám poskytuje práca Bleu od Papineniho [10]. Hlavným cieľom práce bolo zistiť, ktorý z prekladov online slovníkov je najlepší. Najvyššie hodnotenie dostal preklad slovníka, ktorý bol najbližšie vzorovému prekladu. Celá technika je založená na porovnávaní n-gramov vo vetách. Hodnotenie začína postupne unigramami a končí pri n-gramoch dĺžky 4. Ako vzorový text na prípravu n-gramov sa používa preklad vety spracovaný človekom. Tento vzor sa potom porovnáva s prekladmi prekladačov.

Vezmime si príklad výpočtu hodnotenia pre unigramy a bigramy na nasledujúcich vetách. Ako vzorovú vetu máme: "Dnes večer vonku veľmi pršalo". A ďalšie dve testovacie vety: "Večer vonku príliš pršalo" a "Poobede vonku večer pršalo veľmi". V prípade unigramov sa sčíta výskyt každého jedného slova v testovanej vete vo vzore a predelí sa dĺžkou testovanej vety. Pri bigramoch je proces rovnaký, len so slovnými spojeniami dĺžky dva. Hodnotenie prvej vety pre unigrami je 3/4 a pre bigramy 1/3. Druhá veta zase dosiahne 4/5 v unigramoch a 0 v bigramoch.

Výsledok práce [10] ukazuje, že zväčšovaním dĺžky n-gramov použitých na porovnávanie sa dosahujú oveľa lepšie výsledky. V práci sa taktiež ukázali problémy, ktoré súvisia aj s našou prácou. Prvým problémom bola otázka, ako skombinovať rôzne pravdepodobnosti úspechu, ktoré vznikli pri zvyšovaní dĺžky n-gramov použitých na porovnávanie. Ako riešenie tohto problému sa ukázala aplikácia geometrického priemeru.

Ďalším problém je dĺžka vety. Z príkladu vyššie je zjavné, že kratšie testovacie vety, ktoré obsahujú všetky slová zo vzorovej vety dosahujú pri aplikovaní metódy výsledky okolo 1. Toto by skresľovalo celkový výsledok. Ako riešenie autori práce poskytujú aplikovanie faktoru penalizácie stručnosti (angl. brevity penalization - BP). Konečný vzorec pre BLEU metriku je teda takýto:

$$BP = \begin{cases} 1 & c > r \\ e^{1-\frac{r}{c}} & c \leq r \end{cases}$$

Kde c je dĺžka testovanej vety a r je dĺžka vzorovej vety.

$$BLUE = BP * exp(\sum_{n=1}^{N} w_n \log p_n)$$

Celkové BLEU hodnotenie pre vetu dostaneme vynásobením penalizácie stručnosti a geometrického priemeru vypočítaného z pravdepodobnosti pre jednotlivé dĺžky n-gramov. Z práce vyplýva, že použitie n-gramov má zmysel pri spracovaní krátkych textov, ktorými sú aj preklady slovníkov.

2.3 Výber čŕt

Črty (ang. features) sú časti dokumentu, ktoré ho charakterizujú. Ako vhodný príklad čŕt môžu byť početnosti samotných slov korpusu pre dokument. Tento prístup uplatňuje štatistika výskytu výrazu spomenutá nižšie. Avšak črty nemusia byť len slová, môžu to byť aj n-gramy, pri ktorých výbere sa využívajú štatistické prístupy [8].

Počet čŕt je veľmi dôležitý. Pri práci s veľkým množstvom dokumentov, v rámci spracovania textu, rastie časová a pamäťová zložitosť spojená s ré-

žiou algoritmov. Napríklad, ak by sme pri korpuse zostavenom zo stá tisícov záznamov, použili na výber čŕt štatistiku výskytu výrazu, nebolo by možné uskutočniť výpočet kvôli časovej zložitosti.

Kvôli týmto faktom má zmysel venovať sa problematike výberu čŕt (ang. feature selection) detailne. Navyše niektoré algoritmy vedia pracovať len s dátami, ktoré majú určitý tvar. Ďalej uvádzame príklady riešení.

2.3.1 Výskyt pojmu

Štatistika výskytu pojmu (ang. term frequency) nám dáva informáciu o tom, ako často sa vyskytuje pojem v dokumente. Toto číslo potom tvorí črtu dokumentu. Problém však môže nastať pri opakovaní pojmov, preto sa tento počet zvykne deliť celkovým počtom pojmov v dokumente.

$$tf(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Vzorec popisuje výskyt výrazu "i" v dokumente "j", kde "$n_{i,j}$" je početnosť pojmov "i" v dokumente "j" a v menovateli je sčítaný celkový počet pojmov v tom istom dokumente. Následný vektor reprezentujúci záznam v korpuse, je tvorený výskytom pojmu pre každé slovo v zázname.

Prístup používa všetky slová v rámci korpusu. Taktiež je potrebné pre každé jedno z nich vypočítať frekvenciu. Chýba tu redukcia dimenzionality, ktorá vytvára problém pri spracovaní veľkého množstva záznamov. Tento prístup podnietil vznik ďalších štatistík, ktoré ho redukujú na menšie množstvo čŕt.

2.3.2 Inverzný výskyt pojmu

Štatistika určuje váhu jednotlivých pojmov pre záznam s cieľom zistiť ich dôležitosť. Celá myšlienka je založená na tom, že čím častejšie sa nachádza pojem v dokumentoch tým menej je dôležitý. Napríklad ak máme spojku "a", ktorá sa nachádza vo väčšine slovenských viet priradíme jej samozrejme menšiu dôležitosť. Na druhej strane však doménové slová, ktoré sa vyskytujú v dokumentoch menej, budú mať vyššiu dôležitosť. Táto štatistika dokonca rieši problém s ojedinelými gramatickými chybami, ktoré sa v textoch vyskytujú naozaj

výnimočne. Vhodným nastavením hranice sa dajú tieto výrazy eliminovať.

Celá štatistika sa dá takto opísať [11]:

$$tfidf(j) = tf(j) * idf(j)$$

$$idf(j) = log(\frac{N}{df(j)})$$

Pričom váha pre príslušný výraz "tfidf" (j) sa vypočíta ako výskyt tohto výrazu modifikovaný o faktor inverzného výskytu tohto výrazu idf(j). Tento faktor sa vypočíta ako logaritmus podielu počtu všetkých výrazov a počtu dokumentov obsahujúcich tento výraz.

2.4 Diskusia

Pri práci s textami je nutnosť predspracovania zjavná. Neupravené texty by spôsobovali problémy vo fáze aplikovania algoritmov. Okrem pohľadu na vetu spôsobom batohu slov (ang. bag of words) je vhodné sa zamerať aj na kontext, v ktorom sa jednotlivé slová nachádzajú. N-gramy by tu mohli pomôcť. Bleu metrika je jasným príkladom úspešného použitia n-gramov na problémy spojené s krátkymi textami. Výsledky získané aplikáciou tejto metódy sa potom môžu využiť pri výbere čŕt.

Aplikáciou odstránenia gramatiky a interpunkcie však môžme stratiť niektoré dôležité informácie o odpovediach. Toto by sa však mohlo prejaviť na tých, ktoré pozostávajú z väčšieho počtu slov. V našom prípade sa zameriavame na kratšie.

3 Klasifikácia a klastrovanie

Klasifikácia je proces, kedy sa snažíme roztriediť položky do vopred určených skupín. O týchto skupinách máme potrebné informácie. To znamená, že disponujeme príkladmi členov jednotlivých skupín, nad ktorými sa vieme učiť. Takýto prístup sa nazýva učenie s učiteľom (ang. supervised learning problem). V prípade klastrovania toto nevieme. Dokonca niekedy nevieme ani počet tried. Ide o učenie bez učiteľa (unsupervised learning problem). V tejto kapitole sa budeme podrobne venovať jednotlivým prístupom. Cieľom je ich následné využitie pri práci s odpoveďami. Aplikáciou klasifikátorov vieme automaticky opravovať a klastrovaním by sme zobrazovali zhluky podobných odpovedí čo by reprezentovali správanie triedy.

3.1 Klasifikácia

Základným cieľom človeka, ktorý opravuje testy, je urobiť korekciu nielen správne, ale aj čo najefektívnejšie. Na začiatku učiteľ disponuje len odpoveďami študentov. Možno má v hlave nejakú správnu odpoveď, no nemusí ju v záznamoch nájsť presne v požadovanom znení. Pri niektorých odpovediach je to skoro až nemožné. Jedná sa napríklad o odpovede, kde sa požaduje opis. Príkladom môže byť otázka: "Čo je to fotosyntéza ?". Učiteľ vie, že správna odpoveď by mohla znieť nasledovne: "Je to biochemický proces, ktorý premieňa slnečné žiarenie na energiu". Nemusí však práve toto znenie nájsť. Pre takéto typy otázok, by sa nedala vopred vymenovať množina správnych odpovedí. Problém však môže nastať aj pri otázkach, kde sa očakáva názov. Ak je správny výsledok viac slovné pomenovanie, ktoré môže mať viacero synoným, alebo cudzo-jazyčných pomenovaní, vymenovanie všetkých možností je nemožné. Vtedy je potrebné odpovede prejsť manuálne.

Na začiatku nie sú záznamy anotované. Učiteľ však často disponuje odpoveďami študentov z minulých rokov. Takýto jav je častý, keďže predmety sa opakujú semestrálne s prevažne podobnými osnovami. Tieto záznamy môžu v

sebe zahŕňať správne, ale aj nesprávne príklady, nad ktorými sa môže klasifikačný algoritmus učiť. Úvaha je jednoduchá. Čím viac odpovedí je opravených tým je jednoduchšie automaticky vyhodnotiť tie ďalšie.

Manuálne ohodnotená vzorka odpovedí slúži ako učiteľ pre algoritmus. Následne nato sa jeho aplikáciou vieme dostať k triedam pre jednotlivé záznamy. V našom prípade pôjde vždy len o dve triedy a to zoznam potenciálne správnych odpovedí označených ako trieda 1 a zoznam potenciálne nesprávnych odpovedí označených ako trieda 0.

Prvá myšlienka je použitie len jedného klasifikačného algoritmu pri riešení problému hľadania správnych odpovedí. Samozrejme očakáva sa vhodne predspracovaný text a optimálny výber čŕt. Ako vhodné implementácie na výpočet klasifikátoru sa ponúkajú KNN (ang. k-nearest neighbours) a SVM (support vector machines) algoritmus opísané v tejto kapitole.

3.1.1 KNN algoritmus

Algoritmus k-najbližších susedov (k-nearest neighbours), ďalej už len KNN, patrí medzi najznámejšie algoritmy v oblasti klasifikácie, vďaka svojej jednoduchosti a efektivite. Na to, aby sme priradili dokumentu triedu, vezme "k" najbližších susedov tohto dokumentu, pozrieme sa na ich triedy a podľa toho určíme výsledok. Na porovnanie sa používa ich vzdialenosť. Jedným z príkladov môže byť kosínusová vzdialenosť, ktorú môžme zapísať vektorovým zápisom, kde "x" a "y" sú dokumenty:

$$dist(x, y) = \frac{x^T y}{|x||y|}$$

Algoritmus však dosahuje najlepšie výsledky na vybalancovaných datasetoch [16]. Pri opaku nastáva predikcia, kde prevláda početnejšia trieda korpusu, čo by mohlo mať nepriaznivý efekt.

3.1.2 SVM algoritmus

Support vector machines, ďalej už len SVM, patrí medzí ďalší klasifikačný algoritmus. Spolu s KNN ich radíme medzi klasifikačné algoritmy, kde nastáva učenie s učiteľom. To znamená, že na to, aby sme vedeli predikovať triedu

nového dokumentu je potrebná znalosť tried z predchádzajúcej trénovacej sady. Tá tvorí "učiteľa".

SVM klasifikátor sa snaží rozdeliť priestor jednou, alebo viacerými čiarami. Na jednoduché vysvetlenie predstavme si priestor, v ktorom sa nachádzajú len 2 triedy, ktoré chceme predikovať 0, 1. SVM sa pokúsi rozdeliť tento priestor jednou čiarou. Cieľom tejto čiary je oddeliť triedy. Následná predikcia sa uskutoční v tom istom priestore v závislosti od čiary. Najlepšia čiara je taká, ktorá maximalizuje vzdialenosť od najbližšieho reprezentanta triedy. Týmto prístupom sa minimalizuje chyba.

Problémom však nastane pri datasetoch, ktoré nie sú lineárne separovateľné. Príkladom je funkcia XOR. Ak ju nanesieme na 2D os, nie je možné nájsť jednu čiaru, ktorou by sa podarilo rozdeliť triedy. V takomto prípade sa priestor namapuje do n-D priestoru pričom sa využívajú kernel funkcie [17].

3.1.3 Metrika vyhodnotenia

Po úspešnom natrénovaní klasifikátoru prichádza na rad predikcia. Na to, aby sme vedeli povedať či je úspešná je vhodné ju ohodnotiť. Pre tento účel je dobré rozdeliť dataset na trénovaciu a testovaciu množinu v určitom pomere. Trénovanie klasifikátora nastáva na trénovacej množine, zatiaľ čo testovanie prebieha na testovacej (validačnej).

Po získaní výsledkov testovacej množiny je dobré na vyhodnotenie pokusu použiť metriku. Najprv si však musíme uvedomiť javy, aké môžu pri predikovaní nastať. Vezmime si náš príklad, kde pre záznam predikujeme či patrí, alebo nepatrí do nejakej triedy. Môžu nastať 4 prípady [14]:

- Predikujeme, že patrí a naozaj patrí (anglicky true possitive TP).
- Predikujeme, že patrí a v skutočnosti nepatrí (anglicky false positive FP).
- Predikujeme, že nepatrí a naozaj nepatrí (anglicky true negative TN).
- Predikujeme, že nepatrí a v skutočnosti patrí (anglicky false negative FN).

Následne na to chceme pre klasifikátor vypočítať hodnoty, ktoré ho charakterizujú. V štatistike sa pre tento účel používajú dva pojmy. Presnosť (ang.

precision), ktorá nám hovorí koľko je korektne predikovaných dokumentov (TP) voči všetkým dokumentom predikovaným ako správne:

$$precision = \frac{TP}{TP + FP}$$

Druhým pojmom je pokrytie (ang. recall), ktoré vyjadruje počet korektne predikovaných dokumentov (TP) voči skutočne správnym dokumentom:

$$recall = \frac{TP}{TP + FN}$$

Tieto dve hodnoty sa potom používajú pri výpočte F score, ktorá je relevantnou metrikou pri klasifikácii. Jej hodnota je ich harmonickým priemerom:

$$F = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

Náš prípad však zahŕňa predikciu viacerých tried (dvoch), preto potrebujeme vypočítať globálne precision a recall. Existujú dva možné prístupy [14]:

- Makro-priemer. Prístup spočíva vo vypočítaní priemeru z presnosti a pokrytia jednotlivých tried. Vzorce teda budú vyzerať:

$$MacroPrecision = \frac{P_0 + P_1}{2}$$

$$MacroRecall = \frac{R_0 + R_1}{2}$$

- Mikro-priemer. V tomto prípade sa v čitateli nasčítajú TP a v menovateli TP, FN a FP pre jednotlivé triedy.

$$MicroPrecision = \frac{TP_0 + TP_1}{TP_0 + TP_1 + FP_0 + FP_1}$$

$$MicroRecall = \frac{TP_0 + TP_1}{TP_0 + TP_1 + FN_0 + FN_1}$$

3.2 Klastrovanie

V druhej časti tejto kapitoly sa venujeme situáciám, pri ktorých by sme chceli o korpuse zistiť niečo viac ako len správne a nesprávne odpovede. Cieľom je teda transformovať korpus do zhlukov. Každý jeden bude obsahovať odpovede, ktoré medzi sebou súvisia.

Našou úlohou je správny výber algoritmu, ktorý by nám toto umožnil. V kapitole opíšeme dva spôsoby, k-means algoritmus a affinity propagation clustering. Obidva vytvoria klastre nad korpusom.

3.2.1 K-means algoritmus

Tento algoritmus patrí medzi najznámejšie v oblasti zhlukovania. Samotná realizácie je učením bez učiteľa. Teda je poskytnutý len dataset bez akejkoľvek anotácie. Narozdiel od klasifikátorov pri k-means sa vynecháva fáza učenia a prechádza sa rovno na predikciu.

Príslušnosť záznamu do jednotlivého zhluku sa určí na základe najkratšej vzdialenosti od centra. Centrá sú akýmisi reprezentantmi zhlukov. Na výpočet vzdialenosti sa najčastejšie používa Eklidová vzdialenosť. Výsledkom vzdialenosti medzi dvomi bodmi je postupné sčítanie druhých mocnín rozdielov medzi jednotlivými zložkami vektorov. Výsledok sa na konci odmocní. Celý vzorec vychádza z geometrických pravidiel:

$$dis(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Algoritmus ďalej pozostáva z troch krokov [7]:

1. Náhodná inicializácia centier zhlukov. Vo väčšine prípadov sa ako centrá inicializujú niektoré z dokumentov korpusu.

2. Priradenie zhlukov. V tomto kroku sa jednotlivým záznamom priradí taký zhluk, ktorého centrum je k nemu najbližšie.

3. Presun centier. V tomto kroku sa sa vypočíta priemerná pozícia pre jednotlivé záznamy v rámci centier. Tieto výsledky sú potom novými

centrami.

Tento proces potom pokračuje krokom 2 a opakuje sa. Počas cyklu sa počíta o koľko sa centrá posúvajú. V prípade, že posun centier je menší ako nejaká hranica celý proces je na konci. Algoritmus nebol vytvorený na prácu s textami. Je určený hlavne na zhlukovanie bodov v priestore. Kvôli tomuto faktu je potrebné text predspracovať a vhodne vybrať črty.

3.2.2 Affinity propagation

Tento algoritmus je založený na grafovom prehľadávaní korpusu. Ako vstup dostáva maticu s hodnotami s(i,k), ktoré hovoria o tom ako sú si dva body navzájom podobné . Jednotlivé záznamy si počas behu medzi sebou posielajú správy.

Algoritmus, narozdiel od k-means nepotrebuje ako vstup očakávaný počet zhlukov. Na začiatku sa každý záznam matice podobnosti považuje za potenciálne centrum zhluku. Neskôr sa uprednostňujú záznamy, ktoré majú vyššie hodnoty v matici. Na komunikáciu sa používajú 2 typy správ [2]:

1. Záruka (anglicky resposibility) r(i,k) je poslaná z bodu "i" potenciálnemu členovi svojho zhluku "k". Hodnota tejto správy reflektuje na koľko sa bod "k" hodí do klastra bodu "i", berúc do úvahy ostatné okolité body.

2. Dostupnosť (anglicky availability) a(i,k) je odpoveďou na záruku z bodu "k" pre bod "i" a hovorí o tom, ako veľmi je vhodné, aby bod "k" patril do klastra bodu "i". Na začiatku sú hodnoty týchto správ nastavené na 0.

Pri jednotlivých iteráciach sa hodnoty aktualizujú aplikovaním vzorcov [2]:

$$r(i,k) \leftarrow s(i,k) - max_{k's.t.k' \neq k}[a(i,k') + s(i,k')]$$

$$a(i,k) \leftarrow min\{0, r(i,k) + \sum_{i's.t.i' \neq \{i,k\}} max\{0, r(i',k)\}\}$$

Pre záruku sa hodnota správy vypočíta na základe podobností s cieľovým bodom a okolitými bodmi. Záruka tým pádom môže byť záporná. A tak sa pri odvodzovaní postupnosti pozeráme len na kladné záruky.

Pri affinity propagation netreba zabúdať, že úspešnosť závisí aj od vhodného výberu funkcie na určenie podobnosti dvoch bodov. Keďže pracujeme s vetami (reťazcami), ako vhodný kandidát sa javí Levensteinova vzdialenosť. Berme do úvahy dva reťazce "a" a "b". Jej hodnota je počet vymazaní, pridaní a nahradení potrebných na transformáciu reťazca "a" na reťazec "b".

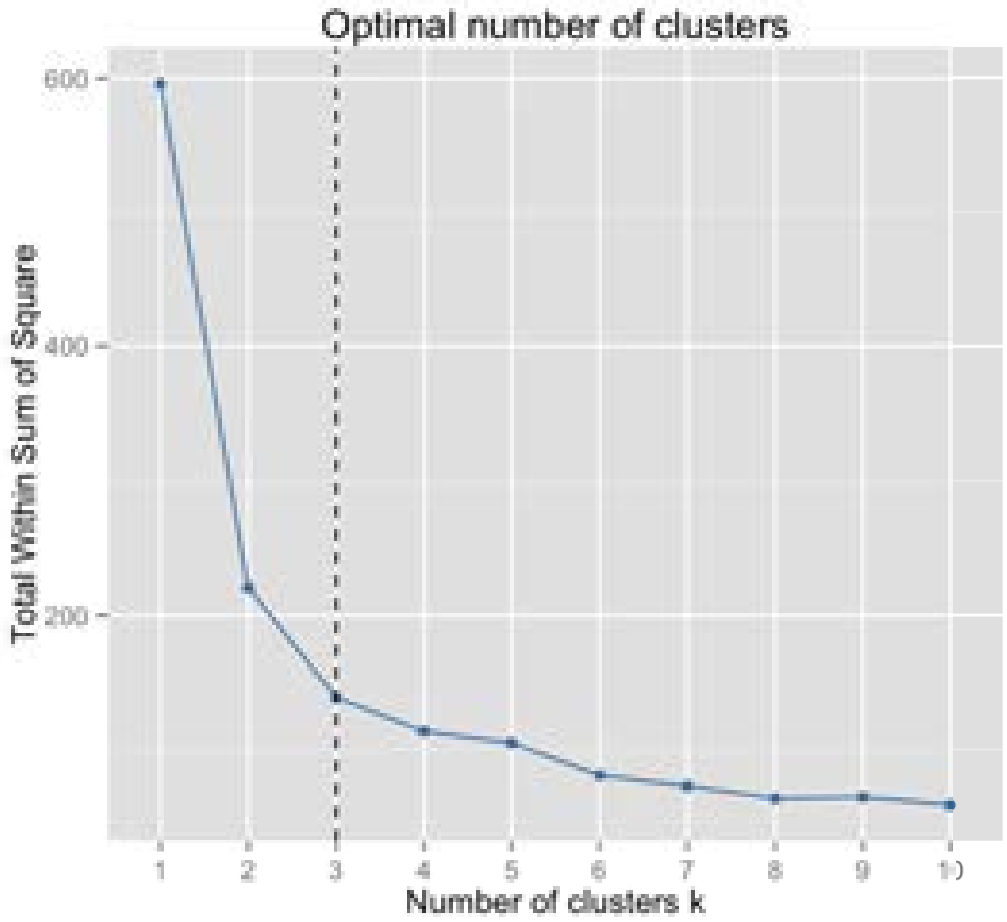
3.2.3 Problém určenia počtu klastrov

Algoritmus k-means patrí medzi jeden z najľahších a najefektívnejších spôsobov klastrovania. Jeho obmedzenie spočíva v potrebe poznať parameter počtu klastrov dopredu. Toto spôsobuje základný problém implementácie [5]. Jedno z riešení je spýtať sa koncového používateľa na očakávaný počet. V našom prípade však učiteľ nevie vopred odhadnúť koľko klastrov očakáva v odpovediach triedy. Tento prístup sa teda nedá realizovať.

Iné riešenie je takzvaná lakťová funkcia (ang. elbow function). V rámci k-means algoritmu môžeme pre aktuálne nastavenie centier počítať pre korpus nákladovú funkciu (ang. cost function) nasledovne [5]:

$$W(C, S) = \sum_{k=1}^{K} \sum_{i \in S_k} ||y_i - c_k||^2$$

Na vstup máme maticu centier C a maticu dokumentov S. Cez každú dimenziu vektora dokumentu Sk patriacemu klastru "k" počítame ich euklidovú vzdialenosť. Výsledky sčítame. Ak potom nanesieme na os "x" rastúci počet klastrov a na os "y" práve hodnotu tejto nákladovej funkcie výsledok bude vytvárať funkciu podobnú lakťu. Príklad zobrazuje obr. 3.1.



Obr. 3.1: *Príklad lakťovej funkcie*

Miesto v bode, keď sa klastre rovnajú trom sa nazýva lakeť a považuje sa za ideálneho kandidáta. Avšak nie vždy sa dá tak ľahko identifikovať. Niekedy je totiž zobrazenie funkcie plytké.

3.2.4 Metriky vyhodnotenia

Po získaní výsledkov klastrovania je dobré si výsledky overiť. Na túto úlohu existuje niekoľko metrík vyhodnotenia zhlukovania. Jedným z príkladov riešenie je rand skóre [3], ktorá sa dá vysvetliť nasledovne.

Vezmime si príklad. Pozrime sa na dva výsledky klastrovania, kde X = X1,

X2 ...Xr je zoznam klastrov pre prvý algoritmus a Y = Y1, Y2 .. Yr pre druhý. Na to, aby sme ich vedeli porovnať, musíme ich prechádzať po dvojiciach. Nech P je zoznam všetkých dvojíc P = P1, P2 ... Px Počas tohto procesu môžu nastať 4 javy, ktorých početnosť označíme nasledovne:

- "a" počet dvojíc z P takých, ktoré sú v rovnakom klastri pre X a Y.
- "b" počet dvojíc z P takých, ktoré sú v odlišnom klastri pre X aj pre Y.
- "c" počet dvojíc z P, takých, ktoré sú v rovnakom klastri pre X a odlišnom pre Y.
- "d" počet dvojíc z P, takých, ktoré sú v rovnakom klastri pre Y a odlišnom pre X.

Zo zápisu jasne vidno, že "a" a "b" vytvárajú zhodu medzi klastrami X a Y preto je výpočet nasledovný:

$$RI = \frac{a+b}{a+b+c+d}$$

V menovateli je počet všetkých možných párov, presne veľkosť množiny P. Rozsah indexu je z intervalu [0, 1].

Druhou možnosťou je adjusted rand index, ktorý sa snaží penalizovať omyly. Na výpočet sa používa nasledujúci vzorec [15]:

$$ARI = \frac{\binom{n}{2}(a+b) - [(a+d)(a+c)+(c+b)(d+b)]}{\binom{n}{2}^2 - [(a+d)(a+c)+(c+b)(d+b)]}$$

Označenie je rovnaké ako v prípade rand indexu. "n" je počet elemetov pre jeden klaster. Rozsah indexu je od [-1, 1].

3.2.5 Diskusia

Kapitola poskytuje pohľad na riešenia problémov spojených s klasifikáciou a zhlukovaním. V prípade klasifikácie je dôležité zvoliť si optimálny klasifikátor. V prípade klastrovania sme predstavili dva odlišné prístupy. Affinity

propagation poskytuje výhodu automatického určenia počtu klastrov, avšak je veľmi ovplyvnený funkciou podobnosti [2]. Na druhej strane máme k-means algoritmus, ktorý pre problémy klastrovania poskytuje lepšie výsledky [7].

V prípade affinity vymieňame efektivitu, spojenú s automatickým počtom klastrov, za časovú a pamäťovú zložitosť. Keďže chceme naše riešenie spojiť so systémami, ktoré pracujú v reálnom čase, tieto faktory môžu zohrávať dôležitú úlohu. Ak by bol výkon algoritmu pomalý malo by to zlý vplyv na reálne využitie.

4 Metóda štruktúrovania odpovedí študentov

V rámci nášho problému máme na vstupe zoznam odpovedí študentov, ktorý je neštruktúrovaný. Nevieme teda o ňom povedať žiadne kľúčové informácie potrebné pre učiteľa. Napríklad počet správnych odpovedí, alebo chyby vyskytujúce sa v odpovediach triedy. Situáciu je dobré vidieť na samotnom systéme ASQ. V prípade, že učiteľ prejde na slajd s otvorenou otázkou a nechá triedu odpovedať, na obrazovke vidí tento neštruktúrovaný zoznam

Naším cieľom je vytvoriť prehľadnú štruktúru v odpovediach. Chceme upustiť od jednoduchého zoznamu odpovedí, ktorý v sebe nenesie žiadnu informačnú hodnotu. Naša štruktúra by bola tvorená zhlukmi identifikovanými v korpuse. Tieto budú zobrazené vyučujúcemu cez aplikáciu prostredníctvom oddelených zoznamov pre jednotlivý klaster. Okrem tohto chceme ukázať riešenie spojené s automatickou opravou odpovedí, použitím natrénovaných klasifikátorov na už opravených odpovediach. Tieto zlepšenia by mohli pomôcť s procesom opravovania testov, no zároveň by vedeli poskytnúť prehľad o nepochopení učiva cez klastre. Ich kombinovanie s ASQom samotným by mohlo posunúť proces výučby.

Celá metóda pozostáva z niekoľkých krokov. Sú nimi predspracovanie, zhlukovanie, prípadná klasifikácia.

4.1 Predspracovanie textu

Odpovede študentov sú texty, ktoré pozostávajú z niekoľkých slov napísaných v jazyku. V našom prípade sa jedná o slovenčinu a doménu softvérového inžinierstva. Učivo podané študentom na tomto predmete je plné pojmov, ktoré môžu byť vysvetľované aj v angličtine. Aby sme vedeli prejsť do fázy klastrovania a klasifikácie je potrebné texty predspracovať [13].

Metóda zahŕňa opravu gramatiky. Aplikovaním korekcie sa snažíme vyhnúť problémom spojeným s nekonzistentnosťou textov. Vezmime si príklad dvoch študentov jeden z nich vloží odpoveď v znení “diagram aktivit” a druhý “diagram

aktivít". Odpovede sa zdajú byť rovnaké, avšak v druhej chýba dĺžeň. Keby ostali neopravené algoritmy klastrovania a klasifikácie by k nim pristupovali ako k odlišným. Po aplikovaní korekcie gramatiky budú obe odpovede v tvare "diagram aktivít".

Ďalší krok je spojený s extrakciou lémy slova. Opäť si vezmime dva príklady odpovedí. Študent 1: "aktivity diagramom", študent 2 "aktivity diagram". Opäť rovnaké odpovede avšak teraz je odlišnosť na úrovni použitia pádov slovenčiny. Aby sme situáciu vyriešili použijeme lemmatizáciu. Po jej aplikovaní dostaneme odpovede v tvare "diagram aktivita".

Nasledujúci krok je spojený s písaním písmen ïä "y". Problém je podrobnejšie rozpísaný v kapitole 3. Vezmime si dve odpovede "aktyvity diagram" a "aktivity diagram". Obsahová stránka odpovedí je totožná. Obaja opýtaní mysleli ten istý druh diagramu, avšak študent číslo 2 odpovedal gramaticky nesprávne. Keďže pracujeme s odpoveďami domény softvérového inžinierstva predmetu princípy softvérového inžinierstva na našej fakulte, gramatické chyby na úrovni písania ïä "y"nás nezaujímajú. To znamená, že môžme použiť nahradenie "y"za ï". Výsledok odpovedí bude teda "aktiviti diagram".

Ďalej je potrebné upraviť detaily spojené s textami, týkajúcimi sa písania veľkých a malých písmen. Táto časť taktiež súvisí s gramatikou slovenčiny. Aby sme predišli nekonzistencii textov zmeníme všetky veľké písmená na malé. Vezmime si dve odpovede "Aktivity diagram" a "aktivity diagram". Po aplikácii dostaneme "aktivity diagram".

Pokračujeme krokom n-gramov. Ich vytvorením vieme zachytiť kontext odpovede. Ako maximálnu dĺžku gramu sme vybrali číslo 4 na základe prác [4, 10], ktoré to odporúčajú. Aplikáciu demonštrujeme na príklade odpovede: "Miera splnenia požiadaviek na softvér". Použitím prístupu "batoh slov" získame slová, ktorými je veta tvorená. Ak však aplikujeme n-gramy výsledok je nasledovný:

- N rovné 1: Miera, splnenia, požiadaviek, na, softvér.
- N rovné 2: Miera splnenia, splnenia požiadaviek, požiadaviek na, na softvér.
- rovné 3: Miera splnenia požiadaviek, splnenia požiadaviek na, požiadaviek na softvér.

- N rovné 4: Miera splnenia požiadaviek na, splnenia požiadaviek na softvér.

Každý člen z každej skupiny bude použitý ako črta.

Posledná časť kapitoly je spojená s extrakciou čŕt. Kapitola 2.3 poskytuje prehľad prístupov v tejto oblasti. Pre náš problém je vhodné použitie štatistiky výskytu pojmu. Dôvod je taký, že inverzný výskyt pojmu penalizuje často opakujúce sa slová v korpuse. Toto by mohlo viesť k strate dôležitých informácii v korpuse pri krátkych odpovediach. Ak by sme mali v datasete veľakrát odpoveď "aktivity diagram", inverzný výskyt pojmu by n-gramy [aktivity, diagram, aktivity diagram] penalizoval natoľko, že by mohli byť vylúčené. Aplikovaním výskytu pojmu získame maticu reprezentujúcu korpus pre všetky n-gramy.

Našu metódu doplňujeme o vymazávanie špeciálnych n-gramov z korpusu. Jedná sa o n-gramy, ktoré sa nachádzajú v otázke, na ktorú sa odpoveď viaže. Vychádzame z predpokladu, že sa v otázke nenachádza ani správna ani nesprávna odpoveď pre učiteľa.

4.2 Klastrovanie odpovedí

Ďalším dôležitým celkom je klastrovanie odpovedí. Naša metóda poskytuje dva prístupy. V prvom z nich vyučujúci pozná počet klastrov, od ktorých očakáva, že sa vyskytnú v datasete odpovedí. V tomto prípade sa vezme matica reprezentujúca dataset a na výpočet klastrov sa použije jeden klastrovací algoritmus. Každej odpovedi je priradený práve jeden klaster a každý klaster musí mať aspoň jednu odpoveď.

Druhý prístup sa odvíja od predpokladu, že vyučujúci nevie odhadnúť očakávaný počet klastrov pre otázku. Vezmime si prípad, kedy sa v teste vyskytne otázka: "Aký diagram sa používa na modelovanie biznis procesov". Učiteľ nevie povedať všetky druhy odpovedí, ktoré sa môžu vyskytnúť. V takomto prípade nemôžme očakávať, že zadá počet klastrov. Postup metódy je predikcia na základe obsahu datasetu. Na túto úlohu nám poslúži affinity propagation, klastrovací algoritmus založený na grafovom prehľadávaní. Affinity pri výpočte považuje každý dokument za potenciálny klaster [2]. Tento prístup

by mohol narušiť výsledky predpovedaním nadmerného množstva zhlukov. Aby sme predišli tejto situácii klastre s počtom odpovedí 1 spájame do jedného. Počet klastrov z výsledku použijeme ako parameter pre algoritmus k means. Ten sa následne snaží klastrovať dáta do toho istého počtu klastrov. Týmto prístupom sme vyriešili problém s určením počtu klastrov pre algoritmus k-means

4.3 Klasifikácia

Posledný celok metódy tvorí klasifikácia. Tá je však aplikovaná len v prípade, že učiteľ disponuje opravenými odpoveďami študentov z minulých rokov, kde sa otázka vyskytovala. Ak sa týmto datasetom nedisponuje, klasifikácia nie je možná. Keď dataset máme, je dôležité, aby sa jeho záznamy predspracovali. Na tento proces sa použije metóda opísaná v podkapitole 4.1.

Následne na to je matica, ktorá je výstup predchádzajúceho procesu použitá ako trénovacia sada pre klasifikátor. Ako vhodné riešenia pre klasifikáciu krátkych textov v slovenskom jazyku považujeme klasifikátor SVM. Po jeho aplikovaní máme dáta opravené. Tieto výsledky môžu slúžiť ako odporúčanie pre učiteľa pri opravovaní, keďže boli natrénované na jeho výstupe opravovania.

4.4 Diskusia

Kapitola poskytla postup metódy, ktorú sme vyvinuli za účelom štruktúrovania odpovedí študentov na otázky. Poskytli sme niekoľko krokov ktorými je tvorená. Je nutné si však uvedomiť, že niektoré konfigurácie sa viazali na slovenský jazyk a doménu softvérového inžinierstva. Príkladom môže byť náhrada písania ïä "y". Ak by sme chceli klastrovať odpovede týkajúce sa gramatiky slovenského jazyka, museli by sme tento krok vynechať, alebo nahradiť. Existuje mnoho nástrojov, ktoré sa venujú tejto problematike v inom jazyku [6, 9, 12]. Výsledná metóda môže poskytnúť klastre s chybami. Toto však nemusí byť problém ak sa z výsledku dá vyčítať správanie triedy.

V oblasti klasifikácie sú výsledky výrazne ovplyvnené trénovacou sadou. Na to, aby boli čo najlepšie je nutné mať čo najviac záznamov z minulých rokov.

5 Overenie

V tejto kapitole sa venujeme overovaniu metódy riešenia, ktorú sme navrhli. Kapitola je rozdelená do niekoľkých častí. V úvode sa venujeme datasetom, s ktorými počas overovania pracujeme. Druhá časť pokrýva experimenty, ktorými sme datasety získali. V tretej časti popisujeme implementačné detaily spojené s časťou predspracovania. Ďalej pokračujeme klasifikáciou. V tejto časti popisujeme vybrané implementačné riešenia, metriku overenia a nakoniec poskytujeme prehľad výsledkov na datasete. Na záver je klastrovanie, v ktorom je obsah štruktúrovaný rovnako ako pri klasifikácii.

5.1 Datasety

Pracovali sme s dvomi datasetmi. Prvý bol zozbieraný za posledných 5 rokov pri vstupných testoch v predmete Princípy softvérového inžinierstva. Študenti na začiatku každého cvičenia odpovedali na dve otvorené otázky do Akademického informačného systému (AIS). Odpovede mohli byť ľubovoľnej dĺžky. AIS ponúka možnosť testovania študentov. Testy je potrebné vytvoriť v systéme. Okrem samotného zadávania otázok je možné v systéme zadať aj správnu odpoveď, ktorá bude študentovi uznaná v prípade, že vyučujúci zvolí možnosť automatickej opravy testov. Z tohoto testovania máme odpovede, ktoré boli vyhodnotené vyučujúcimi ručne, čiže o každej vieme či bola správna, alebo nie. Tabuľka 5.1 poskytuje ukážku záznamov.

Obmedzenie tohto datasetu spočíva v početnosti odpovedí pre jednotlivé otázky. Zvykli sa totiž meniť a neopakovať. Priemerne máme 40 odpovedí pre jednu otázku.

Druhý dataset sme získali použitím systému ASQ na prednáškach z predmetu Princípy softvérového inžinierstva v ak. roku 2016/17. ASQ je webová aplikácia, ktorá pomocou online prezentácií pomáha pedagógom a študentom počas prednášky. Študenti sa pripoja svojimi tabletami, počítačmi, alebo smartfónmi na server, kde sú im počas prednášky sprístupnené slajdy prezentácie [18].

Tabuľka 5.1: *Ukážka záznamu z datasetu FIIT STU Bratislava.*

cvičenie	**test**	**otázka**	**odpoveď**	**body**
Po 16.00-17.50 -1.40 (PU1)	Test na cviceni cislo 3	Aký diagram UML používame pri modelovaní biznis procesov?	zelený	0
Po 16.00-17.50 -1.40 (PU1)	Test na cviceni cislo 3	Aký diagram UML používame pri modelovaní biznis procesov?	aktivity diagram	0
Po 14.00-15.50 -1.40 (PU1)	Test na cviceni cislo 3	Aký diagram UML používame pri modelovaní biznis procesov?	stavovy diagram	0
Po 14.00-15.50 -1.40 (PU1)	Test na cviceni cislo 3	Aký diagram UML používame pri modelovaní biznis procesov?	diagram činností	0

Ide o spôsob učenia sa pomocou otázok a odpovedí. Nástroj taktiež umožňuje vytváranie testov v reálnom čase. Ich použitím si môže pedagóg overiť pochopenie prednášanej témy. Automatická oprava otvorených otázok je v ASQu riešená rovnako ako v systéme AIS porovnávaním s vopred definovanou správnou odpoveďou. Prezentácie sme implementovali použitím webových technológii a prezentačného rámca impress jazyku JavaScrip. V nasledujúcej podkapitole opisujeme samotný experiment.

5.2 Experimenty

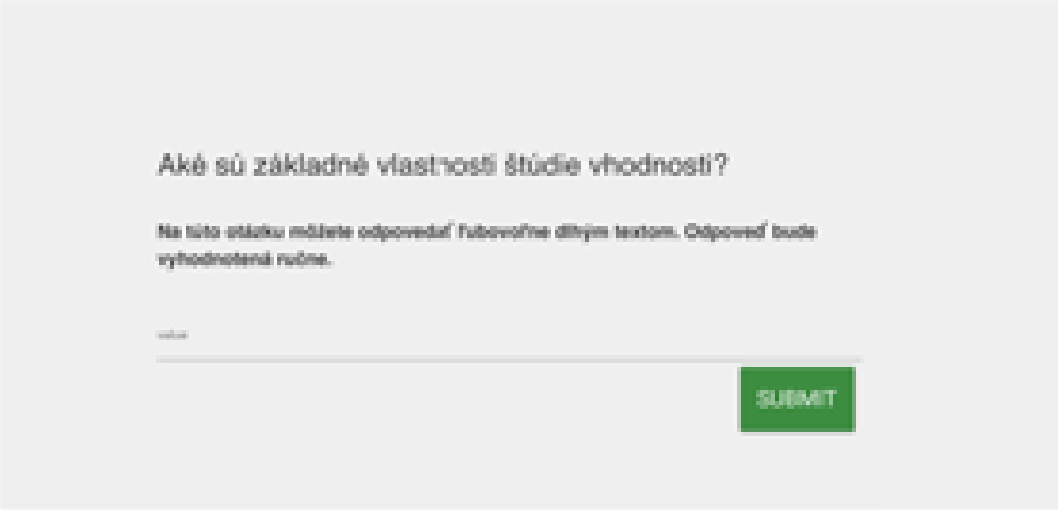
V tejto časti práce ponúkame podrobnosti o experimentoch. Ich realizáciou sme získali dáta, ktoré sme použili pri overovaní metódy. Jednalo sa o dva experimenty. Prvým boli prednášky v systéme ASQ počas predmetu princípy softvérového inžinierstva. Druhý experiment bol tvorený sedeniami s expertami, ktorých úlohou bolo klastrovanie odpovedí.

5.2.1 Experiment ASQ

Systém ASQ sme použili v rámci predmetu Princípy softvérového inžinierstva v akademickom roku 2016/17. Bežne bolo na prednáške viac ako 100 aktívnych používateľov systému. Počas letného semestra sa nám podarilo zozbierať odpo-

vede na 8 prezentácii, kde v každej bolo asi 5 otázok. Každá z nich obsahuje okolo 60 záznamov študentov. Do datasetu sme zahrnuli len otázky s viac ako štyridsiatimi odpoveďami a s priemernou dĺžkou odpovede menšou ako 6 slov. Výsledkom bolo 18 otázok s viac ako 900 odpoveďami. Z datasetu sme neodstraňovali žiadne odpovede (ani prázdne). Cieľom bolo odzrkadliť reálnu situáciu v triede počas prednášky. Študenti sa do systému ASQ prihlasovali cez svoje AIS konto, ktoré obsahuje ich akademické informácie (meno, priezvisko). Prihlasovanie nebolo povinné. To znamená, že ak sa študent prihlásil vedeli sme jeho odpovede neskôr vyhľadať. Pre náš experiment však tento údaj nie je potrebný. Toto, kvázi podpisovanie, simulovalo situáciu reálneho testu. Študenti boli motivovaní k prihlasovaniu bonusovými bodmi z predmetu.

Postup na prednáške bol nasledovný. Prednášajúci vysvetľoval učivo, pričom sa ASQ nepoužíval. V prípade, že chcel preveriť či poslucháreň rozumie prešiel do systému na slajd s otázkou. Toto prepnutie mohol realizovať len prednášajúci na svojom zariadení. Študentom sa zobrazila otázka obdobná tej na obrázku 5.1. Následne na to dostali čas na odpovedanie, v priemere 2 minúty. Po prepnutí na ďalší slajd študenti nemohli ďalej odpovedať. Otázky sa týkali len domény softvérového inžinierstva.

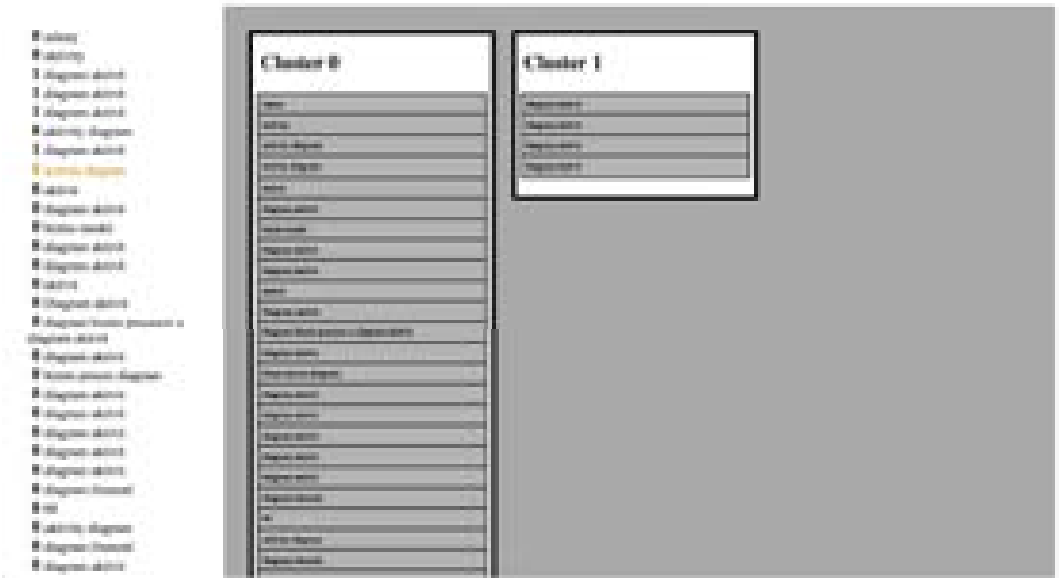


Obr. 5.1: *Ukážka testovej otázky v systéme ASQ.*

5.2.2 Experiment experti

Keďže dáta získané experimentom s ASQom sú doménovou záležitosťou oblasti softvérového inžinierstva, nechali sme odpovede prejsť expertami. Jednalo sa o cvičiacich tohto predmetu. Každý z nich klastroval odpovede študentov zvlášť. Ich úlohou bolo roztriediť zoznam odpovedí do niekoľkých skupín. Tieto skupiny by mali vystihovať správanie triedy. To znamená odpovede, ktoré sú si navzájom podobné budú zahrnuté do jedného klastra. Táto podobnosť nemusela byť len po stránke slov, mala zahŕňať aj celkový význam vety.

Každé sedenie trvalo asi hodinu a podarilo sa prejsť 9 otázok z experimentu s ASQom. Dokopy sa anotovalo viac ako 500 odpovedí. Experti nedostali informáciu o očakávanom počte klastrov. Ako nástroj na tento proces sme vyvinuli jednoduchú aplikáciu, ktorá umožňovala prechádzanie medzi odpoveďami za použitia klávesnice a následné priradenie do klastra. Každá zmena sa postupne vizualizovala, tým pádom bol poskytnutý jasný prehľad o rozložení odpovedí. Obrázok 5.2 poskytuje ukážku rozhrania.



Obr. 5.2: *Ukážka nástroja na manuálne klastrovanie, použitom pri experimente.*

5.3 Implementácia predspracovania textu

Pri implementácii tejto časti sme okrem vlastných funkcií použili externé webové služby, ktoré poskytuje naša fakulta. Umožňujú presnejšie spracovanie textov.

V rámci opravy gramatiky sme použili nástroj na doplňovanie diakritiky, ktorý slúži na korekciu slovenských textov, ktoré ju neobsahujú [13]. Riešenie je implementované prostredníctvom REST rozhrania. Get dopyt na server obsahuje v URL parametre, ktoré sú vlastne textom, kde treba gramatiku doplniť. Následná odpoveď zo serveru obsahuje v tele správne opravenú vetu.

Ako vhodný nástroj na úlohu lematizácie v rámci slovenského jazyka je lematizátor, ktorý využíva morfologickú databázu slovných tvarov z Jazykovedného ústavu Ľudovíta Štúra. Používame implementáciu Róberta Horvátha FIIT STU. [13]. Nástroj berie ako vstup vetu v Slovenskom jazyku, ktorá je súčasťou HTTP POST požiadavky a ako návratovú hodnotu získame odpoveď s lematizovanou vetou. Nástroj funguje úspešne len so slovami, ktoré spĺňajú gramatické pravidlá slovenského jazyka. Je preto nutné zachovať poradie spracovania. V prípade, že je na vstupe nekorektné slovo výstupom je to isté slovo bez zmeny

5.4 Klasifikácia

Pred klasifikáciou AIS datasetu sme korpus spracovali použitím metódy predspracovania textu z podkapitoly 4.1. Pri implementácii algoritmov využívame programovací jazyk Python a knižnice Pandas a Scikit-learn. Pre SVM používame SGDCClassifier s nasledujúcimi parametrami[1]:

- "Loss": Parameter stratovej funkcie. V našom prípade je nastavený na "hinge", ktorý vracia štardardnú implementáciu lineárneho algoritmu SVM.
- "Penalty": Parameter regularizácie. Používame nastavenie "l2", opäť štandardne určené pre SVM.
- "Alpha": Parameter, ktorým sa prenásobuje "penalty". Nastavenie na 0.0001.

[1] http://scikit-learn.org

- "N_iter": Počet iterácii algoritmu. Nastavenie na 5.

Pre KNN taktiež používame knižnicu Scikit-learn. Konkrétne klasifikátor KNeighboursClassifier s nasledujúcimi parametrami[2]:

- "Weight": Parameter váhovania susedov. V našom prípade nastavený na "uniform".
- "N_neighbours": Parameter vyjadruje počet susedov potrebných na výpočet. Nastavenie na 5.
- "P": Parameter výpočtu vzdialenosti susedov. Nastavenie na 2, ktoré zodpovedá Euklidovskej vzdialenosti.

Výstupom je jupyter notebook s funkčnými klasifikátormi. Na vstup stačí poskytnúť dataset odpovedí.

Aby sme vedeli správne ohodnotiť úspešnosť klasifikácie je nutné rozdeliť si množinu odpovedí na trénovaciu a testovaciu, v pomere 80 : 20. Takýmto prístupom sa vyhneme pretrénovaniu. V našom prípade sa snažíme natrénovať klasifikátor pre každú sadu odpovedí zvlášť. To znamená, že proces ohodnotenia opakujeme osobitne. Hodnotíme teda výsledky klasifikátorov SVM a KNN. Okrem toho porovnávame rôzne možnosti konfigurácii ako napríklad:

- Použitie lematizácie. Dáta korpusu sú počas spracovania lematizované. Keďže používame korekciu gramatiky, môže tento krok dopomôcť.
- Použitie inverzného výskytu výrazu. Často opakujúce sa výrazy v korpuse sú penalizované použitím tejto štatistiky. Krok môže mať nepriaznivé účinky na odpovede, ktoré pozostávajú z malého počtu slov.

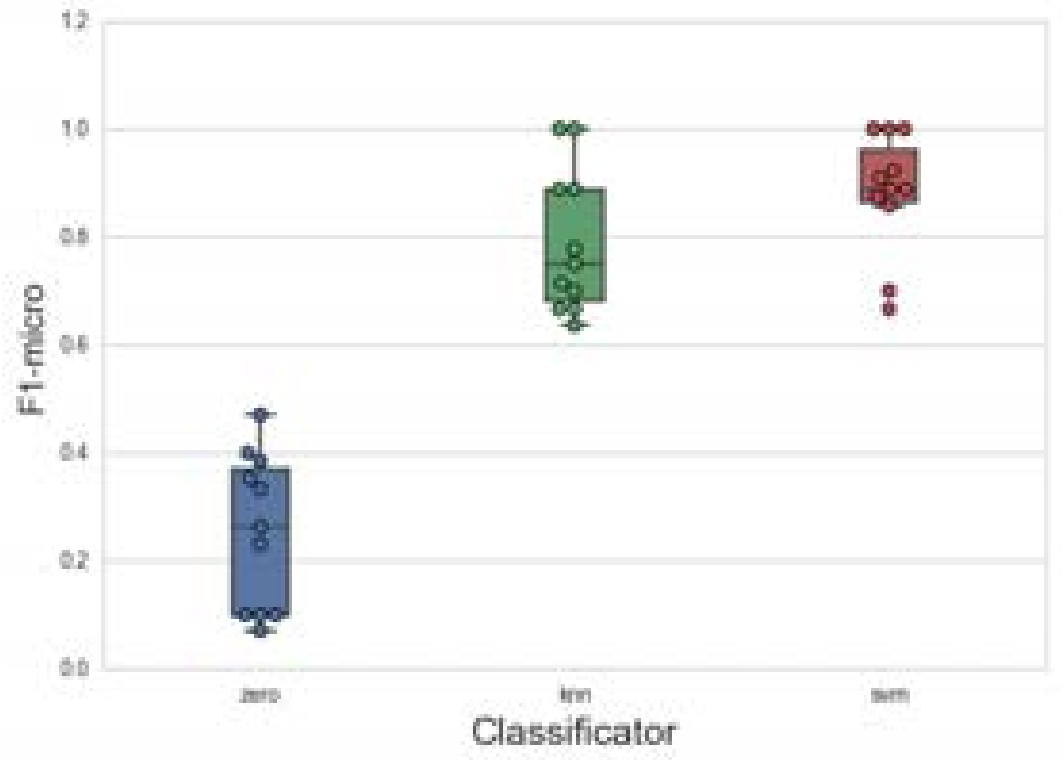
Parametre použité pri spracovaní sú totožné pri obidvoch datasetoch a sú vybraté bez ohľadu na testovací dataset. Posledným krokom overenia klasifikácie ja výpočet f score. Pre túto úlohu používame implementáciu f1_score knižnice scikit-learnu jazyku python.

[2]http://scikit-learn.org

5.4.1 Výsledky

V tejto časti poskytujeme prehľad výsledkov klasifikátorov na datasete zo systému AIS. Keďže otázky v ňom majú častokrát málo odpovedí, vybrali sme len niektoré z nich. Celkovo výsledky prezentujeme v dvoch častiach.

Prvá časť ukazuje výsledky na otázky, ktoré majú aspoň 40 odpovedí s mediánom počtu slov 10. Ohraničenie dĺžky filtruje dataset s otázkami, ktoré očakávajú kratšiu odpoveď. Jedná sa o 12 otázok s viac ako 580 odpoveďami. Obrázok 5 poskytuje prehľad výsledkov. Aby sme ukázali efektivitu klasifikácie oba výsledky porovnávame s takzvaným nulovým klasifikátorom. Ide o taký, ktorý pre každý dokument testovacej sady predikuje triedu 0.

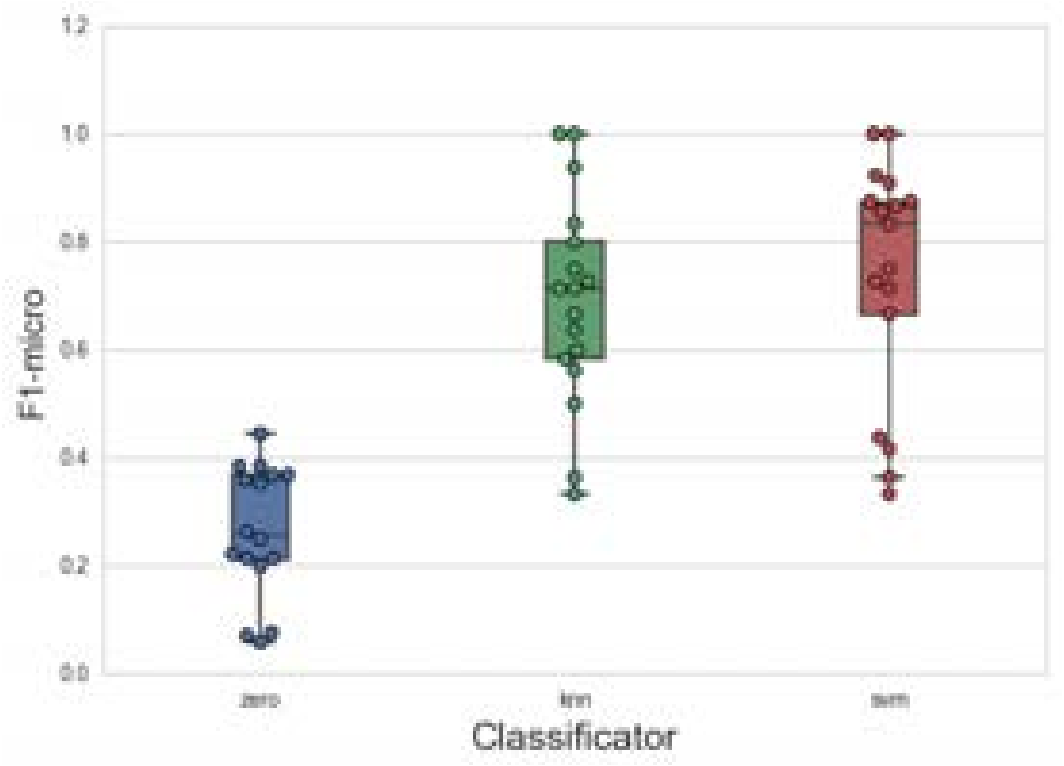


Obr. 5.3: *Výsledky klasifikácie pre otázky s viac ako 39 odpoveďami s mediánom odpovedí aspoň 6.*

Z obrázka 5.3 možno vidieť, že pre datasety s kratšími odpoveďami, kde je menej záznamov, je SVM úspešnejší. Algoritmy pri výpočte použili gramatickú

korekciu, lematizáciu a výber čŕt bez použitia IDF.

Druhá časť pracuje s odpoveďami, ktoré mali aspoň 51 a viac odpovedí a dĺžka odpovede mohla byť neobmedzená. Cieľom bolo zistiť úspešnosť klasifikátora pre rôznu dĺžku odpovedí. V tomto prípade sa počet otázok rovnal 16 dokopy s 1100 odpoveďami. Obrázok 5.4 poskytuje prehľad o výsledkoch. Konfigurácia algoritmov zostala nezmenená.



Obr. 5.4: *Výsledky klasifikácie pre otázky s viac ako 50 odpoveďami a s rôznou dĺžkou odpovede*

Môžeme si všimnúť, že úspešnosť SVM sa o niečo zhoršila. Zvýšenie dĺžky odpovede má nepriaznivý vplyv.

5.5 Klastrovanie

Na klastrovanie podobne ako na implementáciu klasifikátorov používame Python s knižnicami scikit-learnu. Prvým algoritmom je Affinity propagation. Na

realizáciu algoritmu sme použili AffinityPropagation s parametrami:

- "Damping": Parameter tlmiaceho faktoru. Používame ho, aby sme predišli osciláciám výsledku spojenými s prestrelením riešenia [19]. Tento faktor sa využíva pri výpočte správ medzi dokumentmi. Nastavenie na 0.5.
- "Max_iter": Parameter maximálneho počtu iterácii. Nastavenie na 200.

Druhým algoritmom klastrovania je k-means. Spracovanie korpusu bolo rovnaké ako pri affinity s rozšírením o výber čŕt. Celý postup klastrovania prebiehal v súlade s metódou opísanou v podkapitole 4.2. Následne aplikujeme implementáciu KMeans zo scikit-learnu s parametrami:

- "N_clusters": Parameter počtu klastrov. Tento algoritmus ho potrebuje pred výpočtom. Riešeniu tohto problému sa venujeme v ďalšej podkapitole.
- "Max_iter": Parameter počtu iterácii. Nastavenie na 300.
- "Init": Parameter metódy na inicializáciu centier. Nastavenie na "random", ktorý náhodne vyberie reprezentantov z datasetu.

V prípade kombinovaného prístupu klastrovania sme použili obidva algoritmy spoločne.

Ďalej nasleduje overenie metódy klastrovania. Ako pilotný test sme ručne klastrovali odpovede študentov získané experimentom z ASQu. Aplikovaním tejto metódy sme získali anotované dáta, ktoré sme mohli porovnať so získanými výsledkami algoritmov.

Tento prístup však nie je postačujúci. Aby sme mohli reálne overiť úspešnosť algoritmu potrebovali sme anotácie odborníkov. Keďže sa snažíme, aby algoritmus predpovedal klastre s cieľom poskytnúť informatívnu spätnú väzbu práve odborníkom. Na toto slúžil experiment s expertmi. Ďalej pokračujeme výsledkami.

5.5.1 Výsledky

Experti počas experimentu klastrovali odpovede študentov na otázky zozbierané systémom ASQ. Nasledujúca tabuľka 5.2 ukazuje vybraný počet klastrov pre jednotlivé otázky expertami a naším algoritmom.

Tabuľka 5.2: *Počty klastrov vyhotovené expertami a algoritmom pre jednotlivé otázky.*

	q0	**q1**	**q2**	**q3**	**q4**	**q5**	**q6**	**q7**	**q8**
Expert 1	4	6	4	4	4	6	7	9	6
Expert 2	5	3	4	3	4	6	4	3	3
Expert 3	4	5	3	3	5	5	6	7	5
Algorithm	8	4	11	8	4	7	7	12	7

Pre úplnosť tabuľka 5.3 poskytuje znenie otázok aj s priemernou dĺžkou odpovede pre každú z nich.

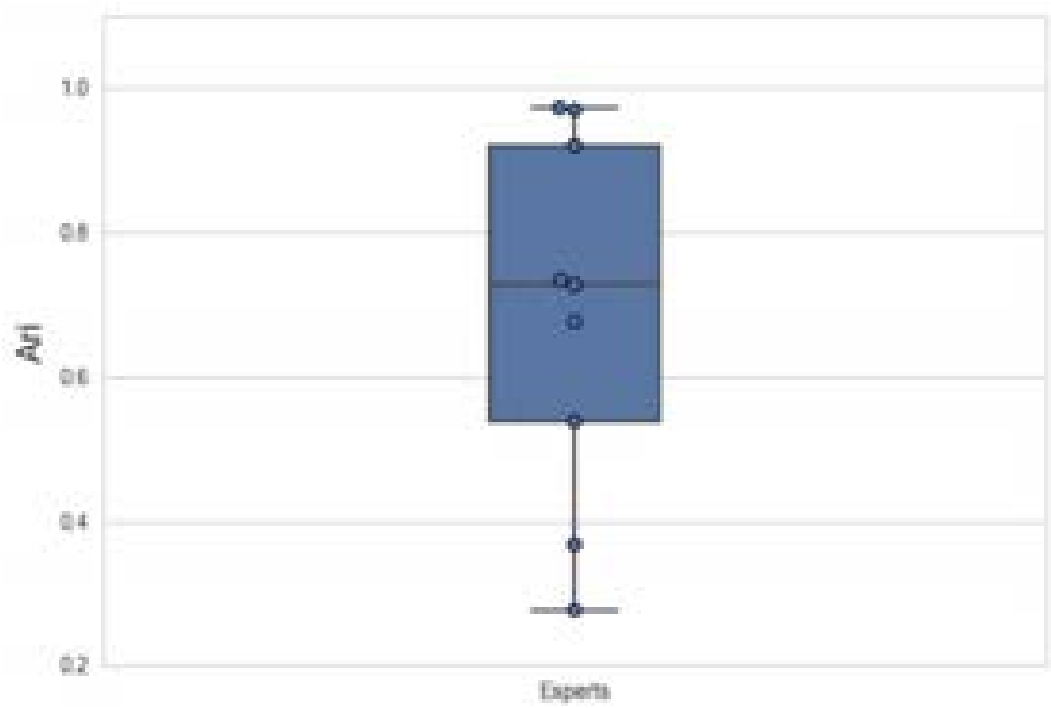
Tabuľka 5.3: *Znenie otázok experimentu s priemernou dĺžkou odpovede.*

Označeni	**Znenie**	**Priemerná odpoveď (slová)**
q0	Aké sú základné vlastnosti štúdie vhodnosti?	4
q1	Aký vzťah v diagrame prípadov použitia vyjadruje plná čiara bez šípky alebo s jednoduchou šípkou?	1
q2	Ktorá etapa životného cyklu softvéru je spravidla najnákladnejšia?	1
q3	Ako je definovaná kvalita softvéru?	4
q4	V ktorej fáze vývoja softvéru vytvárame diagramy prípadov použitia?	1
q5	Aký diagram UML používame pri modelovaní biznis procesov?	2
q6	Uveďte názov elementu diagramu aktivít na obrázku. (koncová aktivita)	2
q7	Uveďte názov elementu diagramu aktivít na obrázku. (fork, join)	2
q8	Uveďte názov elementu diagramu aktivít na obrázku. (decision, merge)	2

Môžeme si všimnúť, že aj samotní experti majú problém zhodnúť sa s počtom klastrov. Na druhej strane vidíme, že naše riešenie má tendenciu predpovedať viac klastrov ako je očakávané. Toto vedie k tomu, že niektoré klastre môžu byť rozdelené. To však nemusí byť problém.

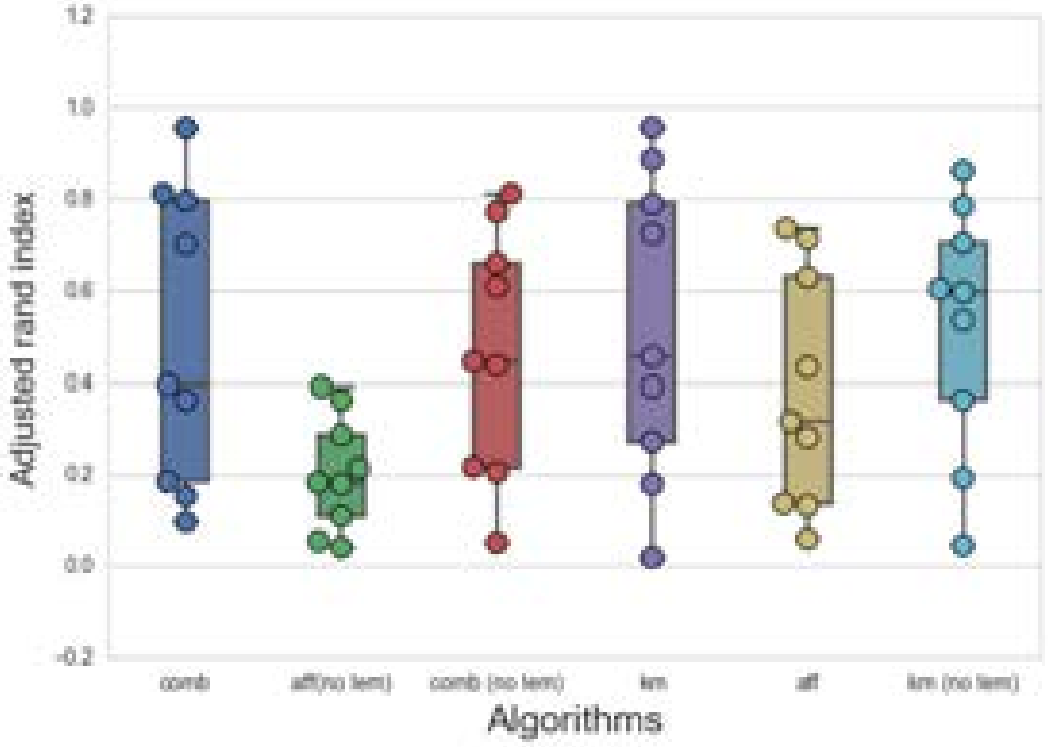
V druhej časti sa venujeme vyhodnoteniu úspešnosti klastrovania medzi expertmi navzájom. Pre tento účel používame metriku ARI. Obrázok 5.5

poskytuje prehľad o výsledkoch.



Obr. 5.5: *Boxplot ari expertov pre jednotlivé otázky.*

Následne na to sme vypočítali ARI medzi jednotlivými expertmi a nami vytvoreným algoritmom pre každú otázku. Na záver počítame priemer týchto hodnôt. Obrázok 5.6 poskytuje prehľad.



Obr. 5.6: *Priemerné ari pre rôzne konfigurácie klastrovania.*

Skratka "comb" označuje kombinovaný algoritmus, "km" je k-means a aff je "affinity propagation". Slovom "no lem" označujeme varianty bez použitia lematizácie.

Môžme si všimnúť, že lematizácia má pozitívny vplyv na klastrovanie textu. Najlepšie výsledky dosahujeme použitím k-means samotného, avšak v tomto prípade sme definovali počet klastrov rovný ideálnemu prípadu (čiže počtu zadaného expertom). Kombinovaný prístup dosahuje podobné výsledky avšak s väčším rozptylom. Na druhej strane však verzia bez lematizácie dosahuje stabilnejšie výsledky. V tomto prípade sa snažíme počet klastrov predikovať.

5.5.2 Vizualizácia klastrov

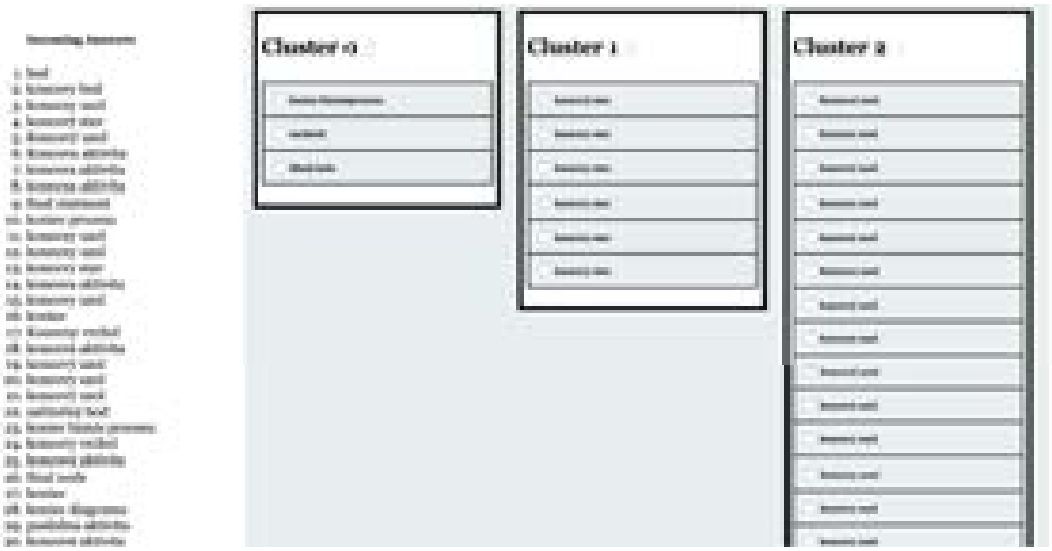
Po získaní výsledkov klastrovania je potrebné ich nejako zobraziť vyučujúcemu. Doterajší prístup systému ASQ jednoducho zobral všetky odpovede a zobrazil

ich ako zoznam po sebe idúcich záznamov, v poradí v akom boli do systému poslané. Výsledok neposkytoval žiadnu štruktúru a neprezrádzal nič vzoroch o v odpovedaní.

Naša metóda zobrazuje výsledky do prehľadne oddelených krabičiek. Každá z nich obsahuje zoznam odpovedí pre daný klaster. Realizácia je cez webovú aplikáciu. Používateľ (pedagóg) komunikuje so serverom, na ktorom beží výpočet klastrov, cez klienta vo webovom prehliadači. Obrázok 5.7 poskytuje úvodný náhľad.

Obr. 5.7: *Ukážka nástroju na klastrovanie (Nahranie súboru).*

Učiteľ nahrá do aplikácie súbor s odpoveďami študentov (csv) a prípadne pridá aj znenie otázky. Následne na to je požiadavka poslaná na server, kde sa aplikuje model klastrovania. Okrem neho na serveri beží aj proces predspracovania textu. Výsledok sa pošle na klienta, ktorý zobrazí každý klaster oddelene. Učiteľ má teda pohľad na všetky odpovede (obrázok 5.8).



Obr. 5.8: *Ukážka nástroju na klastrovanie (zobrazenie zhlukov).*

Nástroj okrem toho ponúka možnosť výberu algoritmu spomedzi troch variant z minulých podkapitol. Pri k means je nutné aby bol zadaný počet klastrov. Naopak pri affinity a custom k-mean je táto hodnota predikovaná.

Celková časová zložitosť výpočtu je momentálne ovplyvnená vybranými webovými službami použitými pri spracovaní textu. Server musí dlho čakať na odpoveď. Ak by sme však preniesli implementáciu služieb priamo na server celkový čas výpočtu by sa odvíjal od časovej zložitosti algoritmov k-means a affinity propagation. V prípade affinity je časová zložitosť závislá od posielania správ a počtu cyklov. Teda každý uzol v jednom cykle musí poslať záruku a dostupnosť všetkým svojím susedom. Z toho vyplýva, že celková časová zložitosť sa rovná $O(TN^2)$, kde T je počet iterácii algoritmu a N je počet uzlov. V prípade k means to môže byť o niečo lepšie. Pre každý cyklus počítame vzdialenosť bodu s každým centrom cez všetky dimenzie. Teda časová zložitosť sa rovná O(NCID). Kde N je počet uzlov, C počet centier, I je celkový počet iterácii a D je stupeň dimenzionality.

V prípade priestorovej zložitosti affinity propagation si potrebuje pamätať pole záruk a dostupností pre každý uzol. Preto je rovná $O(N^2)$. K means si potrebuje pamätať pozície centier a uzlov v priestore preto O((N+C)D).

5.6 Diskusia

V tejto časti sme poskytli postup ako sme naše výsledky overili. V rámci klasifikácie by bol potrebný väčší dataset pre jednotlivé odpovede, avšak pri krátkych odpovediach vidíme, že úspešnosť klasifikátoru SVM je celkom pozitívna. Taktiež sme postrehli, že využitie inverzného výskytu výrazu pri krátkych textoch zhoršuje úspešnosť. V rámci klastrovania vidíme, že sa nám podarilo zhodnúť s výberom klastrov expertov.

6 Zhodnotenie

V našej práci sme sa venovali niekoľkým oblastiam. Prvou z nich bolo spracovanie krátkych textov v slovenčine. V rámci nej sme si preštudovali rôzne spôsoby ako riešiť problém krátkych textov spojený s gramatikou. Poskytli sme aj vhodné riešenie, ktoré v sebe kombinuje nástroje našej fakulty, ale aj vlastnú implementáciu spracovania textu.

V rámci klasifikácie sme po štúdiu existujúcich riešení poskytli výber klasifikátorov vhodných pre krátke texty s obmedzeným datasetom. Jeho aplikáciou môžeme vidieť zlepšujúcu sa tendenciu, v závislosti od veľkosti trénovacieho datasetu. Keďže predmety na školách sa zvyknú ročne opakovať, sada potrebná pri realizácii sa bude postupne navyšovať. Vďaka tomu vieme automaticky vyhodnocovať odpovede.

Navrhli sme ako zjednodušiť prácu s odpoveďami pri testovaní, použitím klastrovacích algoritmov. Toto riešenie by sa mohlo uplatniť pri aplikáciach, ktoré testovanie umožňujú. Príkladom by mohol byť spomínaný systém ASQ. Ak by sa odpovede na otázky zhlukovali okamžite po odpovedaní študentov, výsledný zoznam by poskytol informatívnu spätnú väzbu pre vyučujúceho. Na základe nej by vedel prispôsobiť svoj výklad potrebám triedy. Vďaka čítaniu priamo zo štrukúry by mohol učiteľ okamžite reagovať na učivo, ktoré bolo pre žiakov náročné.

Realizovaný experiment na expertoch ukazuje, že riešenie sa svojou predikciou klastrov zhoduje s potrebami vyučujúceho. Ďalej sa ukázalo, že aj pre dvoch doménových odborníkov je problém zhodnúť sa v otázke zhotovenia klastrov na odpovediach študentov. Okrem toho sme ukázali, že experimentálna kombinácia dvoch algoritmov affinity propagation a k-means algoritmus dokáže odhadnúť očakávaný počet klastrov. Na nakoniec sme implementovali rozhranie, ktoré demonštruje použitie algoritmu v praxi. Umožňuje jednoduché nahranie zoznamu odpovedí s následnou vizualizáciou klastrov.

Ohraničenia riešenia spočívajú v dĺžke odpovedí. V našom prípade musí ísť o kratšie. Pri odpovediach s viac slovami je úspešnosť nižšia. Dokazujú

to aj naše výsledky. Riešenie tohto problému by vedelo našu prácu posunúť ďalej. Efektivita výpočtu riešenia je taktiež obmedzená webovými službami. Server musí dlho čakať na odpoveď. Toto by sa však dalo obísť presunutím ich implementácie priamo na server. Ako ďalší krok by bolo vhodné implementovanie rozhranie pre naše riešenie priamo do systému ASQ. Týmto krokom by sme vedeli pomôcť učiteľom, ktorí ho využívajú.

Literatúra

[1] Clark, A. Pre-processing very noisy text. In *Proc. of Workshop on Shallow Processing of Large Corpora* (2003), pp. 12–22.

[2] Frey, B. J., and Dueck, D. Clustering by passing messages between data points. *science 315*, 5814 (2007), 972–976.

[3] Jain, A. K., and Dubes, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[4] Jurafsky, D. Speech and language processing: An introduction to natural language processing. *Computational linguistics, and speech recognition* (2000), 189–233.

[5] Kodinariya, T. M., and Makwana, P. R. Review on determining number of cluster in k-means clustering. *International Journal 1*, 6 (2013), 90–95.

[6] Korenius, T., Laurikkala, J., Järvelin, K., and Juhola, M. Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (2004), ACM, pp. 625–633.

[7] Krishna, K., and Murty, M. N. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29*, 3 (1999), 433–439.

[8] Li, T., Zhang, C., and Ogihara, M. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics 20*, 15 (2004), 2429–2437.

[9] Palmer, D. D. Text preprocessing. In *Handbook of Natural Language Processing, Second Edition*. Chapman and Hall/CRC, 2010, pp. 9–30.

[10] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (2002), Association for Computational Linguistics, pp. 311–318.

[11] Ramos, J., et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (2003), pp. 1–4.

[12] Schabes, Y., and Roche, E. Spelling and grammar checking system, July 23 2002. US Patent 6,424,983.

[13] Šimko, M. Sémantika, získavanie a reprezentácia informácií a znalostí. Tech. rep., FIIT STU, 2016.

[14] Sokolova, M., and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management 45*, 4 (2009), 427–437.

[15] Steinley, D. Properties of the hubert-arable adjusted rand index. *Psychological methods 9*, 3 (2004), 386.

[16] Tan, S. An effective refinement strategy for knn text classifier. *Expert Systems with Applications 30*, 2 (2006), 290–298.

[17] Tong, S., and Koller, D. Support vector machine active learning with applications to text classification. *Journal of machine learning research 2*, Nov (2001), 45–66.

[18] Triglianos, V., Pautasso, C., Bozzon, A., and Hauff, C. Inferring student attention with asq. In *European Conference on Technology Enhanced Learning* (2016), Springer, pp. 306–320.

[19] Wang, K., Zhang, J., Li, D., Zhang, X., and Guo, T. Adaptive affinity propagation clustering. *arXiv preprint arXiv:0805.1096* (2008), 1–6.

Príloha A: Technická dokumentácia

V rámci tejto prílohy poskytujeme pohľad na dôležité časti kódu, s ktorými pracujeme. Celá implementácia je riešená v programovacom jazyku Python. Úlohou bolo navrhnúť spôsob predspracovania slovenských textov spojený s klastrovaním a klasifikáciou. Okrem toho bolo potrebné vytvoriť rozhranie, ktoré by simulovalo proces klastrovania v reálnom čase, podobne ako je to v systéme ASQ.

V prvej časti poskytujeme prehľad knižníc použitých pri implementácii. Rozhranie je riešené ako klient server aplikácia, kde na riešenie serveru používame rámec Flask. Pri klientovi sme použili webové technológie HTML, CSS a JavaScript. Druhá časť prílohy poskytuje pohľad na zaujímavé časti kódu.

A.1 Použité knižnice

V tejto podkapitole chceme predstaviť najdôležitejšie knižnice jazyku Python, použité pri implementácii. Tabuľka 6.1 poskytuje prehľad.

Tabuľka 6.1: *Zoznam použitých knižníc*

Knižnica	**Popis**
pandas	Knižnica na prácu s dátami.
numpy	Knižnica s matematickými výpočtami.
requests	Knižnica na prácu nad HTTP protokolom.
string	Knižnica určená na prácu s reťazcami.
seaborn	Knižnica na vizualizáciu dát.
sklearn	Knižnica scikit learnu poskytujúca implementácie algoritmov klastrovania a klasifikácie.
statistics	Knižnica určená na štatistické výpočty.
glog	Knižnica na prácu s file systémom.
urllib	Knižnica na prácu s URL.

A.2 Webové služby

Okrem knižníc pri implementácii používame webové služby vytvorené na našej fakulte. Obidve komunikujú s klientom pomocou HTTP protokolu. V prvej časti poskytujeme na obrázku 6.1 riešenie komunikácie s doplňovačom gramatiky.

```
# Function checks grammar in input string
def check_grammar_string(string):
    url = "http://text.fiit.stuba.sk:8081/diakritika/"
    response = requests.get(url+urllib.parse.quote(string))
    tree = ElementTree.fromstring(response.content)
    return tree[1].text
```

Obr. 6.1: *Ukážka použitia webovej služby doplňovaču gramatiky.*

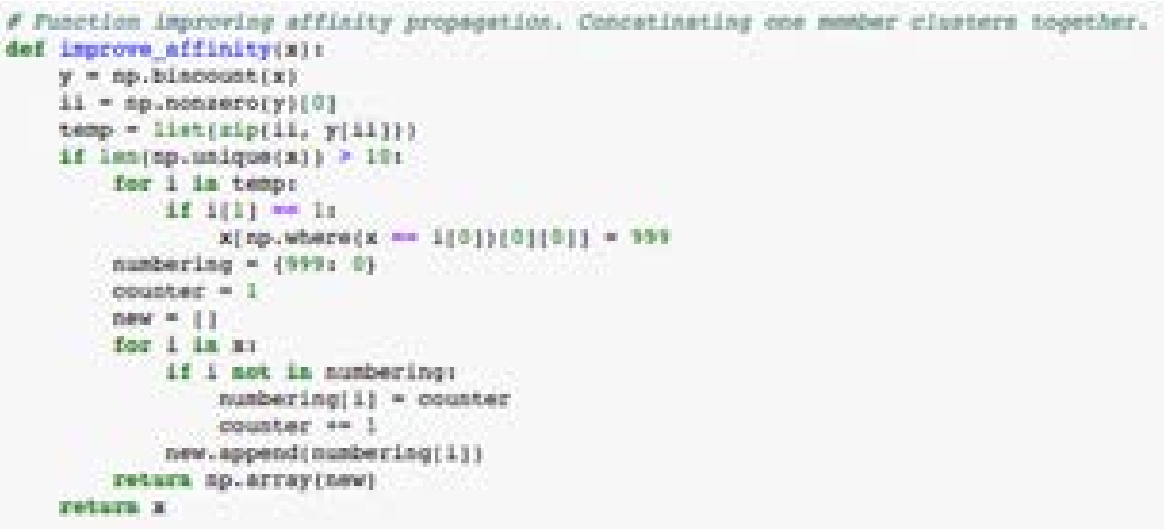
Okrem nej používame lematizátor na zistenie lémy slova. Obrázok 6.2 poskytuje ukážku.

```
# Function lemmatizes string service
def lemmatizeString(string):
    if str(string).isdigit():
        return string
    headers = { 'content-type' : 'text/plain' }
    url = 'http://text.fiit.stuba.sk:8550/lemmatizer/services/lemmatizer/lemmatize/last'
    body = string.encode(encoding='utf-8')
    response = requests.post(url, data=body, headers=headers)
    return response.content.decode('utf-8')
```

Obr. 6.2: *Ukážka použitia webovej služby lematizátora.*

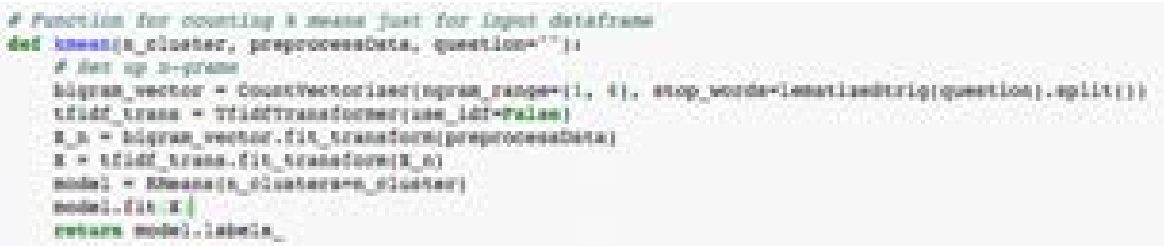
A.3 Dôležité časti kódu

V tejto časti opisujeme najzaujímavejšie časti kódu z hľadiska implementácie. Na zlepšenie predikovania počtu klastrov, algoritmom affinity propagation, sme navrhli vylepšenie. V ňom sa snažíme spojiť klastre s početnosťou jeden do spoločného. Obrázok 6.3 poskytuje ukážku kódu.



Obr. 6.3: *Ukážka vylepšenia affinity propagation.*

Ďalšou zaujímavou časťou je výber čŕt. Na jeho realizáciu používame n-gramy zozbierané z korpusu odpovedí. Okrem toho využívame stop slová, ktoré sú v otázke. Obrázok 6.4 poskytuje ukážku priamo z implementácie algoritmu k means.



Obr. 6.4: *Ukážka výberu čŕt s použitím n-gramov.*

Príloha B: Článok a plagát na konferenciu IITSRC

Clustering and Classification of Student's Answers to Questions

Michal HUCKO*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
hucko.miso@gmail.com

Abstract. In this paper we analyse answers from students on questions, in which is impossible to identify finite number of solutions. We use text clustering and classification. We concentrate on ones written in Slovak language, which are just few words long. Sometimes it is very time consuming to evaluate them manually. Dealing with thousands it can take hours to check. Our main goal is to help with process of checking these answers. Our research question is: How can answer clustering help? In our work we apply different methods and algorithms used in text analysis. Using this method with real time presentation services can lead to an improvement on lectures.

1 Introduction and related work

Evaluating student's answers is essential part of teacher's work. The checking can be time consuming when facing more than hundreds of records. We can see a big lack of automatic methods. It is even impossible to notice some similar parts in answers while going through them one after another. In this case document clustering would be helpful, to monitor similarity in the test answering. Text classification can support teacher in the situations when there is enough labeled data from previous tests. We are working with documents (answers), which are just few words long.

Longer answers are harder to cluster. In work of Hotjo and Staab [1] they found that from longer texts it is very hard to extract essential information which we can use in clustering process. We also provide a method to automatically evaluate answers in situations where same questions repeat after a period. For example, at university each year the same question occurs in tests.

Our main goal is to provide additional information about common mistakes of the class, to person, who is checking results. Gaining this information automatically, could be very helpful with summarization of the test. Teachers would get structural view of answers clustered in several groups. This method can be combined with real time class presentation systems [9], which provide methods to test the class through lecture. In our work we are dealing with Slovak language.

There are a lot of studies working with many languages in text analysis. The key to success lies in a proper text processing. Especially for Slovak language the most important step is preprocessing. In the work of Šimko [7] he uses lemmatization for proper document preparation, but it is not the end. The target group are students and they tend to make mistakes.

Text clustering is quite common technique in the world of text analysis. In the work of Aniket Rangrej and Sayali Kulkarni [6] they use various algorithms to cluster the short tweets of users gained on social network. They found out that k-mean algorithm got the best performance from non-graph based types. Another very popular research of Kishore Papineni [4] is dealing with similarity of automatic translations. They are trying to evaluate accuracy of online translators based on similarity with original translation. They are using the technique called Blue metric which is based on n-gram similarity. All these researches got results based on English documents.

2 Text processing

Before applying different clustering models, we need to process the corpus. Existing researches apply mostly lemmatization and stop words for English texts. But this is not enough for our case. There is also a big lack of approaches dealing with Slovak documents [7].

In our work we first pre-process the documents, then we apply techniques of feature selection and then we continue to stages of applying cluster a similarity algorithms.

---

* Bachelor study programme in field: Informatics
Supervisor: Professor Mária Bieliková, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2017, Bratislava, April 27, 2017, pp. 1–8.

2.1 Text pre-processing

In first phase we apply the lemmatization, we delete special Slovak punctuation and we custom the documents using just letter "i" instead of "y".

For the lemmatization we use Slovak corpus lemmatizator implemented by Róbert Horváth [7]. It works on simple HTTP post request with result of lemmatized document. The success rate of this implementation is around 67%. For better understanding of the functionality suppose the original answer "Bratislava je hlavným mestom Slovenska". After the whole process we get the response "Bratislava byť hlavné mesto Slovensko", which consists of lemmas from original words. Lemma is the base word without any prefix or suffix which can be found in dictionary.

Second step is removal of the punctuation. In our case we just delete the punctuation. It also changes the capital letters to lowercase. Suppose than the output from lemmatization is input for this phase. We will get the result "bratislava byt hlavne mesto slovensko".

Our last step is connected with the grammar. As mentioned at the top of the section we replace all "y" letters with "i". Problem with the correct writing of these two letters belongs to the biggest troubles in Slovak language. Again the result for the input from last step will be "bratislava bit hlavne mesto slovenska".

Last two steps contain the danger of losing some information. When we clear the punctuation of some words there is high chance that from two different ones we will get exactly one.

2.2 N-grams

Each sentence consists of words. We can look at the whole structure as a bag of words. It can cause some significant failures. Words can be connected to phrases which can have few members.

For this purpose, we use n-grams. They are sequence of n ordered elements. In our case we are talking about words. After application of this method we can catch some additional context of terms in the sentence.

We conclude the subsection with some warning. N-grams with high n are memory complex. So it is useful to choose some appropriate n. In our research we use n equals to 4. In many works [2, 4] it is described as the most suitable for measuring document similarity.

2.3 Feature selection

Having the pre-processed data ready we can continue to the next step. Feature selection is the part of the process where we transform string like documents into the numerical ones. There are mostly statistical approaches used so far for the problem. In [3] authors also conclude, that term frequency can be a reliable measure for selecting informative features. Supposing usage od Term frequency the equation for each feature "i" in the document "j" is following:

$$tf(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

where in the numerator we have the total count of term "i" in document "j" and in the denominator we have total number of words in document. Having document "d" equal to "Italian dog is a nice dog" and the query tf ("dog", "d") we will get the result of 0.4.

But there is also a modification of Tf statistic called inverse document frequency (tf-idf) [5]. The main idea behind is giving the most frequent terms in the document lower weight. It makes sense because for example English texts have very high occurrence of terms like "a" or "the". These can easily harm our results. To prevent this situation following equation can be used:

$$tfidf(i,j) = tf(i,j) * idf(j) \tag{2}$$

$$idf(j) = log(\frac{N}{df(j)}) \tag{3}$$

where the original tf term is multiplied with idf term. This represents the weight of competent term "j". In the equation the letter N stands for total number of terms in the corpus and the denominator df stands for the count of term "j" in corpus.

Second technique we proposed is combination of tf-idf with n-grams. We can count the frequency of each "gram" separately. We can than add additional weight to the longer n-grams.

3 Clustering and classification

Having prepared data, we can continue to the phase of applying algorithms. Mentioned in the introduction our main goal is to support teachers who check these answers manually. We identify two main fields where we want to help. First one is automatic evaluation of answers based on sample solutions using text classification methods. This is possible only if we have some previous labeled data. Second is clustering of potentially wrong answers in order to identify common mistakes of class. This is our main field of interest.

3.1 Document classification

For this type of problem, we decided to use text classification algorithms like support vector machine (SVM) and k-nearest neighbor (KNN). For both of these algorithms is feature selection with tf-idf appropriate, but both of them are supervised learning algorithms. Some sample data set is required. This can be helpful in big classes where are more than 40

students. The idea is based on two classes of answers which we want to identify in the rest of document. Number 1 is the class for correct answers and 0 is for incorrect ones.

KNN marks the class of unknown sample based on its K nearest neighbor in the training set [8]. The idea of measuring the distance between two points is the same as in k-mean algorithm. SVM works on a little different principle. It is trying to divide the sample space with hyper line. That represents a border between samples from each class.

3.2 Document clustering

Clustering wrong answers can provide very valuable information for the teacher. Imagine than the software would be able to show common mistake of the class. Combining this with real time presentation software like ASQ where tests are possible, could be very effective. Teachers would be able to change their lecture in order to explain these mistakes, or they can use this information in the final exams.

For the clustering we use the k-mean algorithm. Clustering with it belongs to unsupervised learning problems. We do not know anything about correctness of these answers. We can use this method even when none of them are evaluated. In k-mean clusters are identified with centroids. There are k of them. Each document belongs to one centroid depending on the shortest distance. There many ways how to count the distance. In our work we are trying to apply:

– Euclid distance, where the result is the square root of sum squared error of each component from two points:

$$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (4)$$

– Cosine distance:

$$dist(x, y) = \frac{x^T y}{|x||y|} \qquad (5)$$

The whole algorithm works in three steps:

1. Random initialization of cluster centroids. Ordinary random documents from dataset are marked as centroids.
2. Cluster assignment step. In this step each document belongs to the cluster with nearest centroid.
3. Move centroid step. New centroid for the cluster is counted from average of its documents.

The process continues with step 2 until the centroid stop to move.

With these method is connected the problem of setting the parameter k. It is very hard to guess the right parameter without any knowledge about them. We would like to inform the user of algorithm about the capacity of each cluster for certain k. Later he can manually choose the value. We also get the k from graph based clustering and we use it in k-mean.

For the second approach we use affinity propagation. It is a graph based method where is distance between each document needed. The algorithm tries to divide the graph into suitable number of classes. More close two documents are, more likely they will belong to the same cluster. Comparing it with k-mean, the biggest advantage is in automatic estimation of k. But this approach in general gets worse results.

4 Results

Our data was gathered from teaching at our university. It consists of 15 questions. For each one we have got for 90 answers in average. About each one we know whether it was correct. This set was used for classification results. Except university tests we use also ASQ system [9] for data-set. It is an application for broadcasting and tracking interactive presentations, which can be used to support active learning pedagogues during lectures. Here we have 15 questions together with more than 500 answers. All data was used for clustering results.

In text processing we concentrate on reducing the number of features we get after the inverse document frequency. Before the process we had 17% more features with bag of words. After application of rules in section 3 we have observed decrease of 17%. This result is highly affected with success rate of the lemmatizer.

For the purpose of predicting the correctness of answers we decide to use text classification algorithms SVM and KNN. For both of them we use F1-score "macro" as a metric to provide results. We run two experiments. For both we had smaller set of answers (around 90). In first one we were trying to predict correctness of shorter answers, while in the second we were predicting longer one. Table 1 shows the results for first case. We can see there also predictions for different configurations. In the Table 2 we see the second case. We can conclude that the process of lemmatization has impact on longer documents, also usage of tf-idf.

Table 1. Classification f1 score overview for answers with average word count 3.8974.

	tf-idf lemm	**tf-idf no lemm**	**no tf-idf lemm**	**no tf-idf no lemm**
SVM	0.7460	0.9307	0.9307	0.8545
KNN	0.625	0.7460	0.625	0.7460

Table 2. Classification f1 score overview for answers with average word count 25.9333.

	tf-idf lemm	**tf-idf no lemm**	**no tf-idf lemm**	**no tf-idf no lemm**
SVM	0.9068	0.8295	0.9068	0.8295
KNN	0.9068	0.9068	0.7619	0.7619

For results we split the set for training and test subset. The results were reached on the test set. We used smaller set for prediction, but surprisingly we got high f1 macro score.

In the text clustering field, we have made an application for cluster visualisation. It's a simple tool where teacher can upload his sheet with answers for a question. The program then visualizes the results of k-mean clustering. This tool can also show the process of cluster production in real time. We want to show how it could look in the ASQ system.

Finally, for the clustering we used k-mean algorithm with Euclid distance, also we used affinity propagation with Levenstein distance. To estimate the k in k-mean we used the estimator from affinity propagation. As a metric for our clustering we use ARI. Firstly, we handy mark the clusters for each question and then we compare them using metric.

For the k-mean we end up with ARI equals to 0,391. Affinity propagation clustering got 0,307 this result is highly affected with Levenstein distance which only gives information about quantity of different letters in documents. Surprisingly we found out that the best results we get when we run k-mean without lemmatization. In this case it reaches ARI of 0,497. It means that students tend to make that much grammatical mistakes, that the lemmatizer is unable to work properly. The scale for ARI is [-1, 1].

5 Conclusion

We came up to conclusion that using proper preprocessing and combination of algorithms we can cluster Slovak answers effectively. This method can help to improve whole evaluation process. In combination with real time presentation systems like ASQ, it can be very helpful.

The classification process shows that having repeatedly asked question we are able to predict correct answers. Of course there is the limitation in the number of archived answers.

Acknowledgement: This project is the partial result of the collaboration within the SCOPES JRP/IP, No. 160480/2015.

References

[1] Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, IEEE, 2003, pp. 541–544.

[2] Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

[3] Li, T., Zhang, C., Ogihara, M.: A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 2004, vol. 20, no. 15, pp. 2429–2437.

[4] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[5] Ramos, J., et al.: Using tf-idf to determine word relevance in document queries. In: *Proceedings of the first instructional conference on machine learning*, 2003.

[6] Rangrej, A., Kulkarni, S., Tendulkar, A.V.: Comparative study of clustering techniques for short text documents. In: *Proceedings of the 20th international conference companion on World wide web*, ACM, 2011, pp. 111–112.

[7] Šimko, M.: Sémantika, získavanie a reprezentácia informácií a znalostí. Technical report, FIIT STU, 2016.

[8] Tan, S.: An effective refinement strategy for KNN text classifier. *Expert Systems with Applications*, 2006, vol. 30, no. 2, pp. 290–298.

[9] Triglianos, V., Pautasso, C., Bozzon, A., Hauff, C.: Inferring Student Attention with ASQ. In: *European Conference on Technology Enhanced Learning*, Springer, 2016, pp. 306–320.

Clustering and Classification of Student's Answers to Questions

Author: Michal Hucko

Supervisor: Mária Bieliková

Test

using ASQ

What grows on trees ?

Dataset

Without structure

- no clusters
- no patterns
- no order

Answers:

Labeled data

Classification

k-Nearest Neighbors

Support Vector Machines

Algorithms

Affinity propagation

- We group together clusters with one member

K-means algorithm

1. Random alocation

3. Move centroid

2. Cluster assignment

Text processing

Grammar check

From: "Bratislava je hlavne mesto Slovenska "

To: "Bratislava je hlavné mesto Slovenska "

N-grams

- It is sequence of elements (words)
- N stands for # (maximum)
- We want to catch the context

Lemmatization

From: "Bratislava je hlavné mesto Slovenska "

To: "Bratislava by hlavné mesto Slovenska"

Feature selection

- Term frequency vs Inverse document frequency
- We transform korpus to numerical form (matrix)
- Each document is represented as a vector
- Each column is one n-gram

ASQ With clustering

Cluster 1.

Cluster 2.

Cluster 3.

http://asq.ml.uni.sk/

PeWe@FIIT

Within international project SCOPES in cooperation with University of Lugano

Príloha C: Obsah elektronického média

V nasledujúcej prílohe poskytujeme popis priečinkov a súborov na priloženom elektronickom médiu.

/all_data

- Priečinok s kompletnými dátami z AISU z predmetu princípy softvérového inžinierstva za posledných 5 rokov.

/all_data/PythonExport.xlsx

- Excel s detailnými informáciami o dátachu z AISU.

/asq_analysis

- Priečinok s analýzou klastrovania nad dátami z ASQ.

/asq_analysis/asq_data

- Kompletné dáta zo systému asq. V priečinku sa nachádza priečinok /data, v ktorom sú už spracované odpovede. Ostatné súbory sú data z mongoDB.

/asq_analysis/marked_data

- Oanotované odpovede z pilotného testu.

BP2.pdf

- Baklárska práca elektronická verzia

Classification.ipynb

- Jupyter notebook s implementáciou klasifikácie.

/cluster_visualization

- Kompletná aplikácia na vyzualizovanie klastrov. Priečinok je PyCharm projekt, napísaný v pythone3.

/expert_analysis

- Dáta z eperimentu s expertmi. V rámci neho sú priečinky s pozhlukovanými oodpoveďami.

/expert_analysis/Expert_analysis.ipynb

- Jupyter notebook (python3) s analýzou expertou.

/latex

- Priečinok s Latex verziou bakalárskej práce

/make_cluster

- Aplikácia použitá pri experimente s expertami. Priečinok je PyCharm projekt, napísaný v pythone3.

/psi_presentations

- Priečinok prezentácii z PSI do systému ASQ. Sú napísané nad frameworkom impress.

Príloha D: Zhodnotenie plánu práce

V tejto prílohe rozoberáme plán, ktorý sme si zadali na konci zimného semestra. Tabuľka 6.2 poskytuje prehľad plánových činností.

Tabuľka 6.2: *Plán činností na letný semester*

Kedy ?	**Čo?**
1.1. - 15.1.	Aplikovanie BLEU meriky na klasifikáciu správnych odpovedí.
16.1. - 30.1.	Práca na zhlukovaní nesprávnych odpovedí študentov.
31.1. - 13.2.	Aplikácia písmenkových n-gramov na zistenie podobností slov v slovenčine. Práca na ITSRC.
14.2. - 28.2.	Práca na použití korpusového Pos taggeru pre zlepšenie identifikácie správnych odpovedí. Pos tagger poskytne slovné druhy slov vo vete. Vychádzame z predpokladu, že kľúčové slová v správnej odpovedi sú prevažne kombináciou podstatých a prídavných mien.
1.3. - 14.3.	Použitie interpunkcie pre lepšiu identifikáciu správnych odpovedí. Vychádzame z predpokladu, že aj počet viet v odpovedi môže dopomôcť.
15.3 - 31.3.	Príprava pokusu pre získanie odpovedí počas predmetu PSI.
1.4. - 24.4.	Realizácia pokusu. Následné spracovanie získaných dáta aplikácia algoritmov.
25.4. - 9.5.	Vyhodnotenie výsledkov práce. Dokončenie práce.

V rámci letného semestra sa nám podarilo prejsť všetky body v predstihu. Aplikovanie BLEU metriky nebolo vhodné pre náš problém, avšak využili sme použitie n-gramov. Vďaka nim sa výsledky zhlukovania zlepšili. Zhlukovanie sa nám podarilo implementovať v plnom rozsahu. Úspešne sme sa tento rok zúčastnili konferencie IITSRC s vypracovaným príspevkom.

Použitie postaggeru a interpunkcie nemalo vplyv na zlepšenie výsledkov klastrovania kvôli tomu, že v našej práci sa venujeme kratším odpovediam.

Klastrovanie v závislosti od počtu interpunkcie môže mať zmysel pri dlhších textoch.

Príloha E: Inštalačná príručka

Pre úspešne spustenie klasifikátorv a klastrovania je potrebné mať stiahnutý python3. Okrem toho je potrebné mať k dispozícii všetky knižnice spomínané v prílohe A. Riešenie poskytujeme formou jupyter notebooku. Je to technológia, ktorá okrem priameho vykonávania kódu umožňuje prehľadnú dokumentáciu. Na to, aby sme ju mohli spustiť lokálne je treba mať stiahnutú knižnicu jupyter. Na stránke[1] projektu sa nachádza kompletný manuál s krokmi pre jednotlivé operačné systémy.

Po uspešnom nainštalovní prejdeme do adresára na CD /bakalarka a spustime príkaz:

```
$ jupyter notebook
```

Následne sa spustí interaktívny prehľad na adrese localhost:8888.

Pre inštaláciu ostatných knižníc odporúčame použitie nástroja na správu balíkov pip, ktorý je k dispozícii štandardne s inštaláciou jazyku python. Nižšie uvádzame príklad inštalácie scikit-learnu

```
$ pip install -U scikit-learn
```

Pre spustenie aplikácie na vyzualizáciu je okrem knižníc z prílohy A potrebné mať nainštalovanú knižnicu rámca Flask.

```
$ pip install Flask
```

Pre spustenie interaktivneho módu prezentácii z ASQu je potrebné aby na pozadí bežal server.

```
$ python3 -m http.server
```

Po jeho spustení je možné spustiť súbor index.html príslušnej prezentácie. Odporúčame použitie prehliadača Google Chrome.

[1] http://jupyter.readthedocs.io/en/latest/install.html