VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

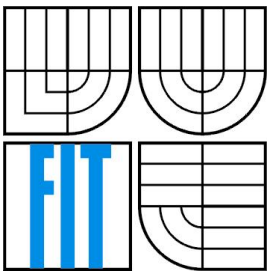
EXTRAKCIA INFORMÁCIÍ ZO SLABO ŠTRUKTÚROVANÉHO TEXTU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Matej Minárik
AUTHOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EXTRAKCE INFORMACÍ ZE SLABĚ STRUKTUROVANÉHO TEXTU

INFORMATION EXTRACTION FROM LOOSELY STRUCTURED TEXT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Matej Minárik

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Radek Burget, Ph.D.

BRNO 2016

Abstrakt

V dnešnej dobe sa stretávame s pojmom Web 2.0, ktorý predstavuje web dokumentov. Dokumenty na webe sú dostupné vo väčšine prípadov v neštruktúrovanej, alebo čiastočne štruktúrovanej podobe. Pre lepšie a jednoduchšie vyhľadávanie však potrebujú mať vyhľadávače dáta v štruktúrovanej podobe. Práca sa zameriava na analýzu spôsobov extrakcie informácií z neštruktúrovaného textu. V práci analyzujeme jednak použitie rôznych typov klasifikátorov, ale aj metód, ktoré nepotrebujú mať k dispozícii anotované dáta na trénovanie interných modelov. Ďalej navrhujeme metódu na extrakciu terapeutických indikácií a účinných látok z príbalových letákov liekov.

Abstract

Nowadays we are speaking about Web 2.0, which means the web of documents rather than the web of data. Documents are mostly unstructured, or just partially structured, but search engines need data in structured form in order to provide better search results. The process of extracting structured data from partially structured documents is the main goal of this work. In this work we are analyzing information extraction methods, namely classification methods, which need annotated training data, in order to create their inner model. We also analyze methods, which do not need training. These methods are initialized with a few data examples we are interested in extracting. We propose an extraction method in order to extract therapeutic indications and active substances from medical information sheets.

Kľúčové slová

extrakcia informácií, strojové učenie, príbalové letáky liekov

Keywords

information extraction, machine learning, medication information sheets

Citácia

MINÁRIK, Matej. Extrakce informací ze slabě strukturovaného textu. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Extrakce informací ze slabě strukturovaného textu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Radek Burget, Ph. D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Bc. Matej Minárik
23. mája 2016

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu za odborné vedenie a konštruktívne pripomienky k práci. Ďalej by som sa chcel poďakovať priateľke a rodine za podporu počas štúdia.

Obsah

1 Úvod

S celosvetovým rozšírením Internetu prišlo aj obrovské rozširovanie jeho obsahu. Každý deň pribúda množstvo nových webových stránok s novým obsahom. Dnes si už nevieme predstaviť vyhľadávanie informácií na internete bez pomoci vyhľadávačov ako je napríklad Google. Vyhľadávače nám pomáhajú rýchlejšie nájsť relevantné odpovede na naše otázky. Táto práca by mala pomôcť pri získaní štruktúrovaných dát pre vyhľadávač liekov, ktorý by som chcel v budúcnosti vytvoriť.

V rámci Slovenskej republiky, existuje databáza všetkých aktuálne dostupných a schválených liekov, ktorú spravuje Štátny ústav pre kontrolu liečiv (ďalej SUKL[1]). Na svojej stránke má ústav zverejnenú databázu liekov s ich základnými informáciami a 2 dokumentami. Jeden z dokumentov je príbalový leták lieku a druhý obsahuje súhrn charakteristických vlastností lieku. Tieto dokumenty sú však slabo štruktúrované a informácie v nich nie sú v adekvátnej forme pre vyhľadávač liekov. Základnou úlohou teda bude extrahovať štruktúrované informácie o liekoch z dostupných neštruktúrovaných dokumentov.

Keďže nemám žiadne predošlé skúsenosti v tejto oblasti, predovšetkým budem musieť preštudovať metódy spracovania neštruktúrovaného textu, resp. prirodzeného jazyka. Metódy extrahovania informácií z takéhoto textu a spôsoby uloženia získaných informácií.

Druhá kapitola sa venuje opisu postupov a metód spracovania prirodzeného jazyka s krátkym doplnkom o korpusoch a konkrétne Slovenskom národnom korpuse SNR. Tretia kapitola podrobne rozoberá extrakciu informácií spolu s jej históriou, typickými úlohami, tvorením a vyhodnocovaním úspešnosti extrakčných systémov. Ďalej rozoberá prístupy extrakcie založené na vopred anotovanom korpuse dát, pomocou ktorého sa natrénuje vnútorný model a následne je možné klasifikovať nové dáta. Tiež sa táto kapitola venuje prístupom extrakcie, ktoré nevyžadujú vopred anotovaný korpus ale na základe výskytov hľadaných dát v textoch si vytvorí pravidlá na ďalšie vyhľadávanie. Štvrtá kapitola opisuje a analyzuje spôsoby uloženia slabo štruktúrovaných dát s dôrazom na sémantický web a RDF. Piata kapitola je venovaná analýze dokumentov príbalových letákov liekov.

V šiestej kapitole opisujem konkrétne zvolené knižnice, pomocou ktorých sa realizuje extrakcia, okrem toho rozoberám celkový návrh metódy extrakcie. Siedma kapitola rozoberá implementáciu metódy, architektonický pohľad na vzniknutú aplikáciu, pseudokód hlavnej funkcionality a analýzu časovej a pamäťovej zložitosti. V poslednej kapitole opisujem, ako som metódu overoval a zhodnocujem celkové dosiahnuté výsledky.

[1] www.sukl.sk

2 Typické úlohy spracovania prirodzeného jazyka

Mnoho zdrojov, ktoré som pri prieskume našiel, sa venujú spracovaniu angličtiny. Angličtina je svojou stavbou pomerne jednoduchá na automatické spracovanie. Okrem toho je najrozšírenejším jazykom na vedecké publikácie a iné zdroje, ktoré následne môžu byť spracovávané. Keďže ja budem čerpať informácie z textov v slovenčine, budem v tejto kapitole čerpať hlavne z [1]. Uvedený zdroj uvádza aplikáciu úloh na Slovenský národný korpus (SNK), o ktorom budem písať koncom tejto kapitoly.

2.1 Tokenizácia

Prirodzený text sa intuitívne dá rozdeliť na menšie časti, ako napríklad odstavce, vety či jednotlivé slová. Práve rozdelenie s najnižšou spomenutou granularitou nám definuje jednotlivé slová. Slovo je základná jednotka textu, oddelená od iných slov medzerami. Tokenizácia je veľmi dôležitou súčasťou spracovania prirodzeného jazyka. Od jej výsledkov priamo závisia ďalšie úlohy, ako napríklad určovanie slovných druhov.

Tokenizáciu avšak nemožno vykonávať iba na základe oddeľovania jednotlivých slov medzerami. Existujú slová, ktoré reprezentujú jednu jazykovú jednotku, no sú zapísané prostredníctvom viacerých slov. Napríklad "1 456" obsahuje dve časti, ale reprezentuje jedno a to isté číslo. Ďalšie príklady sú združené pomenovania ako "Nové Mesto nad Váhom", alebo zložený tvar "na bielo". Rozdeliť uvedené jazykové jednotky na viacero slov je v poriadku, avšak v neskorších etapách je potrebné zapojiť logiku, ktorá bude tieto slová naspäť spájať.

2.2 Lematizácia

Lematizácia je proces konverzie nájdeného slova na jeho základný (slovníkový) tvar, ktorý budeme označovať *lema.* Lemy budeme zapisovať vždy s malým písmenom. Odstránenie veľkých písmen môže viesť k strate sémantickej informácie, avšak pri hľadaní slov s malým písmenom, dostaneme oveľa viac významov. Uvediem niekoľko príkladov lematizácie slovenských slov, pre ďalšie podrobné príklady a pravidlá by som čitateľa odkázal na spomenutú literatúru.

Podstatné mená sa lematizujú prevodom na *nominatív jednotného čísla* (N sg.). Ako príklad môžeme uviesť „*lieku => liek"*. Pri niektorých slovách v množnom čísle môže dôjsť k zmene z množného čísla

na jednotné ako napríklad „*rukavice => rukavica“*. Podstatné mená si pri lematizácii zachovávajú rod, v akom sa nachádzajú v texte. Príkladom je „*nadriadenej => nadriadená“* a nie „*nadriadený“*.

Prídavné mená sa lematizujú prevodom na *nominatív jednotného čísla mužského rodu (N sg. mask.)*. Príkladom je „*účinnej => účinný“*.

2.3 Určovanie slovných druhov

V anglickom jazyku sa môžeme stretnúť s názvami ako *“POS (part-of-speech) tagging”*. Program spracúva jednotlivé slová a určuje ich slovné druhy. Proces nie je tak jednoduchý, ako sa na prvý pohľad môže zdať. Niektoré vety môžu obsahovať zamlčaný podmet, napríklad veta *“Pracuje.”*. V iných sa môže zdanlivo jednoduché slovo vyskytnúť v úplne inom kontexte. Napríklad vo vete *“Miesto toho aby študoval, nerobil nič.”*, sa slovo miesto vyskytuje ako spojka[2], aj keď vo väčšine prípadov je to podstatné meno.

Algoritmy na určovanie slovných druhov sú rôzne. Najrozšírenejšou implementáciou je použitie skrytých Markovových modelov (Hidden Markov Models). Markovove modely sú založené na vytvorení vnútornej reprezentácie pravdepodobnostných tabuliek pre jednotlivé slovné druhy a slová. Napríklad slovo miesto je zo 70% podstatné meno a z 20% spojka. Markovove modely vyššieho rádu fungujú na rovnakom princípe, ale berú do úvahy aj kontext v akom sa slovo nachádza. Teda pravdepodobnostné tabuľky si vytvárajú napríklad pre trojice po sebe nasledujúcich slov. Vlajková loď takýchto systémov bol systém CLAWS, ktorý pracoval presne na takomto princípe a dosahoval 96-97% úspešnosť. Zvyšným slovám, bolo možné priradiť slovné druhy manuálne. [2]

Tie jednoduchšie sú založené na pravidlách *(rule-based)*, podľa ktorých sa určujú slovné druhy. Základný princíp je, že systém je potrebné natrénovať na vzorke označených slov a následne sa iteratívne vylepšujú a dolaďujú vygenerované pravidlá. Pravidlo môže byť napríklad: *Ak je aktuálne slovo má značku A a nachádza sa v kontexte C, zmeň značku na B.* [3]

Iné prístupy používajú rozhodovacie stromy, ktoré preukázali približne rovnako presné výsledky, avšak nepotrebujú tak veľké trénovacie korpusy ako Markovove modely. [4]

2.4 Korpus

V predošlých kapitolách som niekoľko krát spomenul trénovacie korpusy. Spočiatku som bol presvedčený, že pre Slovenský jazyk nenájdem voľne dostupný jazykový korpus, avšak opak bol pravdou.

[2] http://slovnik.azet.sk/pravopis/slovnik-sj/?q=miesto

2.4.1 Čo je to korpus?

Korpus je sada textov, ktoré sa budujú v elektronickej podobe. Môže obsahovať texty rôznych žánrov, ku ktorým sú pridávané informácie o jednotlivých slovách, vetách aj celých textoch. Korpus môže slúžiť hlavne pri bežnom učení jazyka a overovaní vedomostí o jednotlivých slovách v praxi. V informatike sa práve korpus dá použiť ako vstupný súbor trénovacích dát pre štatistické metódy spracovania prirodzeného textu.

Pri tvorbe korpusov je potrebné najprv získať všetky potrebné povolenia od autorov textov. Dáta sa najčastejšie získavajú v elektronickej podobe, zriedka sa používa metóda OCR *(Optical Character Recognition),* teda automatický prevod napríklad skenovaných materiálov na elektronický textový dokument. Korpusové dáta najprv prejdú technickým čistením od znakov editorov, v ktorých boli vytvárané, obrázkov, tabuliek, grafov a pod. Následne texty podstupujú tokenizáciu. Jednotlivým slovám sú potom priradené dodatočné informácie.

2.4.2 Slovenský národný korpus[3]

SNK je korpus, obsahujúci texty z rôznych žánrov v slovenčine od roku 1955. Všetky texty a slová sú obohatené o dodatočné lingvistické a jazykové informácie. Korpus je možné bezplatne používať na jednoduché vyhľadávanie cez WWW rozhranie, alebo je možné ho používať v plnej miere na vedecké, výskumné a učebné účely po registrácii.

[3] http://www.korpus.sk

3 Extrakcia informácií

Extrakcia informácií je proces získavania štruktúrovaných informácií z neštruktúrovaného alebo čiastočne štruktúrovaného textu. Čiastočne štruktúrované dokumenty obsahujú čiastočnú vizuálnu informáciu o štruktúre textu, ktoré obsahujú. Dokument, ktorý má pomenované odseky v určitom poradí, sa dá považovať za čiastočne štruktúrovaný. V tejto kapitole by som sa zameral na históriu tvorby systémov na extrakciu informácií, jednotlivé prístupy k extrakcii a hodnotiace parametre týchto systémov. Extrakcia informácií je preklad z anglického „*Information extraction"* a v ďalších kapitolách sa budem na tento proces odkazovať skratkou IE.

V dnešnej dobe je väčšina informácií uložená na webe v neštruktúrovanej forme, v podobe novinových článkov, vedeckých článkov, blogov, a rôznych iných. Spracovanie takého obrovského množstva informácií je pre človeka neuskutočniteľné. Automatická extrakcia informácií je preto absolútne kľúčová pri analýze takého obrovského množstva dát. Dobrým príkladom je napríklad biomedicínska oblasť, kde každý rok vznikne pol milióna článkov [5] a nemalé finančné prostriedky sú vynakladané na kurátorské aktivity. V odbore zhromažďovania spravodajských informácií panovalo v roku 1990 presvedčenie, že objem informácií, ktoré má pracovník každý deň prečítať je rovný objemu diela *Vojna a Mier*, ktoré napísal *Lev Nikolajevič Tolstoj* a v pôvodnom vydaní obsahovalo 1225 stránok. V roku 1995 si už mysleli, že je toho oveľa viac.

Medzi typické úlohy extrakcie informácií patrí napríklad získavanie štruktúrovaných údajov o odkupovaní firiem. Môžeme si pre tento príklad definovať reláciu:

$$Kúpa(Kto, Koho, Za\ koľko, Rok) \quad (1)$$

Napríklad z vety "Firma Microsoft kúpila v roku 2011 Skype za 8.5 miliardy dolárov." by sme vyextrahovali:

$$Kúpa(Microsoft, Skype, 8.5\ miliardy\ dolárov, 2011) \quad (2)$$

Ďalšie úlohy zahŕňajú napríklad extrakciu dôležitých informácií zo životopisov uchádzačov o zamestnanie alebo získavanie informácií o rôznych udalostiach. Často sa v literatúre spomína extrakcia faktov o bombových atentátoch, miesta útoku, počtu zranených a organizáciu, ktorá sa k útoku prihlásila. Zaujímavá práca je tiež [6] v ktorej autori vyvinuli systém na extrakciu cestovateľských doporučení a varovaní, ktoré zverejňujú ministerstvá Francúzska, Nemecka a Veľkej Británie na svojich stránkach. Ako výsledok vznikol systém *SPROUT*. Projekt vznikol na objednávku leteckej spoločnosti, ktorá ho používala na predikciu vyťaženosti svojich letov. Okrem analýzy obrovského množstva dát

svojich zákazníkov, spoločnosť zahŕňala do svojich algoritmov aj parametre získané extrakciou informácií zo stránok ministerstiev.

3.1 História extrahovania informácií

Extrakciu informácií posunuli vpred konferencie MUC (Message Understanding Conference), ktoré sa konali v rokoch 1987-1997 pod záštitou agentúry DARPA[4]. Konferencie boli organizované formou súťaže, kde organizátori poskytli súťažiacim tímom korpusy, na ktorých mohli jednotlivé tímy natrénovať svoje programy a následne sa v rámci konferencie vyhodnocovali úspešnosti poskytnutých riešení. Korpusy konferencií boli zamerané na:

- MUC-1, MUC-2 - lodné správy
- MUC-3, MUC-4 - správy o aktivitách teroristických skupín v Latinskej Amerike
- MUC-5 - zlučovanie firiem a mikroelektronika
- MUC-6 - správy o zmenách vo vedení firiem
- MUC-7 - správy o vypúšťaní vesmírnych satelitov

Korpusy z posledných dvoch konferencií nie sú voľne dostupné, dajú sa však stiahnuť za poplatok zo stránok *Linguistic Data Consortium*[5]. Prvé IE systémy boli založené na ručnom vytváraní pravidiel, ktoré špecifikovali určité lingvistické vzory na základe ktorých sa vyhľadávali hodnoty extrakcií. Tento proces bol však príliš náročný a vyžadoval si veľa času a skúsenosti experta, ktorý sa vyznal jednak v danej doméne a jednak vedel v akom tvare má konštruovať pravidlá tak, aby sa dali využiť v IE systéme. Neskôr prišli systémy, ktoré si trénovali interný model na základe ktorého následne extrahovali údaje. Tieto systémy bolo potrebné natrénovať na anotovanom korpuse. Anotácia korpusov je menej časovo náročná a môže ju vykonávať aj človek, ktorý toho nevie veľa o IE systémoch avšak stále je to komplikovaná úloha, ktorá si vyžaduje určitú úroveň odbornosti a spotrebovaného času. V dnešnej dobe sa kladie vysoký dôraz na IE systémy, ktoré sú jednak nezávislé na doméne a nepotrebujú vopred anotovaný korpus alebo extrakčné pravidlá. Takéto systémy vychádzajú buď z prvotného náznaku extrahovaných dát, tzv. “seed”, alebo vygenerujú extrakčné pravidlá, ktoré sú následne posúdené a revidované doménovým expertom.

[4] http://www.darpa.mil/
[5] https://www.ldc.upenn.edu/

3.2 Typické úlohy IE

Jedným z výsledkov MUC konferencií bola identifikácia týchto piatich základných úloh IE systémov [7]:

1. Named Entitity recognition (NE) - klasifikuje mená, názvy, dátumy, miesta, atď.
2. Coreference resolution (CO) - identifikuje odkazovanie na entity
3. Template Element construction (TE) - opisuje entity na základe CO
4. Template Relation construction (TR) - identifikuje vzťahy medzi entitami
5. Scenario Template production (ST) - ukladá výsledky TE a TR do šablón

Predpokladajme túto vetu:
"*V utorok vynašli účinný nový liek, ktorý je výsledkom práce doktora MUDr. Palkoviča a jeho tímu. Doktor Palkovič pracuje pre Slovenskú Akadémiu Vied.*"

NE definuje entity utorok, liek, MUDr. Palkovič, tím a Slovenská Akadémia Vied. CO identifikuje, že slovo *ktorý* sa odkazuje na slovo *liek* a MUDr. [8] [8] Palkovič a doktor Palkovič je tá istá osoba. TE priradí lieku vlastnosti *nový* a *účinný*. TR identifikuje že MUDr. Palkovič pracuje pre Slovenskú Akadémiu Vied a že liek objavil on a jeho tím. ST definuje udalosť objavu nového lieku a priradí jej spomenuté entity.

Named Entity recognition už v dnešnej dobe dokáže identifikovať až 95% entít, ktoré sa v danom texte nachádzajú. Ak by sme túto úlohu dali človeku, tiež by nebol schopný identifikovať úplne všetky entity v texte, takže môžeme tvrdiť že NER systémy dnes pracujú na úrovni ľudí. V preštudovanej literatúre som sa stretol iba so systémami, ktoré pracujú nad textami v angličtine.

Coreference resolution identifikuje vzťahy medzi entitami v texte. Ak sa na určitú entitu odkazujeme v ďalšom texte nepriamo, táto úloha by mala byť schopná identifikovať, na ktorú entitu sa odkazujeme. Hlavný zámer tejto úlohy tkvie v poskytnutí podkladov pre ďalšie spracovanie. Táto úloha je závislá na doméne a pri prechode do inej domény je potrebné upraviť jej nastavenia. V praxi sú výsledky pomerne nepresné. Pri nepriamych odkazoch sú systémy schopné odhaliť približne 50-60% odkazovaných entít správne.

Template Element construction stavia na výsledkoch predošlých úloh. Úloha je schopná priradiť dodatočné informácie k entitám. V praxi táto úloha dosahuje približne 80% úspešnosť.

Template Relation production sa zameriava na prepojenie entít s dodatočnými informáciami z úlohy TE. TR úloha môže napríklad určiť zamestnanecký vzťah osoby a nejakej spoločnosti, alebo rodinný vzťah dvoch osôb. V praxi úloha dosahuje približne 75% úspešnosť.

Scenario Template production produkuje inštancie definovaných šablón, ktoré by sme chceli extrahovať. ST je komplexná úloha, ktorá zoskupuje výsledky TE a TR. MUC systémy dosahovali úspešnosť približne 60%, ale ľudia približne 80%, čo vypovedá o komplexnosti úlohy.

3.3 Hodnotenie IE systémov

Veľmi dôležitou súčasťou procesu extrakcie informácií je vyhodnocovanie IE systémov. Na vyhodnocovanie je potrebné disponovať dátami, ktoré slúžia ako referenčné riešenie, v angličtine *"golden standard"*. Vypracovanie takéhoto riešenia je samozrejme veľmi náročné na čas a aj úsilie. Je dôležité poukázať na to, že takéto riešenie by mal vytvoriť doménový expert, pri ktorom je vyššia pravdepodobnosť kvalitného výstupu. Ani to nám však nemusí zaručiť bezchybné referenčné dáta. Už počas MUC konferencií, boli zavedené nasledovné metriky: *precision(presnosť), recall(úplnosť)* a od nich odvodená tzv. *F-measure(F-miera).* [8]

Predpokladajme jednoduchý systém, ktorý má hľadať v texte informácie o kúpach spoločností, ako sme uvažovali začiatkom tejto kapitoly:

$$Kúpa(Kto, Koho, Za\ koľko, Rok) \quad (3)$$

Predošlú reláciu budeme nazývať "Kúpa", ktorá má 4 sloty "Kto", "Koho", "Za koľko" a "Rok". Definujme si nasledovné symboly:

- N - celkový počet referenčných slotov
- M - celkový počet vyplnených slotov
- C - celkový počet správnych slotov - vyplnené sloty, ktoré sa zhodujú s referenčnými a sú ohodnotené ako správne
- S - celkový počet nesprávnych slotov - vyplnené sloty, ktoré sa zhodujú s referenčnými a sú ohodnotené ako nesprávne
- D - počet referenčných slotov, ktoré sa nezhodujú so žiadnymi vyplnenými slotmi
- I - počet vyplnených slotov, ktoré sa nezhodujú so žiadnymi referenčnými slotmi

Z predošlých definícií by malo byť zrejmé, že:

$$N = C + S + D \quad (4)$$

$$M = C + S + I \quad (5)$$

Pre každý referenčný súbor je *N* fixné, avšak *M* sa mení v závislosti od testovaného systému. Závislosť *N* a *M* je zrejmá z predošlých vzorcov a závisí od počtu slotov, ktoré systém neidentifikoval, alebo identifikoval ale nemal.

Na základe týchto symbolov vieme definovať dve základné miery a to sú *presnosť(P)* a *úplnosť(R)*:

$$P = \frac{C}{M} = \frac{C}{C + S + I} \quad (6)$$

$$R = \frac{C}{N} = \frac{C}{C + S + D} \quad (7)$$

Presnosť je percentuálne vyjadrenie vyplnených slotov, ktoré sú správne vyplnené. **Úplnosť** je percentuálne vyjadrenie referenčných slotov, ktoré sme správne vyplnili.
Veľmi neformálne by som presnosť preformuloval ako záruku toho, že keď niečo vyplním tak je to správne vyplnené, zatiaľ čo úplnosť vypovedá o miere identifikácie slotov, ktoré by mal byť systém schopný vyplniť. V našej doméne extrakcie informácií z neštruktúrovaného textu o liečivách je veľmi dôležitá čo najvyššia presnosť, zatiaľ čo úplnosť je až na druhom mieste.

Na základe presnosti a úplnosti vieme ďalej definovať jedinú mieru, ktorá hovorí o presnosti systému a tou je *F-miera,* definovaná:

$$F = \frac{P * R}{(1 - \alpha) * P + \alpha * R} \quad (8)$$

kde α slúži ako váha pre použité miery a je z intervalu <0; 1>. Pri F-miere sa rovnako ako pri predošlých predpokladá, že čím vyššia je daná miera, tým lepší je testovaný systém. V [8] ďalej predstavili mieru chyby ako:

$$E = 1 - F \quad (9)$$

3.4 IE systémy “s učiteľom”

Prvotné IE systémy boli založené na hľadaní vzorov v texte. Vzory, resp. pravidlá boli ručne vytvárané doménovými expertami. Ručné vytváranie pravidiel je proces náročný hlavne na čas. Odhadovaný čas na vytvorenie pravidiel pre systém UMass, ktorý bol použitý na MUC-4 konferencii je 1500

človekohodín. Z tohto dôvodu boli neskôr vytvárané nové prístupy, ako napríklad štatistické metódy alebo strojové učenie.

IE systémy s učiteľom predstavovali prvý krok k redukcii obrovského množstva času, potrebného na adaptáciu systémov na nové domény. Prvotné predstavy boli založené na tom, že anotačnú prácu môže robiť človek, ktorý nie je doménový expert ale má isté poznatky v danej oblasti a zároveň sa zredukuje čas potrebný na anotáciu na rád týždňov.

Na extrakčný proces je možno nahliadnuť ako na klasifikačný problém. Najprv sa napríklad prostredníctvom regulárnych výrazov identifikujú časti textu, v ktorých by sa potenciálne mohli nachádzať potrebné informácie na extrakciu. Následne sa vyberie okolie kľúčového slova (jeho *k* susedných slov). Tento segment je možné previesť na binárny vektor vlastností, v angličtine *"feature vector",* v ktorom jednotlivé termy vektoru vyjadrujú prítomnosť určitého slova. V [9] používajú techniku *Information Gain* na výber najlepších vlastností, teda vlastností s najväčším informačným prídavkom do vektoru, čím sa prirodzene redukuje jeho veľkosť. Podľa [9] na takúto klasifikáciu textu sú najviac vhodné tieto klasifikátory: Naive Bayes, Support Vector Machines, C4.5 a k-Nearest Neighbours. Pre všetky tieto klasifikátory platí, že na natrénovanie ich vnútorného modelu je potrebné disponovať anotovanou vzorkou dát, na ktorej je možné spustiť trénovaciu fázu.

Klasifikácia je proces, pri ktorom máme daný objekt s určitými vlastnosťami a výstupom je zaradenie tohto objektu do určitej klasifikačnej triedy na základe jeho vlastností. Zaradenie objektu do triedy je možné chápať ako priradenie ďalšej kategorickej vlastnosti danému objektu.

3.4.1 Naive Bayes

Bayesovská klasifikácia patrí medzi najjednoduchšie klasifikačné metódy. Pre vysvetlenie tohto prístupu je potrebné si najskôr definovať pravdepodobnosť a podmienenú pravdepodobnosť. Pravdepodobnosť výskytu je daná hodnotou z intervalu <0; 1>.

$P(X)$ je pravdepodobnosť, že nastane jav X. $P(X|Y)$ je pravdepodobnosť, že nastane jav X, ak vieme že nastal jav Y. Na tomto základe môžeme vyjadriť Bayesov vzorec, ktorý sa používa pri tejto klasifikačnej metóde:

$$P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)} \qquad 10$$

Majme daný vektor vlastností X=(x1, x2, … xN), ktorý je potrebné zaradiť do jednej z tried, ktoré budeme označovať C1, C2, … Cm. Vektor X zaradíme do takej triedy Ci, ktorá má najvyššiu podmienenú pravdepodobnosť P(Ci|X). Hľadáme teda maximálne

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}, \text{ kde } P(X) \text{je konštantné pre všetky } C_i \quad 11$$

z čoho vyplýva, že sa snažíme maximalizovať čitateľa. Čiastkové výpočty získame takto:

$$P(C_i) = \frac{|s_i|}{|S|} \quad 12$$

kde $s_i$je množina trénovacích objektov, ktoré patria do triedy C_i a S je množina všetkých trénovacích objektov.

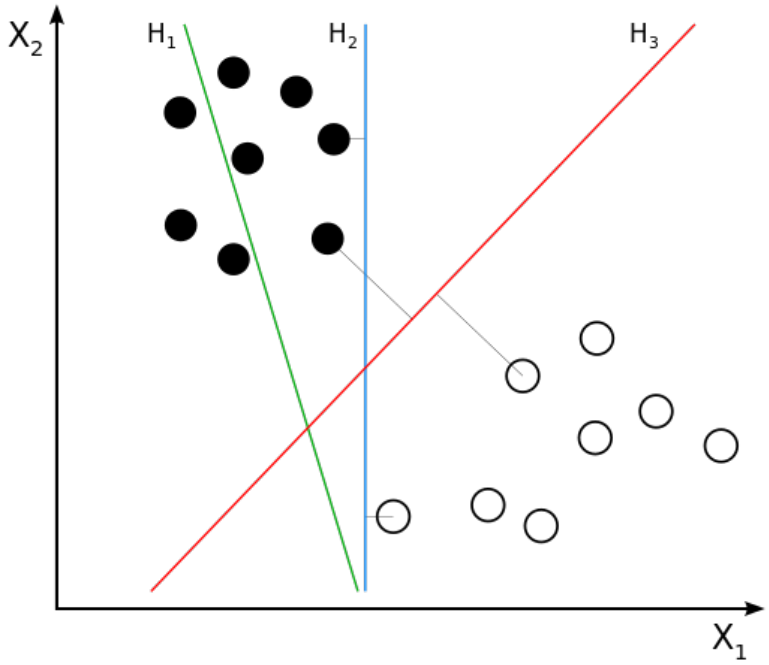
$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \quad 13$$

kde, $P(x_k|C_i) = \frac{|s_{ik}|}{|s_i|}$ a $|s_{ik}|$ je počet trénovacích vzorov z triedy C_i, ktoré spĺňajú podmienku, že hodnota k-teho atribútu je rovná x_k. Tento prípad platí v prípade, že $x_k$nadobúda diskrétnych hodnôt. Pokiaľ $x_k$nadobúda spojité hodnoty, tak $P(x_k|C_i)$ sa získa Gaussovou normálnou funkciou.

Bayesovský klasifikátor má problém v prípade, že nejaká čiastková pravdepodobnosť $P(x_k|C_i) = 0$, keďže nám čiastkový súčin pravdepodobností vráti hodnotu 0. Riešením je napríklad *Laplaceova korekcia,* ktorá poskytuje niekoľko riešení. Jedným z riešení je pridať fiktívnu testovaciu vzorku do každej z tried, do ktorých trénujeme klasifikátor. Inou možnosťou je pridať fiktívnu vzorku iba do triedy, v ktorej dostávame nulovú čiastkovú pravdepodobnosť.

3.4.2 Support Vector Machines

Support Vector Machines môžeme zaradiť medzi binárne klasifikátory, ktoré sa snažia nájsť oddeľovaciu hranicu medzi dvomi triedami tak, že hranica je čo najviac vzdialená od hraničného bodu, ktorý spadá do jednej alebo druhej triedy. Hraničné body sa označujú ako *"support vectors"*. Jednotlivé vzorky, ktoré táto metóda klasifikuje, si môžeme predstaviť ako skupiny bodov v N dimenzionálnom priestore, ktoré sú oddelené viditeľne širokou medzerou a práve uprostred nej sa snaží metóda určiť N-1 dimenzionálny útvar, ktorý tieto skupiny oddeľuje. V 3D priestore, ktorý je reprezentovaný dátami s tromi atribútmi teda metóda nájde 2D plochu. Metódu je možné použiť aj bez anotovaného trénovacieho korpusu dát, resp. čiastočne anotovaného a to tak, že metóda skúša sama rozdeliť vstupnú vzorku dát do tried.



Obrázok 1: Grafické znázornenie hľadania oddeľovacej priamky technikou SVM [10]

Na obrázku môžeme vidieť vzorky v 2D priestore, t. j. každá vzorka má dva atribúty. Vzorky sú rozdelené do dvoch tried, označených bielou a čiernou farbou. Ďalej môžeme vidieť 3 úsečky, ktoré 2D priestor rozdeľujú. Úsečka H1 nie je správna, pretože ani správne nerozdeľuje vzorky do im priradených tried. Úsečka H2 správne oddeľuje triedy vzoriek, ale vzdialenosť medzi krivkou a bodmi tried nie je maximalizovaná. Korektná úsečka je H3.

Formálne môžeme označiť množinu vzoriek ako:

$$D = \{ (x_i, y_i) \mid x_i \in R^N, y_i \in \{-1, 1\} \} \quad 14$$

kde N je počet dimenzií priestoru a -1, resp. 1 označujú triedy, do ktorých kategorizujeme.
Metóda potom hľadá funkciu:

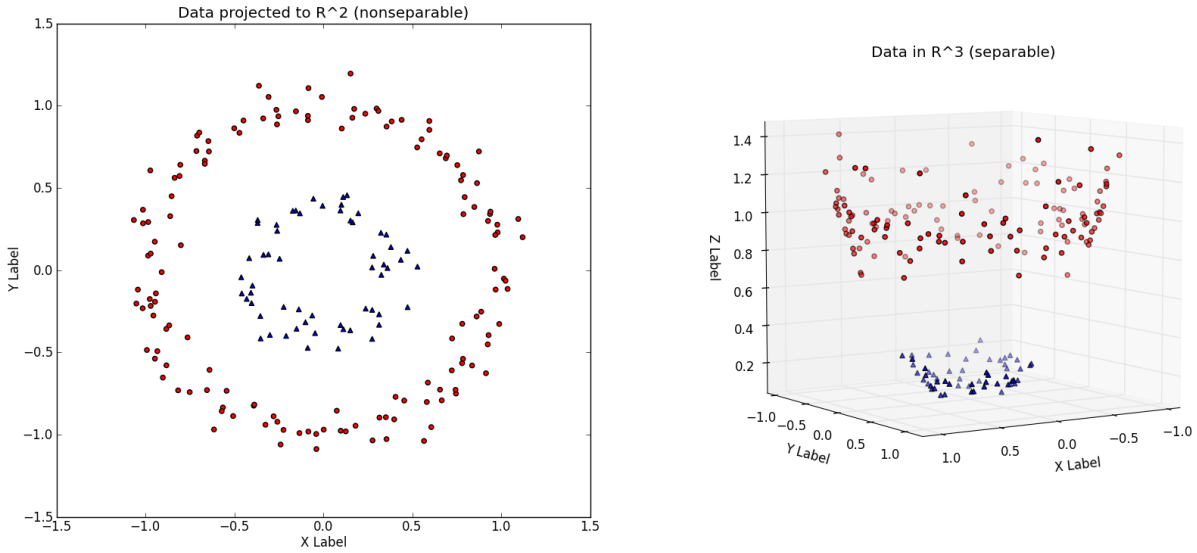
$$h(x_i) = sign\{w \cdot x + b\} \quad 15$$

kde funkcia *sign* vracia +1, ak je vstup kladný a -1 v opačnom prípade. A skalárny súčin je definovaný ako:

$$A \cdot B = ||A|| * ||B|| * \cos\theta \quad 16$$

kde $||X||$ je veľkosť vektoru a θ je uhol, ktorý vektory zvierajú. Parameter *b* špecifikuje posun hľadaného N-1D priestoru od začiatku súradnicovej sústavy vzhľadom na vektor *w*. [11]

Pôvodne táto metóda pracuje s konečným počtom dimenzií. Často však môžeme naraziť na dáta, ktoré nie sú lineárne separovateľné s daným počtom dimenzií. Z tohto dôvodu sa vytvoril nový prístup, kde sa dané vzorky namapujú do viac dimenzionálneho priestoru. Mapovanie väčšinou zabezpečuje mapovacia funkcia, ktorá však musí dbať na zachovanie kolerácie dát vo viac dimenzionálnom priestore. V literatúre sa tento prístup označuje ako *"kernel trick"*. Funkcia sa spolieha na skalárny súčin vektorov, ktorý sme už definovali. [12]



Obrázok 2: Znázornený Kernel trick [13]

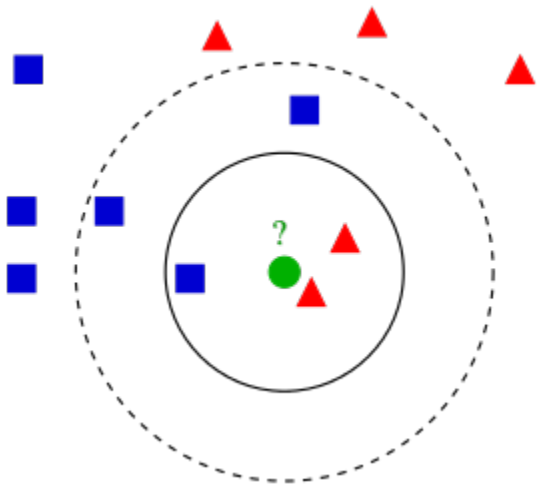
Metóda neprodukuje nijaký výstup modelu, ktorý by bol človek schopný verifikovať, preto je vynakladané úsilie na extrakciu pravidiel z vytvorených interných modelov.

3.4.3 k-Nearest Neighbours

Metóda známa skratkou *kNN* patrí medzi základné a najjednoduchšie klasifikačné a regresné metódy vôbec. Základná myšlienka tejto metódy spočíva v dostupnosti trénovacích dát, ktoré majú priradenú triedu. Na základe početnosti tried *k* najbližších susedných prvkov nového prvku, sa rozhodneme do ktorej triedy nový prvok patrí. Pravidlo k najbližších susedov, ktoré sa používa na klasifikáciu bolo prvýkrát predstavené už v roku 1951 [14]. Častým vylepšením tohto algoritmu je priradenie váh jednotlivým susedným prvkom a ich príspevkom na výsledok na základe vzdialenosti od prvku určeného na klasifikáciu. Najčastejšou funkciou pre určenie váhy prvku je $\frac{1}{d}$, kde *d* je vzdialenosť susedného prvku od prvku určeného na klasifikáciu. Táto funkcia jednoducho priraďuje menšiu váhu

vzdialenejším prvkom a väčšiu váhu bližším prvkom. Nevýhodou *kNN* je, že je náchylná na lokálnu štruktúru dát a veľmi závisí na tom, ako dobre sú dáta separovateľné.

Trénovaciu množinu prvkov si môžeme predstaviť ako vektory v p-dimenzionálnom priestore s priradenými triedami. *K* je používateľom zvolená konštanta, ktorú je možné empiricky odvodiť. Všeobecne platí, že vyššie hodnoty *k* redukujú šum pri klasifikovaní, avšak zmenšujú hranice medzi jednotlivými triedami. Ak zvolíme $k = 1$, metóda sa označuje ako *"nearest neighbour"*.Vzdialenosť medzi jednotlivými prvkami môžeme určiť rôznymi metrikami. Najčastejšie sa používa Euklidovská vzdialenosť, alebo Hammingova vzdialenosť. Konkrétna voľba metriky je závislá na štruktúre dát.



Obrázok 3: Príklad využitia metódy kNN [15]

Na obrázku môžeme vidieť príklad klasifikácie nového prvku s doteraz nepriradenou triedou označený ako zelený kruh s otáznikom. Okolo tohto prvku sú prvky z testovacej množiny, ktorým sme úspešne priradili triedu označenú ako modrý štvorec, alebo triedu označenú ako červený trojuholník. Ak by sme zvolili $k = 3$, to znamená že berieme okolie susedných prvkov vo vnútornom kruhu, zelený prvok by sme označili ako červený trojuholník, keďže sa v danom okolí nachádza viac prvkov tejto triedy. Ak by sme však zvolili $k = 5$, nový prvok by sme označili ako modrý štvorec, pretože vo vonkajšom kruhu sa nachádza viac prvkov tejto triedy. Príklad veľmi dobre znázorňuje problémy tejto metódy, ktoré vznikajú na dátach, ktoré nie sú jednoznačne separovateľné a hranice sa pri danom výbere ich atribútov prelínajú.

Problémy pri tejto klasifikačnej metóde môžu nastať, ak je distribúcia jednotlivých tried v trénovacej množine vychýlená. Prvky triedy, ktorá má väčšie zastúpenie v dátach negatívne ovyplvňujú klasifikáciu nových prvkov, pretože sa všeobecne častejšie vyskytujú v množine [16]. Jeden z možných spôsobov riešenia tohto problému je už spomenuté priradenie váh jednotlivým susedom v závislosti na vzdialenosti od nového prvku. Ďalšou možnosťou je použitie samo-organizujúcich sa máp, známych tiež ako Kohonenove siete [17]. Na trénovacích dátach sa najprv vytvorí Kohonenova sieť, na ktorú sa následne aplikuje *kNN* algoritmus.

3.5 IE systémy “bez učiteľa”

Predošlé metódy boli vysoko závislé na vopred anotovanom veľkom korpuse trénovacích dát. Prostredníctvom tohto korpusu si natrénovali svoj vnútorný model, na základe ktorého mohli následne klasifikovať s väčšou či menšou presnosťou nové dokumenty. Hlavná nevýhoda týchto metód je predovšetkým v potrebe spomínaného anotovaného korpusu a jeho anotovaní. Táto práca je vždy veľmi náročná a vyžaduje si množstvo času. Práve z tohto dôvodu sa začal klásť dôraz na iné metódy extrahovania informácií, ktoré nepotrebujú takýto anotovaný korpus dokumentov ale stačí im na začiatku pomerne malá množina správnych údajov (*“seed”)*. Metódam, ktoré pracujú bez potreby predošlého zdĺhavého anotovania korpusu dát sa budem venovať podrobnejšie, nakoľko by som chcel jednu z týchto metód priamo využiť aj v tejto práci.

Začiatočná množina údajov je väčšinou vo forme množiny dvojíc. Povedzme, že chceme vyhľadávať množinu dvojíc *(autor, kniha)*. Vstupom tejto metódy bude teda množina dvojíc *autor/kniha*. Na základe týchto prvotných dát sa následne hľadajú výskyty údajov v dokumentoch a z kontextu, v ktorom sa vyskytujú je možné odvodiť nové pravidlá vyhľadávania, na základe ktorých zas nájdeme nové výskyty a nové pravidlá. Táto metóda sa v literatúre často označuje ako bootstrapping. [18]

3.5.1 DIPRE

Jednou z metód, ktoré používajú takúto techniku hľadania štruktúrovaných informácií v neanotovanom korpuse dát je metóda *DIPRE (Dual Iterative Pattern Relation Expansion)*. V [19] sa autori zamerali na úlohu získavania dvojíc *(autor, kniha)* s počiatočnou množinou len 5 dvojíc autor, kniha. Metódu testovali na podmnožine 24 miliónov webových stránok o celkovej veľkosti 147 gigabajtov. Množina bola v danom čase súčasťou Stanford WebBase a bola používaná pre výskumný univerzitný projekt “Google search”.

Problém, ktorý metóda rieši sa dá formálne definovať takto. Nech *D* je databáza dokumentov, v ktorých chceme vyhľadávať informácie. Nech $R = r_1, \ldots, r_n$ je výsledná relácia, ktorú sa snažíme nájsť. Ďalej pracujeme s predpokladom, že každá dvojica $t \in R$ sa aspoň raz vyskytuje v *D.* Ak sa nám podarí vytvoriť aproximáciu *R’*, potom môžeme vyjadriť úplnosť a chybovosť ako:

$$Úplnosť = \frac{|R' \cap R|}{|R|} \qquad 17$$

$$Chybovosť = \frac{|R' - R|}{|R'|} \qquad 18$$

Všeobecne sa snažíme maximalizovať úplnosť a minimalizovať chybovosť avšak väčší dôraz sa kladie na znižovanie chybovosti, keďže systém, ktorý má úplnosť 80% je akceptovateľný, ale systém, ktorý má chybovosť +10% je už takmer nepoužiteľný.

Základná myšlienka, na ktorej je metóda DIPRE postavená je myšlienka duality. Dualita vychádza práve z predpokladu, že ak máme k dispozícii kvalitné vzory na vyhľadávanie výskytov dvojíc množiny *R* tak je tiež vysoký predpoklad nájdenia kvalitných dát. Na druhej strane ak máme k dispozícii kvalitné dáta, tak na základe ich výskytov v dokumentoch a kontextu v akom sa vyskytujú, je možné odvodiť kvalitné vzory pre vyhľadávanie týchto dát. Princípom duality je teda hypotéza, že z kvalitných dát vieme odvodiť kvalitné vzory a s kvalitnými vzormi vieme vyhľadať kvalitné dáta.

Metóda DIPRE sa dá zhrnúť do niekoľkých algoritmických krokov:

- Začneme s malou množinou kvalitných dát, ktoré nám poskytne používateľ
- Nájdeme všetky výskyty daných vstupných dát a uložíme si ich výskyty spolu s kontextom v ktorom sa vyskytli
- Vygenerujeme vzory z nájdených výskytov na základe podobného, alebo rovnakého kontextu výskytu. Pri tomto kroku je veľmi dôležité dbať nejakým spôsobom nato, aby vygenerované vzory neboli príliš všeobecné, pretože by nám metóda následne našla aj nesprávne výskyty dát.
- Vyhľadáme výskyty dát na základe novo-vygenerovaných vzorov.
- Ak nie je množina dát dostatočne veľká, opakujeme cyklus s novou množinou vzorov.

Výskyt dát na základe určitého vzoru je definovaný ako:

$$(autor, titul, poradie, url, prefix, stred, suffix) \quad (19)$$

autor - meno autora
titul - názov jeho diela
poradie - boolean hodnota, ktorá označuje v akom poradí sa vyskytol autor a titul
url - URL adresa dokumentu, v ktorom sa našiel výskyt
prefix - *m* znakov, ktoré predchádzajú autora (pri testoch bolo $m = 10$)
stred - text medzi autorom a titulom
suffix - *m* znakov, ktoré nasledujú titul, resp. autora čo závisí od poradia výskytu

Metódu autori overili na 5 iteráciách algoritmu, ktorý inicializovali len prostredníctvom sady 5 dvojíc autor, kniha. Jednotlivé iterácie algoritmu spúšťali na rôzne veľkých podmnožinách databázy webových

stránok. Algoritmus po 5 iteráciách našiel 15257 unikátnych kníh (s príslušným autorom) len s veľmi nízkym počtom nesprávnych dát. V predposlednej iterácii bolo potrebné ručne odstrániť 242 nesprávnych výskytov slova "Conclusion", ktoré sa vyskytovalo ako autor knihy. Toto bola jediná ľudská intervencia ktorú autori vykonali.

3.5.2 Snowball

Na predošlú metódu DIPRE, naviazali v práci [20] metódou Snowball. Systém bol v danom článku overovaný na úlohe hľadania organizácií a ich sídiel. Vo výsledku však overovali iba to, či systém správne našiel sídlo v rámci U.S. alebo mimo U.S.[6] Najviac vylepšili hlavne definíciu a reprezentáciu vzorov tak, aby zachytili čo najviac výskytov, no zároveň neboli príliš všeobecné. Pridali mechanizmy na ohodnocovanie vygenerovaných vzorov a aj nájdených výskytov na základe týchto vzorov tak, aby mohli v priebehu extrahovania odstraňovať vzory a výskyty, ktoré by mohli v budúcnosti zhoršiť celkové metriky systému. Okrem toho vyvinuli aj mechanizmus na automatickú analýzu presnosti a úplnosti ich systému, ktorý však pracoval s "referenčnou" tabuľkou, získanou zo štruktúrovaných dát.

Systém na začiatku dostane zopár správnych dvojíc *(organizácia, sídlo)*. Pre každú z týchto dvojíc systém vyhľadá v korpuse dokumentov jej výskyt a extrahuje kontext v akom sa nachádzal, z ktorého následne môže skonštruovať vzor. Výraznou zmenou oproti metóde DIPRE je, že Snowball pracuje s podsystémom, ktorý dokáže určiť či daný výraz predstavuje organizáciu, alebo sídlo na základe určitých pravidiel. Tento podsystém je vlastne *"Named Entity"* tagger. O NER úlohe som písal na začiatku tejto práce. Príklad vzoru je potom *"<ORGANIZACIA> v <SIDLO>"*. Tento vzor teda nebude hľadať všetky výskyty slov spojených predložkou "v", ale len také spojenia, kde NER podsystém označí prvé spojenie ako organizáciu a posledné spojenie ako sídlo. Príklad: *"Štátny úrad pre kontrolu liečiv v Bratislave"*.

Vzory, ktoré Snowball používa sú navrhnuté tak, aby mohli nájsť výskyty s väčšími alebo menšími obmenami, napr. chýbajúca čiarka, spojka alebo iný menší rozdiel neznamená že vzor daný výskyt nenájde. Vo vzoroch sa reprezentuje ľavý, stredný a pravý kontext vektorom váh pre jednotlivé slová, ktoré sa v kontexte môžu nachádzať. Formálne je vzor pätica: *<ľavý, ltag, stred, rtag, pravý>*, kde *ltag/rtag* sú značenia pre NER podsystém, aké typy spojení hľadáme. *Ľavý, stred a pravý* sú vektory váh prislúchajúce jednotlivým slovám, resp. tokenom. Príklad vzoru:

< {(spoločnosť, 0.3)}, ORGANIZACIA, {(v, 0.4), (pri, 0.1)}, LOCATION, {} > (20)

Príklady výskytov, ktoré vzor nájde:

[6] U.S. - United States je označenie pre Spojené štáty americké

"... spoločnosť Eset v Bratislave ..." 21

"... SUKL v Bratislave ..." 22

"... Billa pri Bratislave ..." 23

Táto metóda ďalej pracuje s funkciou, ktorá vyhodnocuje stupeň zhody. Majme dva rôzne výskyty: *t = <lt, t1, mt, t2, rt>* a *p = <lp, t3, mp, t4, rp>,* funkcia zhody je potom definovaná takto:

$$\text{Zhoda(t, p)} = lt*lp + mt*mp + rt*rp \quad (24)$$

v prípade, že sa *t1 = t3 a t2 = t4.* V opačnom prípade sa funkcia zhody rovná 0. Funkcia zhody sa následne používa pri rozhodovaní, či sa daný výskyt bude v ďalších iteráciách brať do úvahy alebo nie. Okrem tejto metriky používa metóda aj iné metriky na ohodnocovanie výskytov a nových vzorov. Toto ohodnocovanie je jedným z kľúčových vylepšení oproti metóde DIPRE.

Pri overovaní metódy sa autori spoliehali na existenciu referenčnej množiny dvojíc *(organizácia, sídlo)*. Takúto referenčnú tabuľku je však v mnohých reálnych prípadoch nemožné, alebo len veľmi náročné získať. Predpokladajme však, že daná referenčná množina je k dispozícii a označme ju *R*. Extrahovanú množinu záznamov si označme ako *E*. Následne si vytvoríme pomocnú množinu, pre lepšiu interpretáciu metrík. Pomocnú množinu označíme ako *P* a skonštruujeme ju nasledovne:

$$P = \{(o, l, l') \mid (o, l) \in R \wedge (o', l') \in E \wedge o \simeq o'\} \quad (25)$$

V definícií je určitá vágnosť, ktorú je potrebné špecifikovať a tá sa skrýva v symbole ≃. Neformálne sa dá povedať, že tento symbol vyjadruje, že dva rovnaké alebo podobné reťazce predstavujú jednu a tú istú spoločnosť. Príklad: *"Eset"* a *"Esetu"*. V práci použili špeciálny nástroj Whirl [25], ktorý bol vyvinutý v AT&T Research Laboratories a má za úlohu zjednocovať podobné textové reťazce. Pomocou množín *R* a *P* si následne môžeme definovať *presnosť* a *úplnosť* systému ako:

$$\text{Úplnosť} = \frac{\sum_{i=0}^{|P|}[l_i = l_i']}{|R|} * 100\ (\%) \quad (26)$$

$$Presnosť = \frac{\sum_{i=0}^{|P|}[l_i = l_i']}{|P|} * 100\ (\%) \quad (27)$$

Na základe predošlých metrík autori tejto metódy vyhodnotili presnosť a úplnosť systému na sade 178 000 trénovacích článkov a 142 000 článkov určených na testovanie. Články boli získané z *North American News Text Corpus LDC*[7]. Presnosť systému bola následne vyhodnotená na 76% a jeho úplnosť na 45%, čo predstavuje lepšie výsledky oproti metóde DIPRE, avšak je to stále nedostatočné pre projekt, ktorý pracuje s medicínskymi údajmi. Zaujímavosťou je, že autori pri testoch zistili, že interpunkcia v texte výrazne zlepšovala presnosť aj úplnosť tejto metódy. [17]

[7] https://www.ldc.upenn.edu/

4 Metódy reprezentácie čiastočne štruktúrovaných dát

4.1 Sémantický web

Internet bol pôvodne určený pre ľudí. V dnešnej dobe okrem ľudí, pristupuje k informáciám a službám na internete čoraz viac počítačov. Informácie na internete sú dobre čitateľné pre ľudí, avšak počítače týmto informáciám rozumejú vo väčšine prípadov iba veľmi ťažko. Riešením by bolo anotovať všetky dokumenty na webe metadátami takým spôsobom, aby boli informácie jednoducho spracovateľné aj počítačom. [21]

Sémantický web je termín pre rozšírenie Webu podľa určitých špecifikácií, ktoré spravuje World Wide Web Consortium[8] (W3C). Za otca sémantického webu sa považuje pán Tim Berners-Lee, ktorý o sémantickom webe hovorí ako o webe dát, namiesto webe dokumentov. Hlavný dôraz sa pritom kladie na možnosť spracovať akýkoľvek zdroj počítačom. V budúcnosti by sme tak mohli získať obrovské množstvo kvalitných informácií v štruktúrovanej podobe bez potreby systémov na extrakciu informácií. Namiesto ľudí, by mohli po webe vyhľadávať dáta a služby iba počítače. [22]

Neformálne ide o spôsob, ktorým do HTML dokumentu pridáme metadáta, ktoré sú jednoducho čitateľné strojom a opisujú dáta, ku ktorým sú priradené. V praxi sa na tieto účely používa RDF, OWL alebo XML. Ako dopytovací jazyk sa používa SPARQL. So sémantickým webom sa často spája aj pojem Web 3.0. [23]

4.1.1 Resource Description Framework (RDF)

Nasledujúce informácie som čerpal hlavne z [21]. RDF predstavuje prístup k definícii a spracovaniu metadát tak, aby boli prenositeľné medzi aplikačnými doménami a aby ich bolo možné strojovo spracovať. Môže sa použiť napríklad pre lepšie vyhľadávanie na webe, alebo na tvorbu katalógov produktov, digitálnych knižníc, inteligentných softvérových agentov a ďalších programov.

Hlavným cieľom RDF je špecifikovať mechanizmus na opis dát tak, aby bol nezávislý na aktuálnej aplikačnej doméne a zároveň aby neboli metadáta tvorené pre konkrétnu aplikačnú doménu. Definície by mali byť teda doménovo nezávislé, no zároveň by mali poskytovať mechanizmus pre opis informácií o akejkoľvek doméne. Vo veľkej miere tento mechanizmus pracuje so systémom tried, ktoré tvoria schémy a sú hierarchicky organizované. Každú triedu je možné rozšíriť pre potreby danej

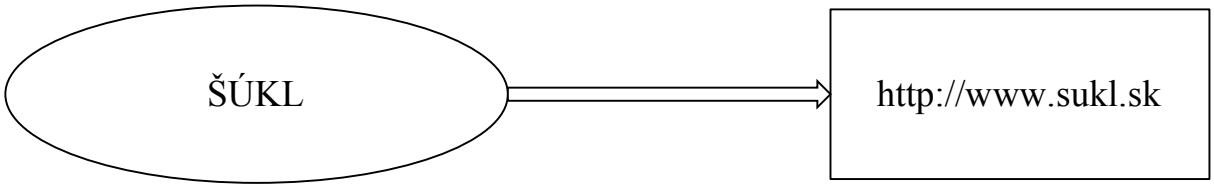
[8] http://www.w3.org/

aplikácie ako to poznáme z objektovo-orientovanej programovej paradigmy. Nové triedy je možné odvádzať aj na základe viacerých rodičovských tried z rôznych schém a zdrojov.

Základný dátový model pozostáva z troch typov objektov:

- Zdroje predstavujú veci, ktoré sú opisované RDF výrazmi. Zdrojom môže byť napríklad webová stránka, časť webovej stránky, ktorá opisuje určitý objekt, ako napríklad produkt, alebo aj celá kolekcia webových stránok. Zdroj môže mať aj fyzickú podobu, ako napríklad tlačená kniha.
- Vlastnosti sú charakteristiky, atribúty alebo relácie ktoré opisujú zdroje. Každá vlastnosť má špecifický význam, nadobúda určité hodnoty a má určitý súvis s ostatnými vlastnosťami.
- Údaje (*"statements"*) sú tvorené zdrojmi, ktoré majú pomenované vlastnosti spolu s ich hodnotami. Jednotlivé časti sa nazývajú *subjekt*, *predikát* a *objekt* v tomto poradí. Objektom môže byť ďalší zdroj, alebo RDF literál, teda číslo alebo reťazec.

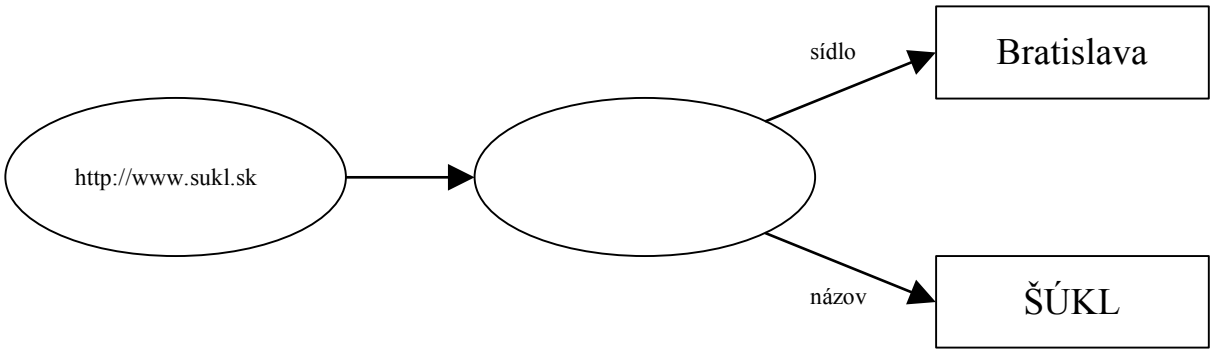
"*ŠÚKL spravuje stránku http://www.sukl.sk*" 28



Obrázok 4: Grafická reprezentácia údaju

Pre daný údaj je "ŠÚKL" subjekt, "spravuje" je predikát a URL webovej stránky je objekt. Smer šípky medzi subjektom a objektom je veľmi dôležitý. Šípka vždy začína pri subjekte a končí pri objekte. Všeobecný zápis takéhoto diagramu je "SUBJEKT má PREDIKÁT OBJEKT". O danom subjekte môžeme vedieť viac informácií, napríklad:

"Úrad, ktorý sa označuje skratkou ŠÚKL a sídli v Bratislave, spravuje stránku http://www.sukl.sk" 29



Obrázok 5: Grafické znázornenie zloženej anonymnej entity

Vlastnosť spravovateľ v tomto prípade vystupuje ako zložená entita, ktorá má vlastnosti sídlo a názov. Keďže vo vete nebolo presne špecifikované ako má entita vystupovať, tak je možné ju zapísať ako anonymnú entitu, ktorá sa značí prázdnym oválom.

Zápis RDF zdrojov je veľmi podobný jazyku XML. RDF poskytuje možnosť využiť dva typy syntaxe: *serializačnú* syntax, ktorá poskytuje všetky možnosti RDF zápisov a *zhustenú* syntax, ktorá obsahuje prídavné štruktúry a zápisy pre zvýšenie celkovej kompaktnosti. Od RDF interpretov sa očakáva, že budú schopné analyzovať oba typy syntaxí. Pre ďalšie informácie, by som čitateľa odkázal na dokument, ktorý som uvádzal v úvode ako zdroj informácií pre túto kapitolu.

4.1.2 SPARQL

Informácie v tejto kapitole som čerpal hlavne z dokumentu [24]. Tento dokument špecifikuje syntax a sémantiku dopytovacieho jazyka SPARQL, pre dopytovanie do dát vo forme RDF. SPARQL je možné použiť v prípade, že dáta sú priamo uložené vo formáte RDF, ale aj v prípade že ich zdroj je v úplne inom formáte, povedzme CSV a cez ďalší program sú zobrazené v RDF formáte. Tento dopytovací jazyk podporuje dopytovanie konjunkcií a disjunkcií dát, testovanie premenných a aplikovanie rôznych obmedzení na požadované dáta. Výsledok SPARQL dotazu môže byť množina alebo ďalší RDF graf.

Väčšina SPARQL dotazov obsahuje trojice vzorov, ktoré sa nazývajú základné grafové vzory (*basic graph patterns)*. Tieto trojice sú veľmi podobné RDF údajom, avšak subjekt, predikát aj objekt môžu vystupovať ako premenná. Tieto vzory sa zhodujú s RDF dátami, ak môžeme jednotlivé časti RDF údaju nahradiť za premenné v dopyte a výsledné dáta sa zhodujú. Ukážeme si jednoduchý príklad:

Dáta: "<example.com/books/1> <example.com/elements/title> "SPARQL Tutorial" . " 30

```
Dotaz: SELECT ?title WHERE {                                    31
        <example.com/books/1> <example.com/elements/title>
?title .
}

Výsledok: "SPARQL Tutorial"                                     32
```

V dotazoch je možné špecifikovať aký konkrétny dátový typ, alebo jazyk je hľadaný. Môžeme tiež použiť príkazové slovo CONSTRUCT, ktoré nám zabezpečí, že výsledok dotazu bude ďalší RDF graf. V tele príkazu CONSTRUCT je možné špecifikovať tvar, v akom výsledný RDF graf očakávame. RDF dopyty poskytujú tiež možnosť filtrovať výsledky príkazom FILTER, v ktorého tele môžeme špecifikovať napríklad regulárny výraz hľadaného textu. Okrem filtrovania textu je tiež možné špecifikovať rozsah vyhľadávaných číselných hodnôt. Ďalšie príklady a možnosti tohto dopytovacieho jazyka je možné nájsť v dokumente, ktorý som uviedol v úvode.

4.1.3 OWL

Alternatívou k RDF je jazyk OWL (Web Ontology Language), ktorý slúži na reprezentáciu ontológií. Ontológia je spôsob, ktorým sa dá formálne vyjadriť taxonómia, resp. určitá hierarchia elementov na základe ktorých prebieha klasifikácia. Ontológie zahŕňajú, triedy, inštancie, atribúty, vzťahy a ďalšie prvky. OWL je v súčasnosti vo verzii 2, ktorej špecifikácia bola zverejnená v roku 2009 a zastrešená je W3C organizáciou.

4.2 Relačné databázy

Ďalšou možnosťou ukladania dát je uloženie do relačných databáz vo forme tabuliek. Dáta v relačných tabuľkách musia mať pevnú a presne danú štruktúru, aby sme ich mohli úspešne uložiť. Relačná databáza je databáza dát v elektronickej podobe. V dnešnej dobe sa vo svete relačných databáz používa niekoľko systémov pre správu takýchto databáz. Okrem asi najrozšírenejšieho MySQL sa často využíva jeho alternatíva PostgreSQL a tiež zjednodušená verzia, ktorá nepotrebuje server, ale všetky dáta ukladá do súborov, SQLite. Všetky relačné databázy vo veľkej miere využívajú štruktúrovaný dopytovací jazyk SQL, prostredníctvom ktorého je možné vytvárať nové tabuľky v rámci databázovej schémy, upravovať, mazať ako tabuľky tak aj dáta uložené v týchto tabuľkách. Relačné databázy nám poskytujú robustnosť v podobe transakcií, ktoré nám zabezpečujú atomicitu prístupu, konzistenciu dát, izolujú jednotlivých používateľov a zabezpečujú perzistentnosť ukladania dát.

4.2.1 MySQL

Patrí medzi najrozšírenejšie open source relačné databázové riadiace systémy (RDBMS). MySQL je dnes vlastnený spoločnosťou ORACLE a je voľné dostupný. Spoločnosť ORACLE poskytuje aj platené verzie tohto systému s rozšírenou funkčnosťou. MySQL sa používa hlavne pri tvorbe webových aplikácií a je súčasťou veľmi rozšíreného LAMP webového súboru programov (Linux, Apache, MySQL, PHP). Systém je okrem Windowsu dodávaný bez grafického používateľského rozhrania a je možné s ním pracovať cez konzolové príkazy.

Systém je naprogramovaný v jazyku C a C++, ako parser príkazov sa používa yacc, ale používajú vlastný lexikálny analyzátor. [25] Na svojich stránkach majú zverejnený veľmi podrobnú a kvalitnú dokumentáciu, ktorá okrem používania príkazov MySQL opisuje aj vnútornú štruktúru systému pre programátorov. Alternatívou k tomuto systému je tiež jeho najpopulárnejší fork MariaDB[9].

4.2.2 PostgreSQL

PostgreSQL prináša alternatívu k predošlému systému. Je to objektovo-relačný databázový systém, ktorý je tiež open source. Má za sebou viac ako 15 rokov aktívneho vývoja. Rovnako ako MySQL je ho možné používať na širokej škále operačných systémov. Enterprise verzia tohto systému poskytuje sofistikované nástroje ako kontrolu konkurentného viac verziového prístupu, možnosť vrátiť sa na dané miesto v čase pri obnovení, asynchrónnu replikáciu, vnorené transakcie a mnohé ďalšie.

PostgreSQL má v sebe implementovaný tzv. GiST (Generalized Search Tree) indexovací systém, ktorý v sebe spája niekoľko vyhľadávacích algoritmov vrátane, B-stromu, B+-stromu, R-stromu, ohodnoteného B+-stromu a mnohých ďalších. Okrem toho poskytuje tiež rozhranie k vytváraniu vlastných dátových typov a tiež rozšíriteľných dopytovacích metód, ktoré umožňujú vyhľadávať v týchto typoch. [26]

4.2.3 Porovnanie MySQL a PostgreSQL

S predošlými systémami sa spája nespočetné množstvo porovnávaní. Oba systémy majú svojich zástancov a odporcov. Vo všeobecnosti sa dá povedať, že väčšina tvrdí, že MySQL je rýchlejšie a jednoduchšie na použitie, avšak PostgreSQL poskytuje viac možností. V nasledujúcej kapitole by som chcel stručne zhrnúť jedno z týchto porovnaní. [27] MySQL začalo v poslednej dobe podporovať ANSI štandardy avšak PostgreSQL malo štandardy implementované od svojho začiatku. Oba systémy podporujú zamykanie pri transakciách na úrovni záznamov tabuliek. MySQL malo od začiatku problémy s vnorenými dopytmi, v poslednej dobe sa to značne zlepšilo, avšak stále nepodporuje FULL

[9] https://mariadb.com/

OUTER JOINS na rozdiel od PostgreSQL. Okrem toho PostgreSQL podporuje ukladanie JSON dokumentov, ktoré sú veľmi vhodné pri ukladaní slabo štruktúrovaných dát, ako v našom prípade. PostgreSQL je tiež absolútne voľne dostupné a použiteľné aj v komerčnej sfére vďaka benevolentnému licencovaniu na štýl MIT.

4.2.4 SQLite

SQLite[10] je knižnica, ktorá implementuje transakčný databázový nástroj, ktorý nepotrebuje žiadnu inštaláciu ani nastavenia, server, spustený proces, používateľov ani práva. Ako píšu na svojej stránke "SQLite just works". Tento projekt je voľne dostupný a použiteľný na akékoľvek použitie, či už súkromné účely alebo v komerčnej sfére. SQLite je tiež najviac používaný databázový systém na celom svete. Používa sa v smartfónoch s Androidom, iPhone a iOS zariadeniach, počítačoch s operačným systémom Windows 10. Ďalej vo webových prehliadačoch Firefox, Chrome, Safari, aplikáciách Skype, iTunes, Dropbox, TurboTax, QuickBooks, PHP, Python a ďalších zariadeniach pre televízne a káblové vysielanie.

Hlavnou výhodou SQLite je, že nepotrebuje na svoj beh nijaký spustený proces ani server. Namiesto toho zapisuje dáta priamo do súborov na disku. Tieto súbory sú ukladané tak, aby mohli byť bez akejkoľvek zmeny otvorené a znovu použité na akomkoľvek operačnom systéme nezávisle na tom, či je 32 alebo 64 bitový a či používa malý alebo veľký endian pri ukladaní čísel. Okrem toho je tento systém veľmi nenáročný na pamäť. Celá knižnica pri správnom preložení zaberá menej ako 500kB miesta a pri minimálnej implementácii stačí dokonca menej ako 300kB. Aj z týchto dôvodov je to obľúbený databázový nástroj v zariadeniach ako sú mobilné telefóny, PDAčka, MP3 prehrávače a iné.

SQLite poskytuje tiež vysokú stabilitu a odolnosť voči programátorským chybám vďaka automatickým testom, ktoré spúšťajú milióny testov so stovkami miliónov SQL dopytov a každá vývojová vetva zaručuje 100% pokrytie testami.

[10] https://www.sqlite.org/

5 Analýza dokumentov príbalových letákov pre lieky

Štátny úrad pre kontrolu liečiv má na svojej stránke[11] zverejnený zoznam registrovaných liekov, ktoré sú schválené a dostupné v rámci Slovenskej republiky. Tento zoznam je pravidelne aktualizovaný podľa toho, či boli schválené nové lieky, alebo odstránené staré nevyhovujúce. Na stránkach je dostupný vyhľadávač liekov. Lieky sú uložené pre každé balenie zvlášť. Ak má liek tabletovú aj tekutú formu, pre obe môžeme nájsť samostatný záznam. Na konkrétnej stránke pre daný liek je dostupný jeho kód, registračné číslo, názov lieku, indikačná skupina, spôsob podania, viazanosť na lekárky predpis a dátum registrácie. Všetky tieto údaje sú dostupné v štruktúrovanej tabuľkovej forme. Okrem toho sú ku každému lieku dostupné dva dokumenty: súhrn charakteristických vlastností (SPC) a príbalová informácia pre používateľov (PIL). Oba tieto dokumenty sú dostupné vo formáte „doc“, resp. „docx“ a obsahujú čiastočne štruktúrovaný text o danom lieku.

Súhrn charakteristických vlastností (SPC) je rozdelený na niekoľko viditeľných odstavcov. Každý odstavec má nadpis, ktorý nasleduje niekoľko riadkov neštruktúrovaného textu. Dokument obsahuje názov lieku, kvalitatívne a kvantitatívne zloženie, liekovú formu, klinické údaje, konkrétne terapeutické indikácie (kedy liek používať), dávkovanie a spôsob používania (ako liek používať), kontraindikácie a ďalšie iné informácie.

Príbalové informácie pre používateľov (PIL) sú tiež rozdelené do niekoľkých odstavcov. Tento dokument slúži aj ako príbalový leták k danému lieku, ktorý sa prikladá do balenia. V dokumente je možné nájsť informácie o tom, ako sa liek používa, kedy je vhodné liek užívať a naopak kedy sa to neodporúča, akým spôsobom sa má liek užívať, aké má vedľajšie účinky, ako je ho potrebné uchovávať a ďalšie informácie potrebné pre koncového spotrebiteľa.

Za účelom analýzy bolo stiahnutých 32 SPC dokumentov. Dokumenty boli programovo prevedené na formát HTML a následne analyzované štandardnými UNIX programami (wc, grep, ...). Všetky dokumenty obsahujú kapitolu „Kvantitatívne a kvalitatívne zloženie“, ktorá je označená poradovým číslom „2“ a kapitolu „Terapeutické indikácie“, ktorá je označená poradovým číslom „4.1“. Práve tieto kapitoly môžu sú kľúčové pre naše vyvíjané riešenie, nakoľko prvá spomenutá obsahuje aktívne a pomocné látky, ktoré dané liečivo obsahuje a druhá spomenutá obsahuje stručný a výstižný opis symptómov, pri ktorých sa odporúča daný liek užívať.

Kapitola „Kvantitatívne a kvalitatívne zloženie“ obsahuje názvy účinných a pomocných látok. Tieto látky predstavujú jadro samotného liečiva. To, či bude liek účinkovať, alebo nie závisí hlavne od účinnej látky, ktorú obsahuje. Rôzni výrobcovia vyrábajú liečivá, ktoré obsahujú podobné, resp. rovnaké účinné

[11] www.sukl.sk

látky, no odlišujú sa v rôznych doplnkových. Pre mnohých ľudí by v dnešnej dobe mohlo byť zaujímavé vyhľadať si liečivo od iného výrobcu, ktoré obsahuje tú istú účinnú látku, ale je napríklad lacnejšie, alebo má menej nepriaznivých účinkov (kontraindikácií). Samotné účinné látky sa vyskytujú v takýchto formách: „... obsahuje 125 mg monohydrátu laktózy ...“, „... sa nachádza 20 ml budenzonidu ...“, „... poskytuje inhalačnú dávku 45 mikrogramov salmeterolu ...“. Napriek tomu, že sú všetky výskyty pomerne odlišné, majú spoločný prvok v podobe vyjadrenia množstva. Vo väčšine prípadov sa množstvo vyjadruje ako ČÍSLO_JEDNOTKY. Jednotky, ktoré som v dokumentoch objavil, sú: ml, mg, g, miligramov, miligramy, gramy, mikrogramy, mikrogramov, mikrogramom, mikrogramu, mmol. Aj táto skutočnosť nám pomôže pri hľadaní účinných látok, o tom však budeme písať až v ďalšej kapitole.

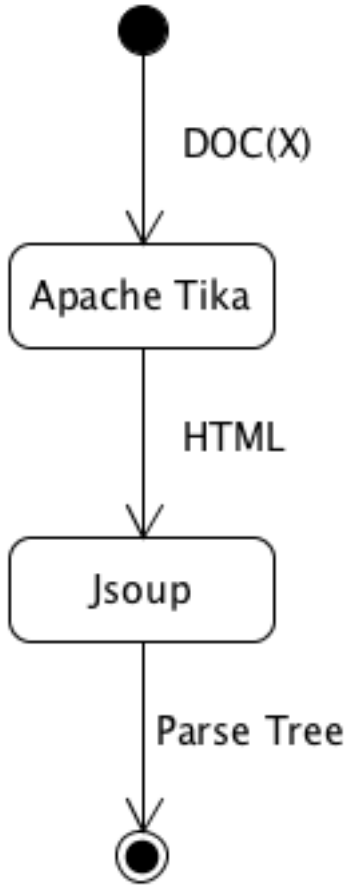
Kapitola „Terapeutické indikácie“ obsahuje vo väčšine prípadov veľmi stručný (3-8 riadkov) opis, v akých prípadoch je vhodné používať liek, aké sú symptómy, pri ktorých sa odporúča používať liek a ako sa liek používa. Túto kapitolu by bolo vhodné „extrahovať“ celú bez ďalších čiastkových extrakcií.

Okrem týchto, je pre nás zaujímavá aj kapitola, ktorá sa zaoberá dávkovaním. Bohužiaľ je diverzita spôsobov vyjadrenia dávkovania príliš komplexná vzhľadom na charakter tejto práce. Dávkovanie môže mať niekoľko kategórií (vek, váha, predošlé choroby, choroby v rodine, zdravotný stav, ...). Okrem toho, že v dávkovaní sa vyskytuje mnoho kategórií, sme pri analýze narazili aj na problém opisu dávkovania. V niektorých dokumentoch to bolo formou tabuľky, niekde formou textu, niekde formou odrážok. Už samotné tabuľky mali rôznu formu a povahu.

6 Návrh

V tejto kapitole sa budem venovať opisu knižníc, ktoré sme sa rozhodli použiť pri extrahovaní informácií. Okrem toho navrhneme postup extrahovania. Po dohode s vedúcim práce, sa zameriame hlavne na kapitolu „Kvantitatívne a kvalitatívne zloženie“, ktorá obsahuje účinné látky a na kapitolu „Terapeutické indikácie“, ktorá obsahuje symptómy, pri ktorých je vhodné liek užívať.

6.1 Spracovanie dokumentov



Obrázok 6: Postup spracovania dokumentov

Na obrázku vidíme postup spracovania dokumentov. Ako sme už uviedli, dokumenty sme stiahli vo formátoch .doc a .docx. Všetky stiahnuté dokumenty sme umiestnili do spoločného priečinku, v rámci ktorého môžeme preiterovať cez všetky súbory a každý samostatne spracovať. Každý dokument najprv spracujeme pomocou knižnice Apache Tika. Prácu s touto knižnicou a podrobný opis jej funkcionality a tried, je opísaný v nasledujúcej kapitole. Knižnica nám umožňuje jednoduchým spôsobom konvertovať .doc aj .docx súbory na HTML dokumenty. HTML formát bol zvolený, kvôli možnosti

využitia vizuálnych informácií, ako je napríklad hrubšie a väčšie písmo v nadpisoch, alebo rozdelenie textu na jednotlivé odstavce. Získaný HTML dokument následne predáme knižnici Jsoup, ktorá je tiež podrobne opísaná neskôr. Táto knižnica rozparsuje HTML dokument a poskytuje nám prijateľné rozhranie, cez ktoré môžeme iterovať cez všetky nadpisy, odstavce a každý následne samostatne spracovať.

6.2 Apache Tika

Apache Tika[12] je knižnica, ktorá dokáže extrahovať metadáta a dáta z rôznych typov dokumentov (ppt, xls, pdf, doc, docx, ...). Výhodou tejto knižnice je, že si dokáže sama určiť typ dokumentu, preto program ktorý ju využíva nepotrebuje explicitne poznať typ dokumentu. Dokumenty dokáže interpretovať vo formátoch HTML, XML, JSON, XMP a čistý text. Pre naše účely sme zvolili ako vhodný výstup do HTML, pretože okrem samotného textu dostaneme k dispozícii aj akýsi meta obsah v podobe HTML tagov a CSS tried na základe ktorých by sme mohli vedieť lepšie určiť, kde je nadpis a kde začína a končí daný odstavec. Knižnica sa dá stiahnuť v podobe Java .jar balíka. Následne je možné ju importovať do Java projektu a využívať jej metódy cez verejné rozhrania, ktoré poskytuje. Okrem toho táto knižnica poskytuje aj možnosť interakcie cez konzolu, ktorú sme využili pri analyzovaní štruktúry dokumentov a prvotných experimentálnych testoch.

Formáty Microsoft Office produktov sú vo formátoch OLE 2 a Office Open XML (OOXML). OLE 2 formát, ktorý je starší, je dostupný od verzie Microsoft Office 97 a bol štandardným formátom (.doc) až do vydania Microsoft Office 2007, s ktorým prišiel nový formát OOXML (.docx). Pre extrakciu dát z dokumentov typu OLE 2 je možné použiť triedu OfficeParser. Pre extrakciu dát z dokumentov typu OOXML sa používa trieda OOXMLParser. Obe triedy používajú knižnicu Apache POI, ktorá je súčasťou Apache Tika. Knižnica je dobre dokumentovaná a distribuovaná pod licenciou Apache verzie 2.0.

6.3 Jsoup: Java HTML Parser

Pre rýchlejšiu a pohodlnejšiu prácu s HTML dokumentami, ktoré sa nám podarilo získať prostredníctvom Apache Tika knižnice, sme sa rozhodli použiť knižnicu Jsoup[13]. Jsoup je knižnica pre programovací jazyk Java, ktorá podporuje pokročilú prácu s HTML dokumentami. Poskytuje API na extrakciu a manipuláciu HTML entít za použitia DOMu, CSS a selektorov podobných tým, ktoré poskytuje knižnica jQuery. Knižnica dokáže pracovať s dokumentami na webe, v súboroch alebo

[12] https://tika.apache.org
[13] http://jsoup.org

dokumentami uloženými v textových reťazcoch. Podporuje používanie CSS selektorov alebo prechádzanie DOMu. Okrem toho umožňuje manipulovať s týmito elementami, meniť ich, pridávať a odoberať nové atribúty a ich obsah.

V našej práci sa nám táto knižnica zíde na spracovanie HTML dokumentu, a možnosti spracovať dokument po jednotlivých elementoch. Experimenty ukázali, že väčšinu dokumentov v našej doméne nám Apache Tika vráti ako súbor paragrafov (p tag). V niektorých prípadoch sú nadpisy správne označené ako h1, resp. h2 tag, ale vo väčšine ako paragraf. Cez tieto paragrafy je možné jednoducho iterovať a extrahovať ich obsah, t. j. samotný text spracovávaného dokumentu. Táto knižnica je otvorená, dobre dokumentovaná a prevádzkovaná pod MIT licenciou.

6.4 Návrh metódy extrakcie

V tejto kapitole opíšeme metódy extrakcie účinných látok a terapeutických indikácií.

6.4.1 Extrakcia terapeutických indikácií

Ako sme už opísali v predošlej kapitole, odstavec terapeutických indikácií môžeme extrahovať ako celok. Prakticky to znamená, že dokument vytvorený programom MS Word, prevedieme knižnicou Apache Tika na HTML textový reťazec. Tento reťazec následne odovzdáme knižnici Jsoup, ktorá ho spracuje a umožní nám ďalšiu prácu s HTML paragrafmi. Prejdeme cez všetky paragrafy a ak narazíme na paragraf, alebo nadpis, ktorý obsahuje reťazec „Terapeutické indikácie“ s poradovým číslom „4.1“ začneme všetky nasledujúce paragrafy extrahovať, až kým nenarazíme na ďalší nadpis s ďalším poradovým číslom.

Nadpis “4.1 Terapeutické indikácie“ je možné identifikovať na základe regulárneho výrazu. Po tom, ako zistíme, že nadpis aktuálne spracúvaného odstavca súhlasí s našim regulárnym výrazom si aktualizujeme stav v stavovom automate na základe ktorého vieme, že máme všetky nasledujúce odstavce považovať za súčasť terepeutických indikácií. Ak narazíme na ďalší nadpis, tak jednoducho prejdeme do pôvodného stavu, čo nám indikuje, že ďalšie spracúvané odstavce už nemáme považovať za súčasť terapeutických indikácií.

```
<p class="základný_text_3">
  <b>
    4.1. Terapeutické indikácie
  </b>
</p>
<p class="normálny">
  Nízka hladina vápnika v krvi (podporná liečba pri rachitíde,
  osteomalácii, osteoporóze, pri hojení fraktúr, tetanii,
spazmofílii),
  pokles tlaku vyvolaný akútnym oslabením kontrakčnej sily myokardu
(napr.
  v anestéziológii pôsobením intravenóznych barbiturátov).
</p>
<p class="normálny">
  Akútne alergické choroby, chronické zápalové ochorenia, svrbiace
  dermatózy, mokvajúce a generalizované ekzémy.
</p>
```

Obrázok 7: Ukážka relevantnej časti HTML dokumentu pre Terapeutické indikácie

Na obrázku je zobrazený nadpis a odstavec, v ktorom sa nachádzajú Terapeutické indikácie. Na obrázku nižšie môžeme vidieť pôvodné zobrazenie tohto odstavca v programe MS Word.

4.1. Terapeutické indikácie

Nízka hladina vápnika v krvi (podporná liečba pri rachitíde, osteomalácii, osteoporóze, pri hojení fraktúr, tetanii, spazmofílii), pokles tlaku vyvolaný akútnym oslabením kontrakčnej sily myokardu (napr. v anestéziológii pôsobením intravenóznych barbiturátov).

Akútne alergické choroby, chronické zápalové ochorenia, svrbiace dermatózy, mokvajúce a generalizované ekzémy.

Obrázok 8: Ukážka relevantnej časti pôvodného dokumentu v programe MS Word

6.4.2 Extrakcia účinných látok

Podobne ako v predošlom prípade, aj pri extrakcii účinných látok si najprv vyhľadáme odstavec s označením „Kvantitatívne a kvalitatívne zloženie“ a poradovým číslom „2“. Pri extrakcii účinných látok sme sa inšpirovali prácami [19] a [20]. Metóda, ktorú sme sa rozhodli implementovať sa skladá z niekoľkých krokov. Na začiatku si celý odstavec prevedieme do pracovnej reprezentácie, v ktorej ku každému slovu priradíme tag (anotáciu). Tieto tagy sú dvojakého typu: prvý druh tagov priraďujeme množstevným vyjadreniam množstva účinnej látky na základe reguárnych výrazov. Druhý typ tagov získame prostredníctvom Slovenského národného korpusu. Nad touto pracovnou množinou tagov následne nájdeme pravidlá, na základe ktorých je možné extrahovať účinné látky.

6.4.2.1 Anotovanie textu

Ako sme už naznačili, účinné látky sa vyskytujú najmä pri číslach a jednotkách, ktoré vyjadrujú množstvo danej látky v konkrétnom liečive. Z tohto dôvodu implementujeme pomerne jednoduchý klasifikátor na klasifikáciu týchto množstevných výrazov. Na takúto klasifikáciu je možné použiť regulárne výrazy, nakoľko sa tam vždy nachádza číslo a jednotiek sa nám podarilo identifikovať iba 12. Následne pomocou tohto klasifikátoru môžeme jednoducho previesť text typu „10 mg monohydrátu laktózy“ na text „MNOŽSTVO monohydrátu laktózy“, kde MNOŽSTVO predstavuje akúsi pomocnú značku (tag).

Pri analyzovaní dokumentov sme si všimli, že pri účinných látkach sa okrem kvantitatívneho vyjadrenia množstva a jednotiek vyskytujú dané výrazy v určitých pádoch. Napríklad: „10 mg monohydrátu laktózy“ alebo „10 mg budenzonidu“. V oboch príkladoch môžeme pozorovať, že účinné látky sú v genitíve jednotného čísla. Okrem pádu sa dajú o týchto slovných spojeniach určiť aj slovný druh, paradigma, rod, číslo a pád. Všetky tieto vlastnosti si vieme programovo zistiť o každom slove prostredníctvom Slovenského národného korpusu, ktorý sme opisovali už v druhej kapitole.

S podporou nášho jednoduchého klasifikátoru a Slovenského národného korpusu je možné text previesť z „10 mg monohydrátu laktózy“ na „MNOŽSTVO SSis2 SSfs2“. V niektorých prípadoch sa môže stať, že slovo ktoré chceme takto anotovať, sa nenachádza v korpuse. V takomto prípade môžeme anotáciu nahradiť NEZNÁME. Môže sa to javiť ako pomerne veľký problém, avšak aj takáto informácia má určitú výpovednú hodnotu, nakoľko korpus obsahuje veľké množstvo slov. V prípade, že sa nejaké slovo nenachádza v korpuse, môže naša reprezentácia vyzerať napríklad takto: „MNOŽSTVO NEZNÁME SSis2“.

Z korpusu vieme pre každé slovo získať značku, ktorá v sebe obsahuje všetky vlastnosti spomenuté vyššie. Príklad takejto značky je „SSis2“, ktorá znamená: S – substantívum slovný druh, S – substantivna paradigma, i – mužský neživotný rod, s – jednotné číslo, 2 – genitív.

6.4.2.2 Extrakcia pomocou pravidiel

Po tejto transformácií dostaneme akúsi internú reprezentáciu paragrafov v odstavci o kvantitatívnom a kvalitatívnom zložení. Následne spustíme takúto transformáciu nad všetkými trénovacími dátami (32 dokumentov), ktoré sme použili na analýzu a ktoré boli stiahnuté náhodne zo stránok úradu. Interné reprezentácie porovnáme s pôvodným textom a na základe našich požiadaviek na extrakciu účinných látok zostavíme sadu pravidiel, podľa ktorých bude prebiehať extrakcia. Pravidlá predstavujú šablóny, podľa ktorých sa budeme rozhodovať, či daný úsek textu zodpovedá výskytu účinnej látky. Príklad takéhoto pravidla je: „MNOŽSTVO SSis2 SSfs2“ alebo „MNOŽSTVO SSis2 NEZNÁME“. Samotná extrakcia účinných látok následne spočíva v prevode daného odstavca na internú reprezentáciu a testovanie výskytu niektorého z pravidiel v tejto reprezentácii. Ak nájdeme výskyt niektorého

z pravidiel, prehlásime daný úsek textu za účinnú látku a hľadanie začíname od prvého slova za týmto výskytom.

Pravidlá sme extrahovali ručne tak, že sme porovnali pôvodný text s anotovaným textom a na základe výskytov pomocných látok a kontextu v akom sa nachádzali sme ich jednoducho definovali. Podarilo sa nám identifikovať 41 pravidiel. Pravidlá berú do úvahy 4-slovné, 3-slovné a 2-slovné výskyty účinných látok. Príklad pravidla a zodpovedajúceho výskytu účinnej látky:

„QUANTITY QUANTITY Ssis2 SSfs2“ => „10 miligramov monohydrátu laktózy“ 33

6.4.2.3 Integrácia Slovenského národného korpusu

Integrácia Slovenského národného korpusu nie je úplne jednoduchá záležitosť. V prvom rade je potrebné vyplniť registračný formulár. Následne dostanete podmienky používania, ktoré je potrebné podpísať a doručiť na adresu jazykovedného ústavu. Napriek tomu, že je korpus webová aplikácia a podľa nášho názoru je používaná hlavne pri strojovom spracovaní textu, sa nám nepodarilo nájsť žiadnu dokumentáciu pre strojové dotazovanie do korpusu.

Na základe týchto skutočností je potrebné webové rozhranie korpusu podrobne preskúmať a zistiť, akým spôsobom je možné integrovať korpus do našej aplikácie. Výsledok je taký, že v zdrojovom kóde je uložená pomerne dlhá URL, ktorá zodpovedá určitej stránke vo webovom rozhraní korpusu. Do tejto adresy pridáme slovo, ktoré chceme anotovať a odošleme HTTP request. Ako odpoveď sa nám vráti kompletný HTML dokument, ktorý by sa normálne zobrazil v prehliadači. My v tomto dokumente na základe predošlej analýzy nájdeme konkrétny element, v ktorom sa nachádza tag pre nami požadované slovo.

```
<html>
...
  <td class="word">SSfs4</td>
  <td align="right" class="frequency"> 1 </td>
  <td align="right">16.7</td>
...
</html>
```

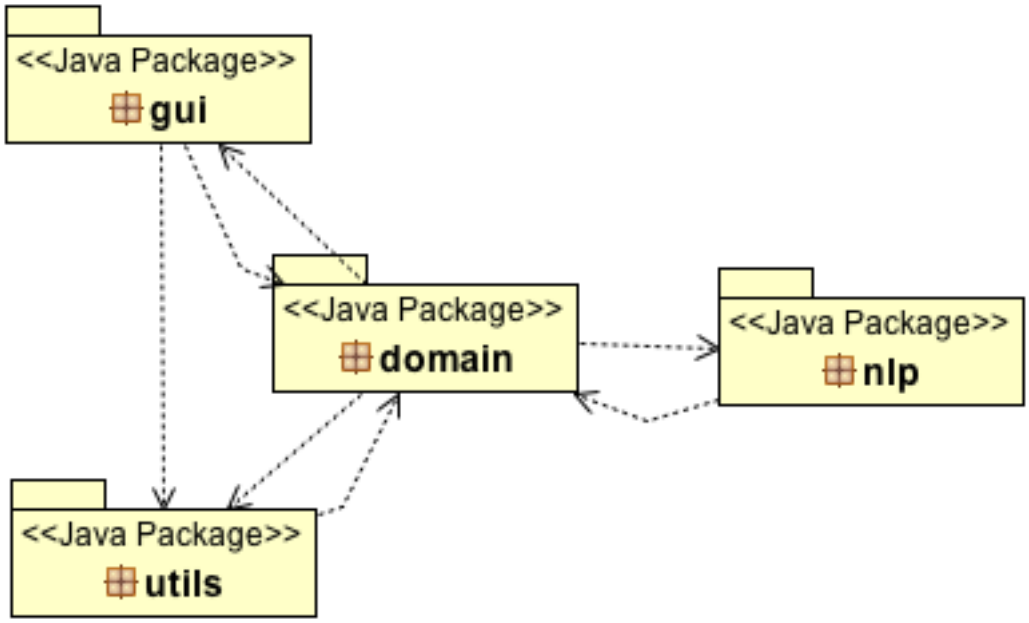
Obrázok 9: Ukážka relevantnej časti HTML dokumentu s tagom

7 Implementácia

V kapitole implementácia sa budeme zaoberať implementačnými detailmi, konkrétnymi postupmi, knžnicami a návrhovými rozhodnutiami. Vzhľadom na predošlé skúsenosti a zvolené knižnice sme sa rozhodli pre implementáciu v programovacom jazyku Java. Tento programovací jazyk má veľmi kvalitnú dokumentáciu, množstvo dostupných príkladov na internete, veľkú komunitu, pokročilé integračné vývojové prostredia a veľmi dobrú prenositeľnosť medzi rôznymi operačnými systémami.

7.1 Architektúra aplikácie

Kód našej aplikácie je rozdelený do 4 balíkov. Každý balík predstavuje logický celok súvisiacich tried a zodpovedností. Pri návrhu a tvorbe aplikácie, sme sa držali princípov tvorby softvéru podľa pána Craiga Larmana.



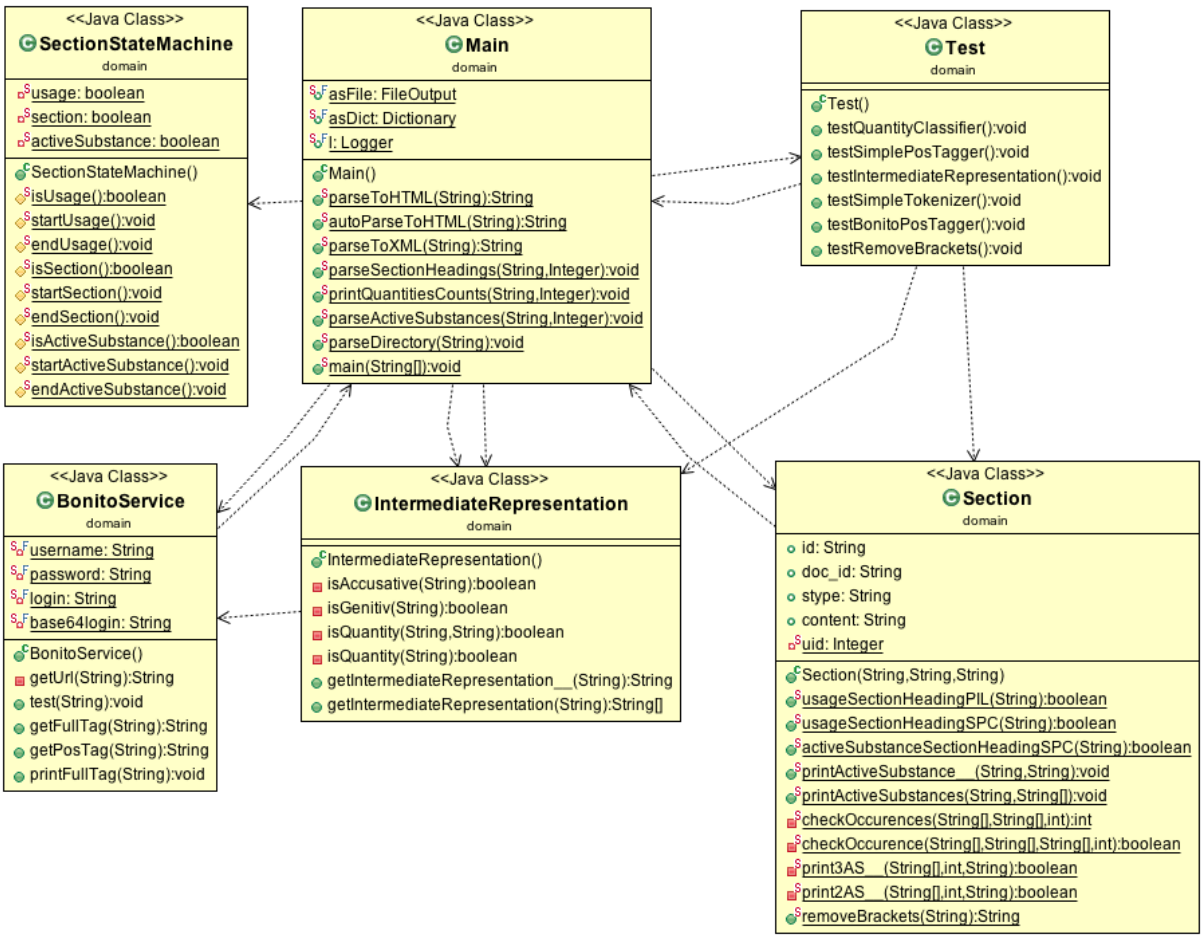
Obrázok 10: Diagram balíkov vytvorenej aplikácie

- balík „utils“- názov vychádza z anglického slova „utilities“, ktorého doslovný preklad je „pomocné programy“. V našom prípade tento názov skôr predstavuje „pomocné triedy“. V tomto balíku sa nachádzajú triedy, ktoré sú zodpovedné za logovanie, pripojenie do databázy a súvisiace pomocné triedy k nim.
- balík „nlp“ – skratka nlp predstavuje „Natural language processing“ a tento balík obsahuje triedy, ktoré súvisia s týmto pojmom. Nájdeme tu naivnú implementáciu vlastného tokenizéru, POS taggeru, klasifikátoru kvanitatívnych výskytov množstva a jednotiek, POS taggeru, ktorý

sa napája na Slovenský národný korpus a ku každej triede je rozhranie, cez ktoré sa napája ku zvyšku aplikácie. Tieto rozhrania boli vytvorené za účelom jednoduchej integrácie nových tried, ktoré súvisia s touto oblasťou. Príkladom môže byť nová implementácia tokenizéru, ktorá bude dodržiavať predpísané rozhranie. Takto vytvorenú implementáciu vieme jednoduchšie integrovať do aplikácie.

- balík „domain“ – obsahuje triedy, v ktorých je implementovaná hlavná logika aplikácie a ich názvy a zodpovednosti sú priamo závislé od našej domény. Nájdeme tu triedy ako Section (paragraf), SectionStateMachine (stavový automat na hľadanie paragrafov), IntermediateRepresentation (konverter textu do internej reprezentácie) a iné
- balík „gui“ – by mal zlučovať triedy, ktoré súvisia s grafickým používateľským rozhraním. V tomto stave tento balík obsahuje iba triedu, ktorá je schopná zobraziť prostredníctvom tabuľky JTable, všetky nájdené aktívne látky v liekoch. Táto funkcionalita je tam za účelom akéhosi „proof-of-concept“, pre budúcu možnosť jednoduchého odstraňovania nesprávne identifikovaných aktívnych látok.

Z diagramu balíkov by som chcel poukázať na nevýhodu rekurzívnych závislostí medzi balíkom domain a všetkými ostatnými. V praxi to totiž znamená, že ak by sme chceli napríklad balík *nlp* použiť v inom projekte, museli by sme ho vzhľadom na jeho závislosť od balíku *domain* najprv upraviť. V našom prípade by sme mohli zoradiť tieto balíky podľa možnosti ich opätovného použitia v inom projekte takto: *utils, nlp, gui, domain*. Doménový balík je priamo závislý na kontexte aplikácie. Určité časti balíka s používateľským rozhraním by mali byť použiteľné medzi projektami. Spracovanie prirodzeného jazyka je dostatočne všeobecná oblasť, ktorú by sme mali byť schopní bez zásadných zásahov integrovať do iného projektu. Pripojenie do databázy a logovanie má takmer každá aplikácia, preto by sme balík s utilitami zaradili na prvé miesto možného opätovného použitia. Vzhľadom na pomery tejto práce však môžeme túto potenciálnu nevýhodu zanedbať, nakoľko je veľmi malá pravdepodobnosť, že by sa vytvorená aplikácia dostala do komerčnej sféry bez ďalších úprav. Ako výhodu by som na druhú stranu vyzdvihol pomerne málo závislostí medzi balíkmi. Doménový balík je centrálny uzol, ktorý spolupracuje s pridruženými balíkmi, avšak jednotlivé balíky o sebe vôbec nemusia vedieť. Závislosť balíka s grafickým používateľským rozhraním od balíka s utilitami vznikla testovacími výpismi (teda používania logovania) a v produkčnej verzii aplikácie by sme mohli túto závislosť odstrániť.



Obrázok 11: Diagram tried doménového balíku

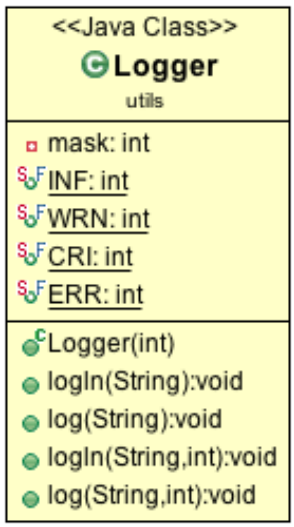
Na obrázku môžeme vidieť diagram tried doménového balíka. Tento balík obsahuje nasledujúce triedy:

- SectionStateMachine – trieda obsahuje metódy na overenie, či daný odstavec predstavuje začiatok nového paragrafu a či sa jedná o konkrétnu sekciu, ktorú hľadáme. Okrem toho udržuje interne aktuálny stav, ktorý nám hovorí v akej sekcii v rámci dokumentu sa nachádzame.
- Main – trieda, ktorá obsahuje vstupnú metódu do aplikácie, základné prechádzanie priečinku s dokumentami a volanie metód z iných tried za účelom extrakcie. Okrem toho na začiatku zavolá testovanie, ktoré je implementované v triede Test.
- Test – obsahuje základné, pomerne jednoduché integračné testy, ktoré testujú jednotlivú funkcionalitu tried spracovania prirodzeného jazyka, nastavenia prostredia a triedy, ktorá komunikuje so Slovenským národným korpusom. Tieto testy majú hlavne za účel zistiť, či po pridaní novej funkcionality, alebo refaktoringu starej, nedošlo k neželanej zmene správania jednotlivých tried a metód.
- BonitoService – adaptér na integráciu Slovenského národného korpusu. Používateľské meno a heslo na pripojenie ku korpusu si táto trieda zoberie z premenných prostredia. Správne

nastavenie premenných prostredia sa testuje v triede Test. BonitoService poskytuje metódy na získanie potrebného tagu pre daný token

- IntermediateRepresentation – trieda zabezpečuje konverziu jednotlivých odstavcov na internú reprezentáciu a pridružené pomocné metódy
- Section – poskytuje metódy na prácu so sekciami, testy či sa jedná o konkrétny nadpis, ktorý hľadáme. Ak sme našli konkrétny nadpis, uložíme nový stav do SectionStateMachine.

Okrem týchto tried, by som rád ešte spomenul triedu Logger.



Obrázok 12: Diagram triedy Logger

Trieda Logger, poskytuje niekoľko stupňov logovania:

- INF – informačné správy o chode aplikácie
- WRN – upozornenia aplikácie, ktoré je možné ignorovať v produkčnom prostredí
- CRI – kritické správy, ako napríklad nájdené aktívne pomocné látky, alebo názov aktuálne spracovávaného dokumentu
- ERR – chybové hlášky aplikácie

K jednotlivým stupňom sú priradené čísla, ktoré reprezentujú ich dôležitosť. Tieto čísla sú mocninami čísla 2. Konštruktor triedy Logger očakáva integer ako masku, v ktorej je zakódované, ktoré výpisy chceme vypisovať a ktoré chceme ignorovať. Príkladom môže byť volanie *new Logger(Logger.ERR | Logger.CRI)*, ktoré nám zabezpečí, že aplikácia bude vypisovať iba chybové hlášky a kritické správy.

7.2 Pseudokód hlavnej funkcionality

V krátkosti načrtneme pseudokód hlavnej funkcionality, ktorá sa stará o konverziu do internej reprezentácie a následné vyhľadávanie vzorov:

```
/*1*/for each document in documentDirectory
/*2*/  for each paragraph in document
/*3*/    if weAreInterestedInParagraph(paragraph)
/*4*/    ir = intermediateRepresentationOfParagraph(paragraph)
/*5*/    for each token in paragraph
/*6*/      for each rule in rules
/*7*/        checkOccurenceFromCurrentToken(ir, token, rule)
/*8*/        if occurence == true
/*9*/          printRespectiveTokens
/*-*/        endif
/*-*/      endfor  //each rule
/*-*/    endfor    //each token
/*-*/   endif      //if weAreinterestedInParagraph
/*-*/  endfor      //each paragraph
/*-*/endfor        //each document
```

1. iterácia cez všetky dokumenty, ktoré sme náhodne vybrali a ručne stiahli z internetovej stránky Štátneho úradu pre kontrolu liečiv
2. iterácia, cez všetky paragrafy, ktoré sme získali konverziou na HTML dokument (Apache Tika) a následným vyhľadaním všetkých paragrafov prostredníctom knižnice Jsoup
3. test, či sa daný paragraf nachádza v nami požadovanej sekcii (túto informáciu uchováva SectionStateMachine)
4. interná reprezentácia je pole reťazcov, ktoré predstavujú tagy po dotaze do Slovenského národného korpusu
5. rozdelenie paragrafu na tokeny je úlohou nášho interného naivného tokenizátoru, ktorý rozdelí reťazec v závislosti od výskytu medzier a špeciálnych znakov ako sú tabulátor, koniec riadku a pod.
6. pravidlá sme vytvorili ručne podľa výskytov aktívnych látok v dokumentoch a predošlou analýzou
7. v závislosti od dĺžky pravidla (2-4 tokeny) skontrolujeme od aktuálneho tokenu v internej reprezentácii tagy získané zo Slovenského národného korpusu s tagmi v pravidle
8. test, či sme v predošlom kroku našli zhodu
9. jednoduchý výpis tokenov z pôvodného paragrafu

7.3 Analýza časovej a pamäťovej zložitosti programu

7.3.1 Analýza časovej zložitosti

Na základe pseudokódu, ktorý sme opísali v predchádzajúcej kapitole, môžeme tvrdiť nasledovné:

- prvé 3 riadky vnorených for cyklov, sa dajú generalizovať do jedného, keďže je to iba iterácia cez všetky „slová“ vo všetkých dostupných dokumentoch, to nám dáva časovú zložitosť *O(n)*, kde *n*, je počet slov vo všetkých analyzovaných dokumentoch
- získanie vnútornej reprezentácie je závislé na počte „slov“, takže opäť *O(n)*
- počet pravidiel je v danom momente stále konštantný, takže iterácia cez všetky pravidlá je bez ohľadu na počet dokumentov (slov), stále konštantná, *O(1)*
- test na výskyt jedného z pravidiel je stále konštantný, keďže počet ani dĺžka pravidiel sa nemení vzhľadom od počtu dokumentov (slov), *O(1)*

Vzhľadom na predošlé tvrdenia môžeme konštatovať časovú zložitosť *O(n)*, teda lineárne závislú od počtu slov v analyzovaných dokumentoch. Je však veľmi dôležité poznamenať, že Slovenský národný korpus je dostupný cez internetovú sieť a teda celkový čas behu aplikácie záleží od rýchlosti internetového pripojenia a od vyťaženosti serverov korpusu. Počas testovania a vyhodnocovania aplikácie sme zistili, že práve dotazy do korpusu prinášajú do behu najväčšiu časovú záťaž a aplikácia viac čaká na odpoveď korpusu ako vykonáva samotné výpočty.

7.3.2 Analýza pamäťovej zložitosti

Pre každý spracúvaný dokument si uchovávame jeho HTML reprezentáciu a k nej duálne internú reprezentáciu textu. Po dokončení práce s daným dokumentom sa tieto dáta odstránia a miesto sa uvoľní pre dáta nového dokumentu. Okrem toho držíme v pamäti po celý čas behu všetky pravidlá a samotný kód aplikácie. Vzhľadom na to, že o už spracovaných dokumentoch neevidujeme v pamäti žiadne informácie si dovolíme tvrdiť, že pamäťová zložitosť nezávisí od počtu dokumentov ani slov, ktoré obsahujú. Je síce pravda, že v závislosti od dĺžky konkrétneho dokumentu, bude raz v pamäti viac údajov a inokedy menej, no celková pamäťová zložitosť aj tak ostáva konštantná a teda *O(1)*.

8 Experimentálne vyhodnotenie

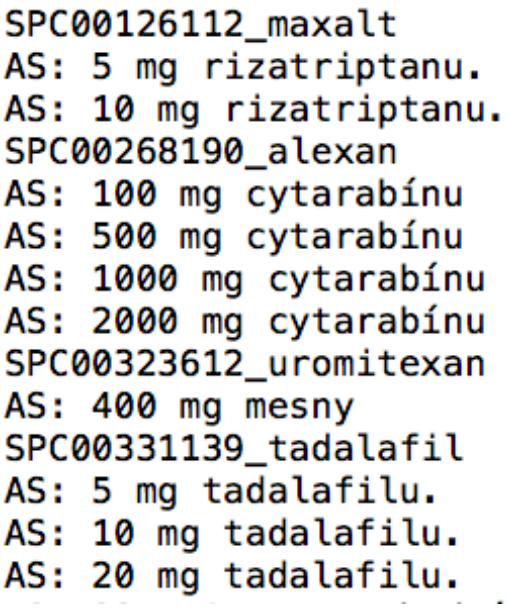
Táto kapitola prináša overenie funkčnosti navrhnutej metódy spolu s vyhodnotením jej výsledkov. Vzhľadom na doménu medicínskeho charakteru by mal takýto nástroj mať takmer nulovú chybovosť pri extrahovaní údajov o liekoch. Akékoľvek pochybenie, resp. nesprávne extrahovanie dát sa môže negatívne odraziť na ľudskom zdraví. V prípade vyššej chybovosti treba prípadným používateľom vyhľadávača jednoznačne oznámiť a upozorniť ich, že dáta majú informačný charakter a môžu sa v nich vyskytovať chyby. Alternatívou môže byť experimentálne vyhodnotenie presnosti každej extrakcie na základe spomenutého slovníka. Takéto rozšíreneie v súčasnosti nie je súčasťou tejto práce. Vzhľadom na kontext vyhľadávača nás nezaujíma až tak úplnosť extrakcie, pretože ak nejaké dáta nie sú k dispozícií, nikoho to priamo neohrozuje.

Vyhodnotenie má skutočne len experimentálny charakter. Dáta, na ktorých by sme takéto riešenie mohli rigorózne vyhodnotiť nie sú k dispozícií a preto sme si referenčné extrakcie museli extrahovať ručne sami. Vzhľadom na túto skutočnosť sa mohlo stať, že samotné referenčné dáta obsahujú chyby, alebo nie sú úplné a presné, a preto zistené výsledky majú skôr informačný charakter.

8.1 Extrakcia referenčných dát

Za účelom vyhodnotenia našej metódy sme zo stránky Štátneho úradu pre kontrolu liečiv stiahli náhodným výberom 30 dokumentov rôznych liekov. Dokumenty sme nevyberali podľa toho, ktoré lieky sú nám známe. Do vyhľadávača, ktorý stránka poskytuje sme zadali náhodné písmená (vo väčšine 3) a ak nám stránka zobrazila výsledky, tak sme náhodne niektoré lieky vybrali a dokumenty stiahli vo formátoch MS Word (.doc, .docx). Pre úplnosť by sme mali poznamenať, že pri tomto výbere sme narazili na jeden veľmi exotický dokument vo formáte PDF, ktorý sa svojím obsahom ani konštrukciou nepodobal na žiadne iné dokumenty, na ktoré sme narazili. Keďže takýto prípad nastal iba jeden, dokument sme nestiahli a tento výber sme ignorovali.

Po stiahnutí všetkých dokumentov, sme vytvorili referenčný súbor tak, aby odpovedal výstupu z nášho programu:



Obrázok 13: Snímok obrazovky súboru ručne extrahovaných aktívnych látok

V súbore nájdeme názov dokumentu a pod ním zoznam všetkých aktívnych látok, ktoré sme z dokumentu extrahovali ručne. Niektoré látky sa v dokumente vyskytujú opakovane s rôznymi množstvami. Toto plynie z povahy dát, pretože v jednom dokumente sa môžu vyskytovať dávky a obsahy látok pre rôzne balenia. Príkladom je napríklad liek Ibuprofen 200/400/800.

Referenčný dokument obsahuje 167 riadkov, z ktorých je 137 riadkov s aktívnymi látkami a teda 137 aktívnych látok. Zvyšných 30 riadkov je ponechaných pre názvy dokumentov, čo súhlasí s množstvom referenčných dokumentov.

8.2 Proces vyhodnocovania

Výstup nášho programu má rovnaký formát ako referenčný súbor. Vypíšeme názov dokumentu a nasledujú všetky extrahované aktívne látky, každá na samostatnom riadku. Program bol za účelom vyhodnocovania spustený tak, aby dokumenty ktoré analyzuje zodpovedali referenčným dokumentom. Prakticky disponujeme 2 priečinkami. V jednom priečinku sú dokumenty, ktoré sme používali pri tvorbe pravidiel a analýze dokumentov a v druhom priečinku sú dokumenty, ktoré slúžia ako referenčné. Toto rozdelenie je analogické k trénovacím a testovacím dátam pri trénovaní a testovaní klasifikátorov. Výstup programu následne uložíme do súboru out.txt. Referenčný súbor má názov ref.txt.

Samotné vyhodnocovanie sme realizovali poloautomaticky za použitia štandardného UNIX programu diff, príkazom:

```
diff -iBawy ref.txt out.txt                    34
```

Použité prepínače programu *diff*:

- „*i*“ – ignoruje veľkosti písmen (*ignore-case*)
- „*B*“ – ignoruje riadky, ktoré sú prázdne (ignore-blank-lines)
- „*a*“ – všetky súbory spracúva ako textové (text)
- „*w*“ – ignoruje všetky biele znaky (medzery, tabulátory, ...) (ignore-all-space)
- „*y*“ – výstup zobrazuje v 2 stĺpcoch (side-by-side)

Všetky riadky, ktoré sú v súboroch rozdielne nám program *diff* označí symbolmy:

- “<“ – riadok je naviac v súbore out.txt (pomocné látky extrahované „naviac“)
- „>“ – riadok z referenčného súboru chýba v súbore out.txt (neextrahované pomocné látky)
- „|“ – riadky nie sú totožné (čiastočné extrakcie)

8.3 Vyhodnotenie výsledkov

Výsledky z výstupov programu *diff* boli polo-automaticky kvantifikované pomocou programov *grep* a *wc*. Sumárne štatistiky môžeme vidieť v nasledujúcej tabuľke:

	Počet
Korektne extrahované	109
Čiastočne extrahované	21
Nesprávne extrahované	21
Chýbajúce extrakcie	7
Spolu	158

Tabuľka 1: Sumárne štatistiky výsledkov vyhodnotenia

Výsledky, ktoré nám zobrazuje táto tabuľka si podrobne rozoberieme:

- „Korektne extrahované“ – vyjadrujú počet riadkov, ktoré pri sebe nemali ani jednu zo značiek, ktoré by naznačovali nejaký problém na danom porovnávanom riadku. Do výsledného súčtu riadkov 109, nie sú započítané riadky, ktoré označovali názvy dokumentov. Tieto riadky sme pri analýze vôbec neuvažovali
- „Čiastočne extrahované“ – vyjadrujú počet riadkov, ktoré mali pri sebe znak „|“ a teda riadok sa podobal na riadok v referenčnom súbore, ale líšil sa o niektoré znaky. Takéto odchýlky vznikajú prílišnou všeobecnosťou extrakčných pravidiel, alebo chýbajúcimi extrakčnými

pravidlami. Túto skupinu si podrobnejšie rozdelíme. 4 riadky v tejto skupine sa líšili iba o spojku „a“, vzhľadom na túto skutočnosť môžeme dané 4 riadky považovať za takmer úplne korektne extrahované. 11 riadkov sa líšilo v jednom slove, príkladom môže byť referenčná „*kyselina klavulanová“* a extrahovaná „*kyselina*“ bez prívlastku. Tieto nepresnosti vznikli absenciou pravidiel, ktoré by zahŕňali aj prívlastok danej aktívnej látky. Zvyšných 6 riadkov správne obsahovalo množstvo danej látky, avšak látku sa extrahovať nepodarilo. Príkladom je text „*propylénglykol 600 mg dávka obsahuje“,* kde sme namiesto “*propylénglykol 600 mg“* extrahovali „*600 mg obsahuje*“. Tieto nepresnosti predstavujú skutočný problém, nakoľko v takomto prípade by sme našu metódu museli rozšíriť o nejakú heuristiku, resp. o spomenutý pomocný slovník.

- „Nesprávne extrahované“ – charakterizované symbolom „>”. Takéto extrakcie poukazujú na všeobecnosť extrakčných pravidiel. Ak sa v texte vyskytuje text, ktorý obsahuje množstvo a slová v tvare nášho pravidla, program takýto text extrahuje ako aktívnu látku. Príklady sú: „500 mg granulátu“, „25mg tableta“, „1 ml infúzneho koncentrátu“. Takto extrahované „aktívne látky“ avšak v kontexte vyhľadávača nepredstavujú žiadny problém. Na jednej strane je človek schopný rýchlo identifikovať že sa nejedná o aktívnu látku a okrem toho v prípade použitia, kedy človek zadá aktívnu látku a chce zoznam liekov, ktoré ju obsahujú, tak nejaké údaje naviac nie sú problém, ak ich nezobrazujeme. Okrem toho je možné takéto chyby pomerne presne odstrániť buď manuálne, alebo pomocou slovníku.
- „Chýbajúce extrakcie“ – aktívne látky, ktoré sa nachádzajú v referenčnom súbore ale nie vo výstupe programu. Vzhľadom na počet týchto prípadov sme analyzovali všetky podrobnejšie a zistili sme, že dané látky neboli extrahované, pretože sa medzi našimi extrakčnými pravidlami nenachádzali také, ktoré by ich extrahovali. Tento problém je možné odstrániť rozšírením množiny dát na tvorbu pravidiel, avšak naskytá sa otázka, koľko dokumentov by bolo dostačujúcich pre extrakciu všetkých látok.

Riadok tabuľky označený ako „Spolu“ obsahuje číslo 158. Ak k tomuto číslu pripočítame 30 (počet dokumentov), dostaneme presný počet riadkov súboru, ktorý sme dostali po spustení programu *diff*. Tento súbor obsahuje 188 riadkov: 30 riadkov názvov dokumentov, 109 korektných extrakcií, 21 čiastočných extrakcií, 21 nesprávnych extrakcií a 7 neextrahovaných aktívnych látok.

Ak by sme tieto výsledky chceli vyčísliť percentuálne ako presnosť extrakcie našej metódy, mohli by sme postupovať takto:

$$158\text{-}21 = 137 \quad (35)$$

$$109+4 = 113 \quad (36)$$

$$113 / 137 = 0{,}8248 \quad (37)$$

Prvý výpočet nám udáva celkový počet aktívnych látok, ktoré by sme podľa referenčného súboru mali extrahovať (158 celkovo – 21 nesprávnych extrakcií). Druhým výpočtom dostávame celkový počet extrahovaných látok, ktoré považujeme za správne extrahované (91 korektných extrakcií + 4, ktoré sa líšili iba o spojku „a“). Posledný výpočet nám dáva do pomeru počet korektných extrakcií k celkovému počtu aktívnych látok. Ak tento pomer vyjadríme percentuálne, dostaneme 82,48% presnosť extrakcie. Dovolíme si tvrdiť, že vzhľadom na naivitu použitej metódy a malú vzorku trénovacích dát je to dobrý výsledok. Pre úplnosť je však potrebné doplniť, že 30 testovacích dokumentov nie je reprezentatívna vzorka celkového počtu dokumentov a preto tieto výsledky môžu byť signifikantne skreslené povahou testovacích dokumentov.

V predošlej kapitole sme mali ukážku referenčného súboru. V súbore sme mohli vidieť, že niektoré aktívne látky sa tam vyskytovali viackrát avšak s rozdielnym množstvom. Na jednej strane by sme mohli takéto výskyty považovať za odlišné, no na druhej strane vo väčšine prípadov sa takéto duplikáty vyskytovali v rovnakých vetách. Takáto skutočnosť nám do výsledkov vniesla nepresnosti a to takým spôsobom, že niektoré aktívne látky (duplikáty) prispievali či už pozitívne alebo negatívne k celkovému výsledku rôznou mierou. Ak by sme mali dokument, ktorý by obsahoval 50 aktívnych látok s rozdielnym množstvom a všetky by sme extrahovali správne, v skutočnosti len jedna extrakcia by nám signifikantne skreslila výsledok pozitívnym smerom. Vzhľadom na túto skutočnosť sme sa rozhodli ručne odstrániť duplikáty z referenčného súboru aj z výstupu nášho programu. Za duplikáty sme považovali riadky, ktoré až na množstvo boli rovnaké. Následne sme rovnakým spôsobom analyzovali výsledky a výsledky si ukážeme v tabuľke:

	Počet
Korektne extrahované	89
Čiastočne extrahované	9
Nesprávne extrahované	9
Chýbajúce extrakcie	7
Spolu	114

Keďže tieto výsledky sú svojou povahou totožné s predošlými, nebudeme ich podrobne analyzovať. Pre korektnosť však podotkneme, že z čiastočne extrahovaných je 1 látka, ktorá sa líšila iba spojkou „a“,

čo presne odpovedá skutočnosti, že v predošlých výsledkoch sa jednalo o tie isté 4 extrakcie líšiace sa iba v množstve a vo výsledku sa líšili iba o spojku „a“.

Pri percentuálnom vyjadrení presnosti budeme postupovať rovnako ako minule:

$$114 - 9 = 105 \quad (38)$$

$$89 + 1 = 90 \quad (39)$$

$$90 / 105 = 0{,}8571 \quad (40)$$

Rozdiel nie je príliš veľký, ale ak zoberieme do úvahy argumenty, ktoré sme uviedli prečo sme sa rozhodli nebrať duplikáty, tak tento výsledok považujeme za presnejší a tým pádom môžeme tvrdiť, že naša metóda je schopná správne extrahovať ~86% aktívnych látok.

9 Záver

Táto diplomová práca sa zameriava na problém extrakcie konkrétnych údajov z čiastočne štruktúrovaného textu. Začiakom práce sú podrobne popísané rôzne metódy spracovania prirodzeného jazyka, história extrakcie informácií a rôzne metódy extrakcie. Následne boli analyzované dokumenty, z ktorých extrahujeme dáta. Dokumenty sa dajú voľne stiahnuť zo stránky Slovenského úradu pre kontrolu liečiv. Tento úrad zodpovedá za vydávanie povolení na predaj liekov v rámci Slovenskej republiky.

Extrakcia z medicínskych dokumentov je zasadená do kontextu budúceho projektu vyhľadávača informácií o dostupných liekoch na Slovensku. Vzhľadom na tento kontext boli robené viaceré návrhové rozhodnutia a brali sme to do úvahy aj pri vyhodnocovaní nami navrhnutej metódy.

Z dostupných dokumentov sme sa rozhodli extrahovať krátke odstavce, v ktorých je napísané za akých okolností sa má daný liek užívať. Tieto odstavce sú natoľko krátke, stručné a výstižné, že detailnejšiu extrakciu nie je potrebné robiť a samotná extrakcia spočíva iba v nájdení daného nadpisu odstavca a extrakcie celého odstavca. Okrem týchto krátkych odstavcov sme sa rozhodli extrahovať aktívne látky, ktoré sa v lieku nachádzajú. Tieto látky sa tiež nachádzajú v špecifických odstavcoch, ktoré vieme presne v dokumente vyhľadať. Samotné aktívne látky sú extrahované pomocou Slovenského národného korpusu, ktorý pri extrakcii slúži ako pokročilý značkovač prirodzeného jazyka. Na základe značiek, ktoré získame z korpusu sme ručne vytvorili sadu pravidiel, pomocou ktorých extrahujeme aktívne látky.

Metóda bola vyhodnotená na sade 30 dokumentov, ktoré slúžili ako referenčné. Z týchto dokumentov bol ručne vytvorený referenčný súbor, ktorý mal štruktúru rovnakú ako výstup nášho programu. Program bol spustený na sade referenčných dokumentov a jeho výstup bol následne polo-automaticky porovnaný s referenčným súborom. Výsledkom je približne 86% presnosť extrakcie.

Túto metódu je v budúcnosti možné rozšíriť o slovník, ktorý môže byť priebežne overovaný doménovým expertom. Tento slovník môže slúžiť ako referenčný register aktívnych látok, podľa ktorého môžeme túto metódu spresniť a pomôže nám odstrániť nesprávne extrahované slová, ktoré nepredstavovali aktívne látky. Okrem toho sa v dokumentoch nachádza množstvo ďalších zaujímavých informácií, ktoré by určite stálo za to skúsiť extrahovať. Medzi ne patria napríklad kontraindikácie alebo dávkovanie.

10 Použitá literatúra

[1] R. Garabík, L. Gianitsová, A. Horák a M. Šimková, Tokenizácia, lematizácia a morfologická anotácia Slovenského národného korpusu, Bratislava: www.korpus.sk, 2004.

[2] R. Garside, The CLAWS word-tagging system, Longman, 1987.

[3] E. Brill, A simple rule-based part of speech tagger, Association for Computational Linguistics, 1992.

[4] H. Schmid, Probabilistic Part-of-Speech tagging using decision trees, zv. 12, Citeseer, 1994, pp. 44--49.

[5] J. R. Hobbs a E. Riloff, „Information extraction," *Handbook of natural language processing,* zv. 2, 2010.

[6] S. Busemann a H.-U. Krieger, Resources and Techniques for Multilingual Information Extraction, 2004.

[7] H. Cunningham, „Information Extraction, Automatic," *Encyclopedia of language and linguistics,* pp. 665--677, 2005.

[8] J. Makhoul, F. Kubala, R. Schwartz a R. Weischedel, Performance Measures for Information Extraction, 1999, pp. 249--252.

[9] A. Téllez-Valero, M. Montes-y-Gómez a L. Villaseñor-Pineda, A Machine Learning Approach to Information Extraction, Springer, 2005, pp. 539--547.

[10] Wikipedia, „Support vector machine," 16 May 2016. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine. [Cit. 10 April 2016].

[11] T. Joachims, Learning to Classify Text Using Support Vector Machines, Kluwer Academic Publishers, 2002.

[12] P. Knoth, Extrakce informací z biomedicínských textů, Brno: Vysoké učení technické, 2008.

[13] E. Kim, „Everything you wanted to know about the Kernel Trick," 1 September 2013. [Online]. Available: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html. [Cit. 20 March 2016].

[14] E. Fix a J. L. Hodges Jr, Discriminatory analysis-nonparametric discrimination: consistency properties, DTIC Document, 1951.

[15] Wikipedia, „K-nearest neighbours algorithm," 19 April 2016. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. [Cit. 1 May 2016].

[16] D. Coomans a D.L.Massart, „Alternative k-Nearest neighbour rules in supervised pattern recognition,“ *Analytica Chimica Acta,* zv. 136, pp. 15--27, 1982.

[17] J. L. Wyatt, „Kohonen Networks,“ [Online]. Available: https://www.cs.bham.ac.uk/~jlw/sem2a2/Web/Kohonen.htm.

[18] R. Yangarber, Scenario Customization for Information Extraction, DTIC Document, 2001.

[19] S. Brin, Extracting patterns and relations from the world wide web, Springer, 1998, pp. 172--183.

[20] E. Agichtein a L. Gravano, Snowball: Extracting Relations from Large Plain-Text Collections, ACM, 2000.

[21] W3C, „Resource Description Framework,“ 1999. [Online]. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[22] T. Berners-Lee, J. Hendler a O. L. a. others, „The semantic web,“ *Scientific american,* zv. 284, pp. 28--37, 2001.

[23] Williams, „Introducing The Concept of Web 3.0,“ 2012. [Online]. Available: http://www.tweakandtrick.com/2012/05/web-30.html.

[24] W3C, „SPARQL Query Language for RDF,“ 2008. [Online]. Available: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[25] MySQL, „MySQL Documentation - Internals,“ [Online]. Available: http://dev.mysql.com/doc/internals/en/.

[26] PostgreSQL, „About page,“ [Online]. Available: postgresql.com/about.

[27] „Why I choose PostgreSQL over MySQL-MariaDB,“ [Online]. Available: http://insights.dice.com/2015/03/19/why-i-choose-postgresql-over-mysqlmariadb/.

[28] W. W. Cohen, Integration of heterogeneous databases without common domains using queries based on textual similarity, zv. 27, ACM, 1998, pp. 201--212.

Zoznam príloh

Príloha č. 1 - Dokument, stiahnutý zo stránky úradu pre liek Calcium Chloratum

Príloha č. 2 – Obsah priloženého média

Príloha č. 1

Dokument, stiahnutý zo stránky úradu pre liek Calcium Chloratum

Príloha č. 1 k notifikácii zmeny v registrácii lieku, ev.č.: 2011/04644-ZME

SÚHRN CHARAKTERISTICKÝCH VLASTNOSTÍ LIEKU

1. Názov lieku

CALCIUM CHLORATUM Biotika

Injekčný roztok

2. Kvalitatívne a kvantitatívne zloženie lieku

Liečivo: Calcii chloridum dihydricum 0,671 g v 10 ml.

10 ml obsahuje 183 mg vápnika, to zodpovedá 4,56 mmol.

10 ml obsahuje 324 mg chloridov, to zodpovedá 9,12 mmol.

Pomocné látky: úplný zoznam pomocných látok, pozri časť 6.1.

3. Lieková forma

injekčný roztok

Popis lieku: číry, bezfarebný roztok, bez mechanických cudzorodých častíc.

4. KLINICKÉ ÚDAJE

4.1. Terapeutické indikácie

Nízka hladina vápnika v krvi (podporná liečba pri rachitíde, osteomalácii, osteoporóze, pri hojení fraktúr, tetanii, spazmofílii), pokles tlaku vyvolaný akútnym oslabením kontrakčnej sily myokardu (napr. v anestéziológii pôsobením intravenóznych barbiturátov).

Akútne alergické choroby, chronické zápalové ochorenia, svrbiace dermatózy, mokvajúce a generalizované ekzémy.

4.2. Dávkovanie a spôsob podania

a) Dávkovanie deťom:

Deťom sa podáva intravenózne. Dávka sa upravuje podľa veku, druhu výživy a hodnôt kalcémie.

Dávka denná pre deti do jedného roku: do 0,25 g.

Dávka denná pre deti od 1 do 6 rokov: od 0,25 do 0,5 g.

Dávka denná pre deti od 6 do 15 rokov: od 0,5 do 1g.

Pri parenterálnej aplikácii vápnika je potrebné monitorovať srdcovú činnosť.

b) Dávkovanie dospelým:

Dospelým sa podáva 5 až 10 ml prísne intravenózne, veľmi pomaly, počas 3 až 10 minút. Výnimkou je kardiálna resuscitácia, kedy sa intravenózny bolus podáva veľmi rýchlo.

Pri akútnej symptomatickej hypokalcémii sa dospelým pacientom iniciálne podáva vápnik vo forme bolusu prísne intravenózne v dávke 100 - 200 mg elementárneho vápnika v priebehu 10 minút, potom nasleduje udržiavacia infúzia elementárneho vápnika 1 - 2 mg/kg/h. Hladina vápnika v sére sa obyčajne v priebehu 6 až 12 h dostane pri tomto režime do normálnych hodnôt, takže udržiavaciu dávku možno znížiť na 0,3 - 0,5 mg/kg/h. Hypomagnéziová alebo hypermagnéziová hypokalcémia nereaguje dobre na podávanie vápnika. Pri intravenóznej podpornej terapii vápnikom sa odporúča dávka 4 - 7 mg/kg/deň.

4.3. Kontraindikácie

Hyperkalcémia, hyperkalciúria (hyperparatyreoidizmus, predávkovanie vitamínu D, tumory spôsobujúce dekalcifikáciu ako je plazmocytóm, kostné metastázy), ťažká insuficiencia obličiek, náhly vzostup sérových hladín vápnika, pri osteoporóze zapríčinenej imobilizáciou pacienta, galaktozémia, anafylaktická reakcia s príznakmi hroziaceho šoku, terpia kardioglyzidmi (s výnimkou ťažkej symptomatickej hypokalcémie).

4.4. Osobitné upozornenia a opatrenia pri používaní

Intravenózne vápnikové preparáty dráždia vény, preto je potrebné riediť ich v 50 - 100 ml 5 % roztoku glukózy. Vápnik sa musí podávať opatrne u pacientov, ktorí sú na digitalisovej terapii, pretože hyperkalcémia pôsobí predispozične na digitalisovú toxicitu. Optimálna terapia vyžaduje časté monitorovanie hladín vápnika, horčíka, fosforu, draslíka a kreatinínu v sére, ako aj stanovenie elektrokardiografického a hemodynamického statusu.

Chlorid vápenatý je v svojej podstate acidifikujúci, preto nie je vhodný na liečbu hypokalcémie u pacientov s renálnou insuficienciou a acidózou. Liek sa nesmie podať intramuskulárne, resp. paravenózne pre nebezpečenstvo vzniku nekróz.

Vápnik môže spôsobiť nekrózu pečene u novorodenca, ak sa podáva umbilikálnou žilou.

4.5. Liekové a iné interakcie

Lokálne anestetiká, vrátane prokaínu, inhibujú alebo potláčajú transport ionizovaného vápnika z vodného prostredia do lipidovej fázy. Tento efekt sa neobmedzuje len na lokálne anestetiká, ale aj na

iné liečivá, ako sú propranolol, digitalisové glykozidy, CNS aktívne liečivá, vrátane morfínu a opiátových analgetických liečiv. Pri nečakanej hyperkalcémii sa zvyšuje riziko toxicity u pacientov, ktorí sú liečení digoxínom. Tiazidové diuretiká zvyšujú renálnu reabsorpciu kalcia, znižujú jeho vylučovanie močom a môžu vyvolať hyperkalcémiu. Súčasné podanie furosemidu a kalciových prípravkov môže vyvolať u novorodencov hyperkalcúriu a nefrokalcinózu.

4.6. Gravidita a laktácia

Vápnik prechádza placentárnou bariérou a vylučuje sa do mlieka. Jeho absorpcia je u kojencov limitovaná obsahom fosforu v mlieku.
Údaje o teratogenite a embryotoxicite neboli publikované.

4.7. Ovplyvnenie schopnosti viesť vozidlá a obsluhovať stroje

Nedochádza k ovplyvneniu pozornosti.

4.8. Nežiaduce účinky

Injekcie majú lokálne extrémne dráždivý účinok, pri extravazálnej aplikácii sú bolestivé a môžu spôsobiť nekrózu. Príliš rýchla intravenózna aplikácia vyvolá vazodilatáciu, pocit šíriaceho sa tepla a pálčivej kriedovej chuti v ústach. Soli vápnika vo vysokých dávkach podávané samotné, alebo s vitamínom D môžu vyvolať hyperkalcémiu s prejavmi anorexie, nauzey, dávenia, bolesti brucha, svalovej slabosti, bolesti kostí, polydipsie, polyúrie, zmätenosti, predráždenosti, kardiálnej dysrytmie, oslabenia až zastavenia činnosti srdca a kómy. U detí môže vzniknúť acidóza. Hyperkalcémia je reverzibilná, ale pretrvávajúce vysoké hladiny kalcia môžu spôsobiť ireverzibilnú nefrokalcinózu, nefrolitiázu a poruchu koncentračnej schopnosti obličiek. Menej závažným nežiaducim účinkom suplementácie chloridu vápenatého je obstipácia. V tomto prípade sa doporučuje potrava bohatá na vlákninu a primeraný príjem tekutín.
Rýchle podanie vápnikových preparátov zapríčiní náhle zvýšenie koncentrácie vápnika v sére, čo má za následok bradykardiu a srdcové arytmie. Extravazálny výstup vápnikového roztoku do subkutánneho tkaniva môže spôsobiť ťažké nekrózy tkaniva. Toto môže nastať pri použití infúznej pumpy.

4.9. Predávkovanie

Príznaky: Klinické príznaky sa manifestujú v závislosti od koncentrácie vápnika v sére. Pri ľahších stavoch je to nauzea, dávenie, únavnosť až somnolencia. Hyperkalcémia pri vyšších koncentráciách (nad 3,5 mmol/l) sa prejavuje návalmi tepla, poruchami mikcie, poruchami chute, periférnou vazodilatáciou, bolesťami brucha, psychickými poruchami, polydipsiou, polyúriou, svalovou slabosťou.

Liečba: Vo väčšine prípadov postačuje na zlepšenie stavu prerušenie suplementácie vápnika a zabezpečenie primeranej hydratácie. Hyperkalcémia maligných prípadov vyžaduje aktívnejšiu terapiu. Podávajú sa slučkové diuretiká a kálium šetriace diuretiká. Tiazidové diuretiká sú kontraindikované. Nutná je kontrola sérových elektrolytov. Pri veľmi ťažkom stave je možné použiť hemodialýzu.
Údaje o akútnej a chronickej toxicite preparátu nie sú publikované.

5. FARMAKOLOGICKÉ VLASTNOSTI

5.1. Farmakodynamické vlastnosti

Farmakoterapeutická skupina

Minerálne látky.

ATC kód: A12AA07

Mechanizmus účinku

Vápnik je esenciálny ión, ktorý je nevyhnutne potrebný pre normálnu funkciu mnohých biologických procesov organizmu, ako je vedenie nervových vzruchov, synaptická transmisia, sekrécia hormónov, srdcová automaticita, mitotická aktivita, spojenie kontrakcie-relaxacie vo svaloch, zrážanie krvi. Vápnik je tiež hlavný intracelulárny mediátor, je potrebný pre plnú aktivitu enzýmov. Z výpočtu týchto a mnohých ďalších funkcií vyplýva dôležitosť udržania fyziologických koncentrácií vápnikových iónov, pretože hypokalcémia sa môže manifestovať patologickými stavmi s ťažkým priebehom, ktoré ohrozujú život pacienta.
Úloha vápnika v regulácii excitability tkanív spočíva pravdepodobne v regulácii permeability bunkovej membrány pre ióny Na^+ a K^+. Svalový akčný potenciál stimuluje uvolňovanie vápenatých iónov zo sarkoplazmatického retikula a aktivuje kontrakciu. Mierne zníženie koncentrácie vápnika môže značne znížiť prah dráždivosti, čo má za následok vznik tetanických kŕčov. U niektorých pacientov s hypokalcémiou sa znížená koncentrácia iónov vápnika prejaví parestéziou, laryngeálnym spazmom, tetaniou. Intravenóznou suplementáciou sa klinický obraz zmierni a po čase upraví.
Vápnik je nepostrádateľným iónom pre spojenie excitácie-kontrakcie v srdcovom svalovom tkanive, ako aj pre vedenie elektrických impulzov v určitých oblastiach srdca, najmä v oblasti AV uzla. Depolarizácia myokardiálnych vláken otvára napäťovozávislé vápnikové kanály a spôsobuje pomalé vnútorné prúdy, ktoré sa tvoria cez plató akčného potenciálu. Tieto prúdy umožňujú permeáciu dostatočného množstva vápnikových iónov na uvoľnenie ďalších iónov zo sarkoplazmatického retikula a tým vyvolať kontrakciu. V rámci kardiovaskulárneho systému sa hypokalcémia manifestuje príznakmi, ako je hypotenzia, srdcová insuficiencia, dysrytmie (bradykardia, ventrikulárna fibrilácia). Ďalej sa môže prejaviť zníženou reakciou na liečivá, v mechanizme účinku ktorých sa zúčastňuje vápnik (noradrenalín, digoxín, dopamín). Pomalou intravenóznou aplikáciou ionizovaného kalcia sa obnoví u pacientov s hypokalcémiou vaskulárny tonus a upraví sa srdcová kontraktilita.

Vápnik hrá dôležitú úlohu pre udržanie integrity membrán slizníc, adhéziu buniek ako aj funkcie samotných bunkových membrán. Je potrebný pre exocytózu, a preto má dôležitú úlohu pre stimuláciu sekrécie u väčšiny exokrinných a endokrinných žliaz. Od iónov vápnika závisí uvoľňovanie katecholamínov z drene nadobličiek, neuromediátorov na synapsách, histamínu zo žírnych buniek atď. Aplikácia solí vápnika je indikovaná v šokových stavoch, kedy dochádza v organizme ku zvýšeniu priepustnosti bunkových membrán, a tým k úniku vápnika z vaskulárneho systému. Znížená schopnosť mobilizácie skeletálneho vápnika v dôsledku hypofunkcie PTH alebo deficiencie vitamínu D prispieva k zhoršeniu šokového stavu.

5.2. Farmakokinetické vlastnosti

Najväčšia časť absorbovaného množstva vápnika, približne jedna tretina, sa vstrebáva v proximálnych segmentoch tenkého čreva. Intestinálna absorpcia solubilnej ionizovanej formy vápnika prebieha v dvoch separátnych krokoch:

1) príjem vápnika na strane sliznice a

2) prechod na seróznu stranu intestinálneho epitelu.

Vápnik sa rýchlejšie absorbuje vo forme chloridu, ako vo forme karbonátu, glycerolfosfátu alebo orotofosfátu.

U ľudí sa extracelulárne nachádza len asi 1 000 mg vápnika. Najväčším depotným miestom vápnika je skelet, v ktorom sa nachádza asi 1,2 kg tohto prvku, z čoho 4000 mg je dostupné pre rýchlu výmenu s extracelulárnym prostredím a na pufrovanie plazmatického vápnika. Kostra je dynamickým tkanivom, ktoré podlieha každodennej transfromácii. V tomto procese sa približne 500 mg vápnika extrahuje z extracelulárneho depa pre formovanie nového tkaniva kosti a rovnaké množstvo starého kostného tkaniva sa rozkladá.

Koncentrácia vápenatých iónov v extracelulárnej tekutine a v plazme kolíše len veľmi málo, čo umožňuje udržať fyziologické hladiny intracelulárneho vápnika. Fyziologické hladiny vápnika v plazme sú v rozmedzí od 8,6 do 10,6 mg/dl. Približne polovica z celkovej koncentrácie vápnika v plazme je v ionizovanej forme: 40 % z vápnika sa viaže na proteíny, hlavne na albumín a 10 % vytvára neionizované ultrafiltrovateľné komplexy, ako je uhličitan vápenatý. Rovnováha medzi ionizovateľnou formou a viazanou formou závisí od pH. Alkalózou sa zvyšuje väzba vápnika na proteíny a znižuje sa koncentrácia ionizovaného vápnika, kým pri acidóze sa pozoroval opačný efekt. Hladina vápnika v krvi sa fyziologicky udržuje vo veľmi úzkych hraniciach senzitívnym spätným mechanizmom regulácie. Na regulácii homeostázy minerálnych látok a aj vápnika sa podieľajú tri hlavné orgány (obličky, gastrointestinálny trakt a kostra) a tri hlavné hormóny (vitamín D, parathormón PTH a kalcitonín CT).

Vápnik sa vylučuje predovšetkým močom, menej stolicou, pankreatickou šťavou, žlčou, slinami, potom i mliekom.

5.3. Preklinické údaje o bezpečnosti

Embryotoxické, cytotoxické, teratogénne a karcinogénne účinky chloridu vápenatého nie sú známe.

6. FARMACEUTICKÉ INFORMÁCIE

6.1. Zoznam pomocných látok

aqua ad iniectabilia

6.2. Inkompatibility

Soli vápnika by sa nemali podávať s bikarbonátom, pretože dochádza k precipitácii.

6.3. Čas použiteľnosti

5 rokov

6.4. Špeciálne upozornenia na uchovávanie

Uchovávať pri teplote od 10 °C do 25 °C.

Uchovávajte mimo dosahu a dohľadu detí.

6.5 Druh obalu a obsah balenia

Ampula z bezfarebného skla s etiketou, výlisok z PVC, papierová škatuľka, písomná informácia pre používateľov.

Veľkosť balenia: 5 ampúl po 10 ml

6.6 Špeciálne opatrenia na likvidáciu a iné zaobchádzanie s liekom

Výdaj lieku viazaný na lekársky predpis.

Nepoužitý liek vráťte do lekárne.

7. DRŽITEĽ ROZHODNUTIA O REGISTRÁCII

BB Pharma a.s., Pod Višňovkou 1662/21, 140 00 Praha 4, Česká republika

8. REGISTRAČNÉ ČÍSLO

39/0775/92-S

9. DÁTUM PRVEJ REGISTRÁCIE/ PREDĹŽENIA REGISTRÁCIE

14.12.1992

10. DÁTUM REVÍZIE TEXTU

Október 2011

Príloha č. 2

Obsah priloženého CD

Obsahom priloženého CD sú sady dokumentov na ktorých sme riešenie trénovali a testovali. Okrem toho sú súčasťou aj zdrojové kódy práce. Program bol vyvíjaný v programovacom jazyku Java vo vývojovom prostredí Eclipse. Pre úspešné spustenie programu by malo stačiť importovať priečinok do vývojového prostredia Eclipse. Okrem toho je potrebné nastaviť prihlasovacie údaje do korpusu ako premenné prostredia.

data	- priečinok obsahuje sadu dokumentov, ktoré slúžili na vyhľadávanie pravidiel
lib	- priečinok obsahuje všetky potrebné knižnice na beh programu
ref	- priečinok obsahuje sadu dokumentov, ktoré slúžili na vyhodnocovanie úspešnosti
ref.txt	- ručne extrahované aktívne látky, ktoré slúžili na porovnanie výstupu programu
sql	- jednoduchá SQL schéma ako proof-of-concept pre ukladanie odstavcov
src	- priečinok obsahuje všetky zdrojové súbory