Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a umelej inteligencie

Dolovanie pojmov a sémantických relácií z textov v prirodzenom jazyku

Vedúci diplomovej práce:
doc. Ing. Tomáš Sabol, CSc.

Diplomant:
Viliam Ročkai

Konzultant diplomovej práce:
Ing. Róbert Kende

Košice 2005

Čestné prehlásenie

Prehlasujem, že som diplomovú prácu vypracoval samostatne s využitím uvedenej odbornej literatúry.

V Košiciach dňa 25.04.2005

. . . . . .

vlastnoručný podpis

Poďakovanie

Moje poďakovanie za odbornú spoluprácu a cenné pripomienky pri vypracovaní diplomovej práce patrí vedúcemu diplomovej práce doc. Ing. Tomášovi Sabolovi Csc. a konzultantovi Ing. Róbertovi Kendemu.

Názov práce :	Dolovanie pojmov a sémantických relácií z textov v prirodzenom jazyku.
Katedra :	Katedra kybernetiky a umelej inteligencie, TU FEI Košice
Autor :	Viliam Ročkai
Vedúci DP :	doc. Ing. Tomáš Sabol, CSc.
Konzultant DP :	Ing. Róbert Kende
Dátum :	02.05.2005
Kľúčové slová :	Sémantické okolie, asociácie, teória asociačného učenia pojmov, koncept

Anotácia : Táto diplomová práca sa zaoberá automatickým generovaním ontologických foriem znalostí. V prvej časti je popísaný prehľad súčasného stavu danej problematiky. Ďalej je v práci popísaná teória asociatívneho učenia pojmov, ktorá je založená na novom prístupe k danej problematike. Na základe tejto teórie je navrhnutá reálna implementácia a na nej vykonané experimenty.

Thesis title :	Mining of Concepts and Semantic Relations from Texts in Natural Language.
Department :	Department of Cybernetics and Artificial Intelligence, TU FEI Košice
Author :	Viliam Ročkai
Supervisor :	doc. Ing. Tomáš Sabol, CSc.
Tutor :	Ing. Róbert Kende
Date :	02.05.2005
Keywords :	Semantic surrounding, associations, theory of associative leraning, concept

Annotation : This work discuss about automatic ontology knowledge representation genration. First part gives brief view about actual state of this knowledge representation. Next the heory of asociation learning is denoted. An implementation of this theory is noted in the next chapter. The end of the work are experiments processed upon this implementation.

Obsah

Úvod

Ipsa scientia potestas est. (Sir Francis Bacon)

Myšlienka o nevyhnutnej potrebe a hybnej sile znalostí bola vyslovená už v 16. storočí v období renesancie. Ale až dnes, v modernom svete informácií, nachádzame pre túto vetu pravé uplatnenie. Svet je preplnený informáciami, ľudstvo sa topí v mori zbytočných dát. Pojem znalosti ako užitočnej informácie nadobúda nový rozmer, šetrí čas i peniaze a posúva dopredu výskum.

Ontológia ako filozofická disciplína, veda o bytí, je ešte staršia. Keďže sa tento pojem dnes používa snáď až príliš často, začína byť zavádzajúci. Ontológia sa stala modernou, žiadanou, no doteraz nemá ucelenú definíciu.

Je fascinujúce ako sa my, ľudia, necháme radi inšpirovať našou vlastnou minulosťou. Ešte prekvapivejšie je, že filozofia spred pár storočí nás dokáže inšpirovať dodnes. Renesancia, inšpirovaná antickou kultúrou vyzdvihla silu znalosti, novoveká filozofia pomohla k lepšiemu pochopeniu ontológií. V dnešnej dobe nachádzame obrovskú silu v spojení týchto pojmov. Využitie ontológií ako formu reprezentácie znalostí.

Žijeme vo svete faktov a pracoviská sa zapĺňajú obrovskými kopami dokumentov. Čím ďalej, tým je orientácia v nich ťažšia, znalosti sa strácajú v dátach. Je nevyhnutné nájsť správne prostriedky k ťažbe týchto znalostí. Dolovanie znalostí nadobúda na dôležitosti každým dňom. Na manuálne vyhľadávanie už hlavy nestačia, budúcnosť je otvorená automatizácii!

Táto práca pojednáva o automatickom dolovaní znalostí z textov v prirodzenom jazyku. Teória asociatívneho učenia pojmov, vypracovaná pánom R. Hecht-Nielsenom bude možno v budúcnosti aj vďaka tejto práci podstatným mechanizmom pri automatickom budovaní ontologických lexikónov, stane sa srdcom jazykových prekladačov, možno pomôže hudobníkom pri tvorbe nových hudobných diel. Dizajn kopírujúci prírodu sa uplatňuje vo všetkých technických oblastiach, tak prečo sa neinšpirovať vlastným mozgom? Učme sa z minulosti a neodsudzujme nové teórie, kým sa naozaj nepresvedčíme o ich nefunkčnosti. Správne riešenie je často tá najjednoduchšia cesta...

1 Formulácia úlohy

- Spracovať úvod do teórie ontológií ako formy reprezentácie znalostí a podať prehľad súčasného stavu, čo sa týka možností automatického vytvárania "ontologických" foriem znalostí.
- Spracovať popis teórie asociatívneho učenia pojmov.
- Implementovať model asociatívneho učenia pojmov. Model musí zahŕňať všetky podstatné aspekty teórie, avšak môže byť vhodne zjednodušený zameraním sa na doménu textov v prirodzenom jazyku.
- Navrhnúť, realizovať a vyhodnotiť experimenty. Experimenty majú demonštrovať podstatné vlastnosti implementovaného modelu.

2 Ontológie ako spôsob reprezentácie znalostí

2.1 Definícia pojmu znalosť

Je veľmi dôležité nezamieňať si pojem „znalosť“ s pojmom „informácia“. V bežnom živote je zámena týchto pojmov bežná a nespôsobuje žiadne problémy pre prax UI, je však potrebné definovať si pojem „znalosť“ presne a nezamieňať ju s informáciou.

Definícia znalosti by mohla znieť:
Znalosť je informácia, ktorú možno efektívne využiť.

Lepšie je ale definovať znalosť pomocou niektorých charakteristík:

- Znalosť je ľudská schopnosť, napr. schopnosť vytvárať úsudky či už v prítomnosti alebo budúcnosti. Je to transformácia informácie osobou.
- Získavanie znalostí je dynamický proces. Či jedinec prijme znalosť korektne z určitého zdroja, závisí hlavne od dvoch faktorov:
    - podobnosť medzi kontextom prijímateľa (situácia, minulosť, predpoklady, …) a popisovaným kontextom,
    - stupeň zhody medzi tým, ako je látka štruktúrovaná a ako sa štruktúra domény javí pre prijímateľa.
- Znalosť je generatívna a viacrozmerná. Znamená to, že môže byť analyzovaná a na jej základe možno vytvárať abstrakcie. Znalosť umožňuje nositeľovi vytvárať nové tvrdenia o subjekte a nie len reprodukovať naučené.
- Znalosť je komplikovaná, je to útvar organizovanej informácie, získavanej postupne v blokoch. **Chyba! Nenalezen zdroj odkazů.**

3 Úvod do ontológií

3.1 Pojem ontológia

Pojem „ontológia“ má mnoho významov a je úzko prepojený s kontextom, v ktorom sa vyskytuje. Kým v kontexte zdieľania znalostí pod ním rozumieme špecifikáciu konceptualizácie, môžeme ho rovnako chápať aj ako filozofickú disciplínu, či slovník používaný logickou teóriou. Tento pojem môže rovnako označovať aj :

- Neformálny konceptuálny systém
- Formálny sémantický popis
- Reprezentáciu konceptuálneho systému pomocou logickej teórie
- Špecifikáciu logickej teórie

3.2 Ontológia ako filozofická disciplína

Ontológia ako učenie o súcne, čiže o tom, čo existuje, tvorí spolu s učením o božskom bytí od vzniku filozofie, centrum metafyziky. Systematické vymedzenie ontológie v podobe vedy, to znamená v súvislosti s najvšeobecnejšími určeniami bytia, významami bytia a pojmami bytia, sa nachádza v diele Ch. Wolffa "Ontológia" (1730). Základnú otázku ontológie „Čo je bytie?“ vyslovil už v období antiky Aristoteles, ktorý sa všeobecne pokladá aj za jej zakladateľa aj napriek tomu, že so slovom „ontológia“ sa prvýkrát stretávame až začiatkom 17. storočia pri rozvoji novovekej filozofie **Chyba! Nenalezen zdroj odkazů.**.

3.3 Ontológia v umelej inteligencii

Pojem „ontológia“ bol prevzatý z filozofie, pretože pre systémy s umelou inteligenciou „existuje“ práve to, čo sa dá reprezentovať. Pokiaľ je znalosť domény reprezentovaná deklaratívnym formalizmom, množinu objektov, ktorá môže byť týmto spôsobom reprezentovaná, nazývame univerzom. Táto množina objektov a popis ich vzájomných vzťahov sú premietnuté do jazyka, ktorým program reprezentuje znalosti. Tak teda môžeme v kontexte umelej inteligencie popísať ontológiu definovaním množinou reprezentovaných termov.

3.4 Ontológia ako formálna špecifikácia

Znalostný systém je programový systém, ktorý je špecifický tým, ako sú v ňom znalosti organizované, začlenené a využívané. Znalosti sú v rámci tohto systému perzistované explicitne v samostatnej modulárnej štruktúre - báze znalostí, takže pri jej modifikovaní nie je nutné modifikovať aj ostatné časti systému. Tieto systémy pracujú a komunikujú pomocou výrazov formálnej reprezentácie znalostí. V distribuovanom UI prostredí si navzájom prenášajú a vymieňajú znalosti. Na umožnenie tohto zdieľania znalostí je potrebné ujednotiť komunikáciu na troch úrovniach

- Formáte reprezentačného jazyka
- Formáte komunikačného protokolu agentov
- Špecifikácie obsahu zdieľaných znalostí

Ontológie riešia práve poslednú požiadavku na ujednotenie komunikácie – špecifikáciu obsahu zdieľaných znalostí.. Zavádzajú dohodu o znalostiach ako spoločné predpoklady a modely reálneho sveta [4].

Najvýznamnejší výskum v oblasti ontológií sa vedie na Stanfordskej Univerzite v „Laboratóriu znalostných systémov“. Odtiaľ pochádza aj s menšími pripomienkami v UI komunite prijímaná definícia ontológie od T. R. Grubera:

Ontológia je explicitnou špecifikáciou konceptualizácie [4].

Ontológia je popisom (ako formálna špecifikácia programu) konceptov a ich vzťahov. Oblasť záujmu tak predstavuje množina konceptov a vzťahov medzi nimi a nazýva sa univerzom (universe of discourse). Ontológia má slúžiť na zdieľanie a opätovné využívanie znalostí. V tomto kontexte slúži ontológia na vytváranie tzv. ontologických záväzkov (ontological commitments) pre množinu agentov. Agent sa viaže (commits) na ontológiu, ak jeho pozorovateľné akcie sú konzistentné s definíciami v tejto ontológii. Spoločná ontológia definuje slovník, pomocou ktorého sa medzi agentmi vymieňajú dotazy a tvrdenia. Záväzok ku spoločnej ontológii zaručuje

jednotnosť, ale nie úplnosť, keďže agent viazaný na ontológiu, nemusí zodpovedať všetky otázky, ktoré sa dajú pomocou zdieľaného slovníka formulovať.

Telo formálne reprezentovanej znalosti je založené na konceptualizácii : objektoch, konceptoch a ostatných entitách, o ktorých predpokladáme, že existujú v nejakej oblasti záujmu a tiež vzťahy medzi nimi (Genesereth & Nilsson, 1987). Konceptualizácia je abstraktom, zjednodušeným pohľadom na skutočný svet, ktorý chceme pre nejaký účel reprezentovať. Každý agent je viazaný na nejakú konceptualizáciu či už explicitne alebo implicitne.

Definícia ontológie z pohľadu znalostného inžinierstva [1]:

Ontológia pre telo znalostí obsahujúcich určitý problém alebo oblasť popisuje taxonómiu konceptov pre tento problém či oblasť, ktorá definuje sémantickú interpretáciu znalostí.

Táto definícia vzhľadom na súčasný stav prakticky realizovaných ontológií postačuje, ale z niektorých teoretických hľadísk nie je celkom uspokojivá. Sémantická interpretácia zahŕňa aj faktické situácie a netýka sa vždy iba taxonómie.

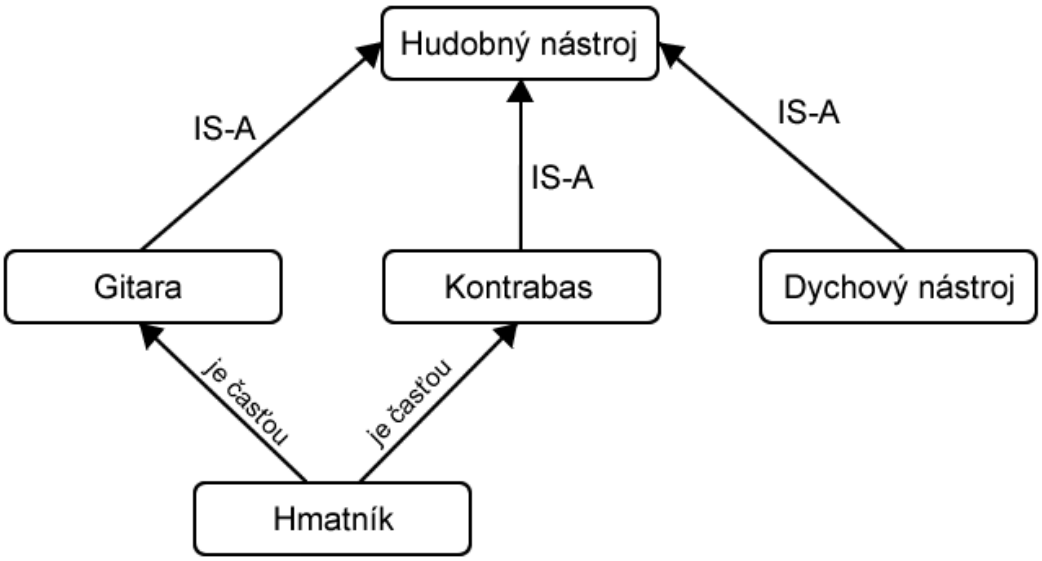
N. Guarino predkladá aj ďalšie definície [6], stavia sa k nim kriticky a odhaľuje ich nedostatky z viacerých hľadísk. Za uspokojivú pokladá túto definíciu:

Ontológia je explicitná, čiastočná špecifikácia konceptualizácie, ktorá sa dá vyjadriť ako meta-úrovňový náhľad na množinu možných doménových teórií za účelom modulárneho návrhu, prestavby a opätovného použitia systémových častí súvisiacich so znalosťami [11].

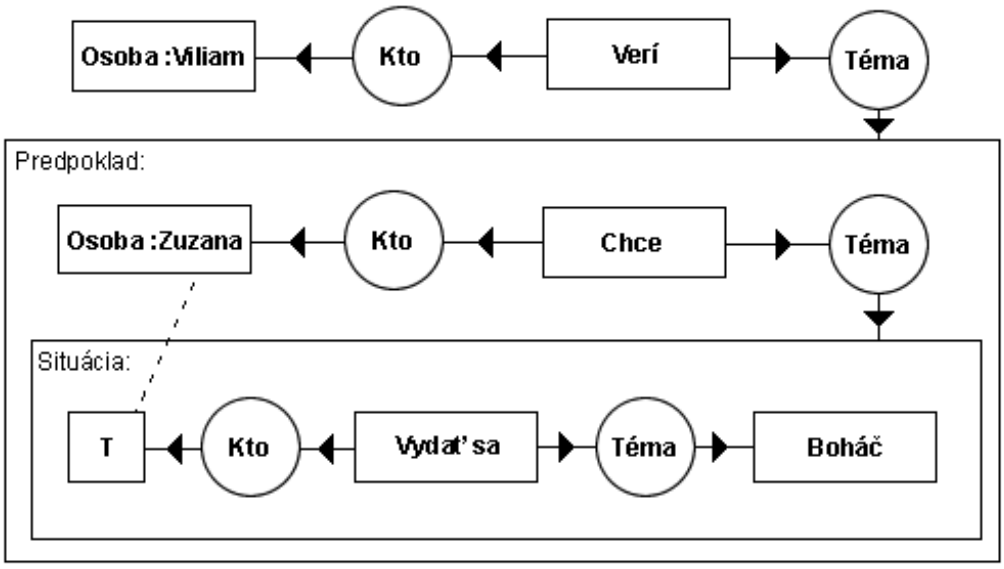
Ontológia sa pokladá za meta-úrovňový popis reprezentácie znalostí. Neobsahuje iba znalosti, ktoré súvisia s danou doménou, ale explicitne definuje a kladie ohraničenia na ich formu. To dodáva ontológiám ich vlastnosť opätovného využitia a zdieľania znalostí.

3.5 Príklady ontológií

V praxi tvorí ontológiu väčšinou taxonómia konceptov, vyjadrenie vzťahov medzi nimi a ohraničenia pre ich tvorbu. Ľudskej predstavivosti najprijateľnejšia reprezentácia ontologických znalostí je asi reprezentácia v podobe grafu. V grafoch väčšinou predstavujú uzly, koncepty a hrany vzťahy, ktoré medzi nimi nastávajú. Na vizualizáciu ontológií ale môžeme rovnako využiť Konceptuálne grafy podľa Sowu [12], alebo sémantické siete.



Obr. 1 Ontológia o hudobných nástrojoch, z ktorej možno odvodiť, že hmatník je časťou hudobného nástroja



Obr. 2 Sowov konceptuálny graf popisujúci zložitejšiu situáciu.

Konceptuálny graf na obrázku č. 2 vyjadruje vetu “Viliam verí, že sa Zuzana chce vydať za boháča”. Viliam je „prežívateľom“ konceptu „Verí“, ktorý je prepojený s témou predpokladu, v ktorý Viliam verí. Ohraničené políčko predpokladu obsahuje ďalší konceptuálny graf, ktorý vyjadruje, situáciu, ktorú by chcela Zuzana raz zažiť. Situácia je znova popísaná ďalším vnoreným konceptuálnym grafom., ktorý hovorí, že Zuzana (reprezentovaná konceptom [T]) sa vydá za boháča. Prerušovaná čiara, koreferenčné?? prepojenie, ukazuje, že koncept [T] v políčku situácie ukazuje na rovnakú osobu ako koncept „Osoba: Zuzana“.

Zápis tejto ontológie v textovej podobe by vyzeral takto:

[Osoba: Viliam] <= (Kto) <= [Verí] =>Téma -

[Predpoklad : [Osoba: Zuzana *x] <= (Kto) <= [Chce] => (Téma)] –

[Situácia: [?x] <= (Agnt) <= [Zuzana] => (Téma) => [Fotograf]]

3.6 Existujúce systémy využívajúce ontológie

3.6.1 Cyc

Cyc je projektom, ktorý sa snaží zhromaždiť komplexnú ontológiu a databázu tvorenú každodennými praktickými znalosťami s cieľom umožniť UI aplikáciám rozhodovanie podobné ľudskému.

Projekt vznikol v roku 1984 pod vedením Douga Lenata ako časť „Microelectronics and Computer Technology Corporation“. Názov „Cyc“ (odvodený od slova encyclopédia) je registrovanou obchodnou značkou firmy Cycorp, Inc. V Texase. Báza znalostí, ktorú systém využíva, je majetkom spoločnosti, ale jej menšia verzia vydaná pod názvom OpenCyc je vydaná pod open source licenciou.

Typickými príkladmi znalostí reprezentovaných v databáze je napríklad veta „Každý strom je rastlina“ a „Rastliny hynú“. Pri otázke, či hynie strom, dôjde inferenčný systém k logickej odpovedi. Báza znalostí obsahuje vyše milióna ručne napísaných výrokov, pravidiel a jednoznačných myšlienok. Tie sú formulované v jazyku CycL, založenom na predikátovom počte so syntaxou veľmi podobnou Lispu.

Názvy konceptov sa v Cyc označujú ako „konštanty“. Konštanty začínajú znakom „#$“ a môžu označovať individuálne položky, množiny, pravdivostné funkcie

a funkcie. Pravdivostné funkcie sa delia na logické spojky, kvantifikátory a predikáty. Funkcie vracajú nové termy na základe vstupných parametrov. Najdôležitejšími predikátmi sú #$isa a #$genls. Kým #$isa popisuje, že položka je inštanciou nejakého zoznamu, #$genls vyjadruje, že množina je podmnožinou inej množiny. Fakty o konceptoch sú vyjadrené pomocou špecifických CycL viet. Predikáty sú zapísané pred argumentami v zátvorkách:

(#$isa #$BillClinton #$UnitedStatesPresident)

"Bill Clinton patrí do množiny U.S. presidents" a

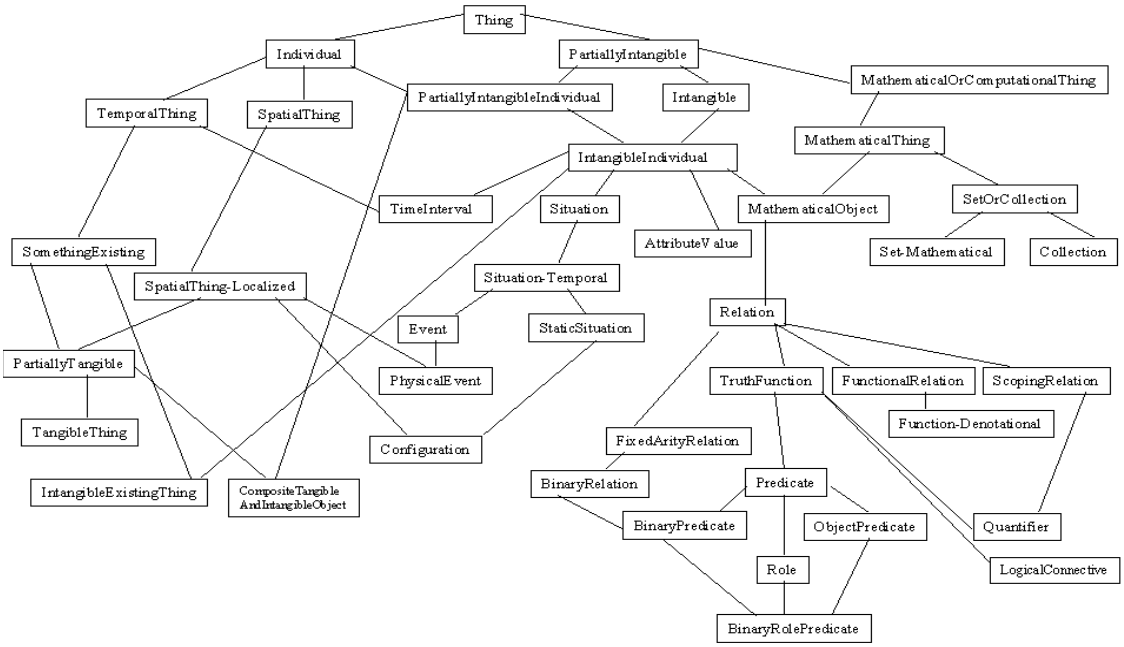
(#$genls #$Tree-ThePlant #$Plant)

"Všetky stromy sú rastliny".

Vety môžu obsahovať aj premenné, označené reťazcom začínajúcim znakom „?“:

```
(#$implies
  (#$and
    (#$isa ?OBJ ?SUBSET)
    (#$genls ?SUBSET ?SUPERSET))
  (#$isa ?OBJ ?SUPERSET))
```

čo interpretujeme ako „ak OBJ je inštanciou množiny SUBSET a SUBSET je podmnožinou množiny SUPERSET, potom OBJ je inštanciou množiny SUPERSET“[13].



Obr. 3 Hierarchická štruktúra ontológie Cyc

3.6.2 WordNet

WordNet je sémantický lexikón anglického jazyka. Zoskupuje anglické slová do množín synoným, nazývaných synsety, poskytuje krátke definície a záznamy rôznych sémantických relácií medzi týmito synsetmi. Má to dvojaký účel: poskytnúť slovník a tezaurus, ktorý by bol viac intuitívne použiteľný a poskytnúť podporu automatickej analýzy textov a aplikácií pre UI. Databáza a softwarové nástroje sú vydané pod BSD style licenciou a môžu byť voľne stiahnuté a používané. Databázu je možné prezerať si aj on-line.

WordNet vznikol v roku 1985 a je udržiavaný na pôde Princetonskej univerzity v Laboratóriu kognitívnych vied pod vedením profesora psychológie George A. Millera.

V roku 2003 obsahovala databáza okolo 140 000 slov organizovaných do viac ako 110 000 synsetov s celkovým počtom 195 000 slovných párov.

WordNet rozlišuje medzi podstatnými menami, slovesami, prídavnými menami a príslovkami na predpoklade, že aj v ľudskom mozgu sa tieto štruktúry ukladajú rozdielne. Každý synset obsahuje množinu synonymických slov alebo slovných spojení,

ktoré produkujú špecifický význam (napr. Vysoká škola). Slová sa zvyčajne vyskytujú vo viacerých synsetoch naraz. Význam synsetov je ďalej doladený krátkymi poznámkami. Typickým príkladom synsetu s poznámkou je:

správny, vhodný, akurátny -- (najvhodnejší pre dané špecifické účely; „správny čas oraťˮ, „vhodný moment na ústupˮ, „akurátny prístup k danej problematikeˮ).

Každý synset je relačne prepojený na iné synsety. Tieto relácie závisia od typu slova. Jedná sa o tieto typy relácií.

- Synonymá – synsety s podobným významom
- Hypernymá - *Y* je hypernymom k *X,* ak každé *X* je (ako) *Y*
- Hyponymá - *Y* je hyponymom k *X,* ak každé *Y* je (kind of) *X*
- Holonymá - *Y* je holonymom k *X,* ak *X* je časťou *Y*
- Meronymá - *Y* je meronymom k *X,* ak *Y* je časťou *X*
- Antonymá – slová opačného významu
- Rovnocenné pojmy – slová zdieľajúce bežné hypernymum

WordNet poskytuje premennú „polysemy countˮ, ktorá vyjadruje počet synsetov, ktoré dané slovo obsahujú. Ak sa dané slovo vyskytuje vo viacerých synsetoch (t.z. má viacero významov), tak je zrejmé, že niektoré významy sú bežnejšie ako iné. WordNet to kvantifikuje pomocou „frequency scoreˮ. V niekoľkých vzorkách textov sp?? všetky slová sématicky označené korešpondujúcim synsetom a neskôr je spočítané koľkokrát sa vyskytli s daným špecifickým významom.

Databázové rozhranie vie z užívateľského vstupu dedukovať koreňovú formu slova a iba do databázy sa ukladajú iba tieto formy [15].

3.6.3 TextToOnto

TextToOnto je balík nástrojov postavených nad technológiou KAON za účelom podporiť procesy ontologického inžinierstva pomocou techník dolovania znalostí. Poskytuje súbor nezávislých nástrojov na automatické a semi-automatické získavanie, údržbu a čistenie ontológií. Súčasná distribúcia balíka obsahuje tieto nástroje:

• TaxoBuilder na budovanie hierarchií konceptov,

• TermExtraction pre pridávanie konceptov do ontológií,

• InstanceExtraction pre pridávanie inštancií do ontológií,
• RelationExtraction na semi-automatické učenie koncpetuálnych relácií,
• RelationLearning na automatické a semi-automatické učenie relácií,
• OntologyComparison na porovnanie dvoch ontológií,
• OntologyPruner na adaptáciu ontológie pre doménovo špecifický korpus.

TaxoBuilder

Pri tvorbe novej ontológie je tradičným postupom zapĺňanie OI-Modelu konceptmi a inštanciami. TaxoBuilder automaticky vytvorí hierarchiu konceptov z najfrekventovanejších termov obsiahnutých v korpuse ich pridávaním do prázdneho OI-Modelu. Užívateľovi je poskytnuté prostredie, v ktorom si môže špecifikovať korpus ako aj počet zvažovaných slov. Môže si zvoliť z dvoch prístupov tvorby taxonómie :

- FCA prístupe založenom na silných selekčných reštrikciách, ktorými slovesá pôsobia na svoje argumenty.
- Kombinácia Hearst-Patternov, WordNetu a rozličných heuristík, ktoré si volí užívateľ sám.

TermExtraction

Tento nástroj môže byť využitý na tvorbu nových konceptov z možných relevantných termov obsiahnutých v korpuse. Parametre, ktoré sú k dizpozícii na posúdenie relevantnosti termu sú jazyk (angličtina, nemčina a taliančina), ako aj maximálny počet slov na term, ale aj minimálna frekvencia výskytov.

InstanceExtraction

Potom, ako sa pridá jeden alebo viac konceptov do OI-Modelu, mali?? by sa vyextrahovať ich inštancie z korpusu. InstanceExtraction je nástrojom na automatické a semi-automatické učenie inštancií aplikáciou rôznych vzorov, vybraných užívateľom. Pri semi-automatickom móde je každý termový kandidát navrhnutý užívateľovi ako možné riešenie, ktoré musí potvrdiť.

RelationExtraction

Je jedným z dvoch nástrojov balíka TextToOnto na rozšírenie OI-Modelu pridávaním konceptuálnych (a taxonomických) relácií medzi konceptami a inštanciami, ktoré už sú naučené. Na rozdiel od nástroja RelationLearning, RelationExtraction poskytuje iba semi-automatické učenie. Oba prístupy môžu byť využité na extrakciu zoznamu kandidátskych relácií z textu. Zatiaľ čo prvý prístup je založený na asociačných pravidlách, druhý aplikuje množinu vzorov veľmi podobných tým od Hearsta. Užívateľ má možnosť z daného zoznamu vybrať množinu kandidátov, ktorú pridá do OI-Modelu, či už ako vlastnosť, alebo taxonomickú reláciu.

RelationLearning

Tento nástroj narozdiel od RelationExtraction podporuje automatické aj semi-automatické učenie konceptuálnych relácií. Pre semi-automatickom móde sú užívateľovi navrhnuté meno, doména a rozsah pre každú reláciu.

OntologyComparison

V prípade, že chceme ohodnotiť OI-Model, ktorý sa naučil automaticky alebo semi-automaticky z korpusu, musíme ho porovnať s iným, či už automaticky naučeným, alebo manuálne zostrojeným modelom ontológie. Tento modul porovnáva dva alebo viac OI-Modelov s prihliadnutím na lexikálne a konceptuálne vlastnosti modelu. Jedná sa napríklad o taxonomické alebo relačné prekrytie, ktoré môže byť osobitne nastavené užívateľom.

OntologyPruner

Po získaní OI-Modelu z príliš špecifického korpusu je niekedy potrebné adaptovať daný model na požiadavky doménovo špecifickejšieho korpusu. S nástrojom OntologyPruner nemusíme vytvoriť novú, špeficikejšiu ontológiu. Navrhne nám na základne frekvencií koncepty, ktoré je potrebné odstrániť. Parametre, ktoré má k dispozícii užívateľ sú jazyk a hornú hranicu kumulatívnej frekvencie, po prekročení ktorej sú termy pokladané za relevantné [7].

4 Teória asociatívneho učenia pojmov

Metóda asociatívneho učenia pojmov je nekontrolované učenie (učenie bez učiteľa), ktoré prebieha nad tokom symbolov. Cieľom učenia je získať definície symbolov, ktoré sémanticky modeluje (zachytáva ich význam).

Podstata učenia je v indukovaní asociácií medzi jednotlivými symbolmi (ukotvenými tokenmi) na základe pozorovaných výskytov relevantných tokenov v kontextovom okne, cez ktoré učenie prebieha. Podmienka pre vytvorenie asociácie medzi dvoma tokenmi je ich "štatisticky" nenáhodný výskyt. Ak dôjde k vzniku asociácie, počas učenia sa mení už len jej váha. Sila asociácie je odvodená od pozorovaných spoločných výskytov dvoch tokenov počas učenia. Každá asociácia je v tejto teórii ponímaná ako symbol definujúca relácia. Takouto definíciou sa dané symboly "ukotvujú". Symbol je teda definovaný len svojím asociatívnym okolím.

Ďalší dôležitý aspekt teórie spočíva v zachytení kontextu. Kontext je modelovaný ako množina asociácií. Tieto množiny (fascikle) vznikajú učením asociácií cez kontextové okno, v ktorom sú rozpoznané symboly zoradené podľa pravidiel syntaxe generujúceho jazyka. Vo fascikloch sú zachytené asociácie medzi jednotlivými symbolmi. Každý fascikel zachytáva asociácie pre konkrétnu kontextovú vzdialenosť v okne. Asociácie sú jediným druhom znalostí, s ktorou táto teória pracuje.

Základná hytpotéza tejto teórie zakladá na neuro-fyziologickom modeli talamo-kortikálneho spracovania informácií R.Hecht-Nielsena. Hecht-Nielsen predpokladá existenciu ustáleného lexikónu „symbolov" v talame ľudského mozgu, ktorý sa vyvinie v kritickom období vývoja jedinca. Učenie prebieha vytváraním prepojení medzi kortikálnymi regiónmi mozgovej kôry, ktoré je založené na klasickom Hebb-ovom učení.

Implementovaním zjednodušeného modelu talamo-kortikálneho spracovania informácií sa budeme v tejto práci pokúšať zovšeobecniť model učenia založený na asociovaní tokenov – senzorických invariant a dokázať emergenciu sémantiky použitím, najprimitívnejšej formy znalosti: asociácie.

4.1 Token

Token, ako je definovaný v teórii talamo-kortikálneho spracovania informácií Hecht-Nielsena [8], je množina excitovaných neurónov v nejakom kortikálnom regióne. Zovšeobecnene, budeme token v ďalšom chápať ako senzorickú invariantu z nejakej konečnej množiny. Túto množinu budeme ďalej nazývať lexikónom. Token teda reprezentuje rozpoznaný senzorický vstup. Neuro – fyziologickou opodstatnenosťou takejto "symbolickej" reprezentácie informácie sa podrobne zaoberá [8].

4.2 Symboly a Asociácie

Symbolom budeme v ďalšom chápať akýkoľvek token s aspoň jednou asociáciou. Hovoríme, že táto asociácia ukotvuje tento symbol. Symbol je vo všeobecnosti ukotvený množstvom asociácií. Asociácia je elementárna forma znalosti, s ktorou teória pracuje a je fundamentálnym aspektom tejto teórie. Asociácie sa vzťahujú k atribútom symbolov. Atribút symbolu chápeme ako množinu asociácií daného symbolu k iným symbolom.

4.2.1 Vzájomná signifikancia

Otázkou je kedy sa asociácie vytvárajú. Základné operácia, ktorú tu využijeme, vychádza z vyhodnocovania vzájomnej signifikancie S(i,j). Táto miera vychádza z klasickej teórie informácií:

$$S(i,j) = \frac{\mathrm{p(i,j)}}{\mathrm{p(i),p(j)}}$$

Signifikancia je definovaná ako pomer vzájomnej pravdepodobnosti tokenov *i,j* k ich apriórnym pravdepodobnostiam.

Je známe že

$$\mathrm{p(i,j) = p(i),p(j), ak\ i\ a\ j\ sú\ nezávilé}$$

V našom prípade to znamená, že ak dva tokeny sa vyskytujú v spoločnom kontexte náhodne (sú teda nezávislé), bude platiť:

$$S(i,j) = \frac{p(i,j)}{p(i),p(j)} < 1$$

Za asociované budeme tokeny považovať ak $S(i,j) > t$, kde t je nejaká hranica významnosti. V učení budeme túto hranicu považovať za parameter.

4.2.2 Váha asociácie

Definícia: **Váha asociácie** symbolov x a y je daná vzorcom:

$$w(x,y) = \frac{p(i,j)}{p(j)}$$

Váha asociácie je jediná znalosť, ktorú zachytávame. Tieto váhy sú reprezentované vo fascikloch fX, čo sú symetrické štvorcové matice s rozmermi veľkosti slovníka označenou symbolom s:

$$fX = \begin{bmatrix} w(0,0) & . & . & . & w(0,s) \\ . & . & & & . \\ . & & w(i,j) & & . \\ . & & & . & . \\ w(s,0) & . & . & . & w(s,s) \end{bmatrix}$$

Vypočítavanie váh $w(i,j)$ spočíva v aproximovaní vyššie uvedeného vzťahu, daným jednoduchým počítaním vzájomných a nezávislých výskytov tokenov i, j pre každý z fasciklov. Aproximáciu vzájomnej signifikancie a váhy počítaním výskytov môžeme teda určiť následovne:

$$S(i,j) = \frac{\frac{C(i,j)}{T}}{\frac{p(j)}{T}}, \quad w(i,j) = \frac{\frac{C(i,j)}{T}}{\frac{C(i)}{T} \cdot \frac{C(j)}{T}}$$

4.2.3 Sémantické okolie symbolu

Definícia: **Sémantické okolie O symbolu** *x* O(x) je taká množina symbolov y, že pre každé x patriace L existuje asociácia a medzi x a y, kde L je lexikón symbolov jazyka.

Sémantické okolie symbolov nám umožňuje modelovať sémantickú podobnosť cez nejaký atribút nasledovne:

$$Si = \frac{|\, O(x) \bigcap O(y) \,|}{|\, O(x) \bigcup O(y) \,|}, \text{ kde i je i - ty atribút,}$$

S leží prirodzene v intervale <0, 1>.

Skladanie podobností cez rôzne atribúty môžeme určiť nasledovne:

$$S = \sum wi.Sai$$

$$Si = \sum wi. \frac{|\, Oi(x) \bigcap Oi(y) \,|}{|\, Oi(x) \bigcup Oi(y) \,|}$$

5 Implementácia modelu asociatívneho učenia pojmov

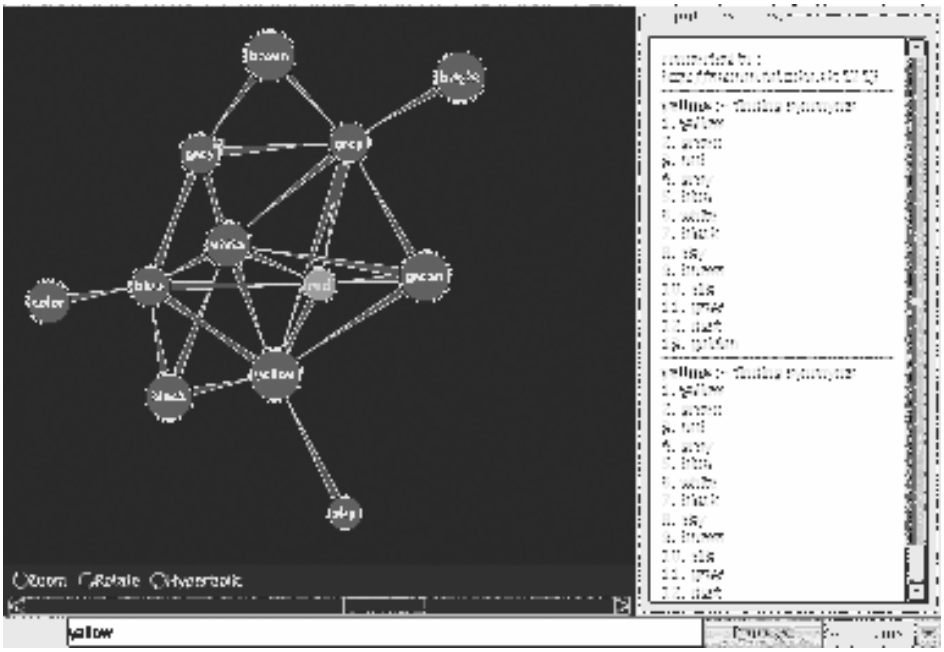
5.1 Popis funkčných blokov implementácie

5.1.1 Úvod

Model pre prácu teórie asociatívneho učenia pojmov je implementovaný s rôznymi obmedzeniami v programovacom jazyku java. Program sa nazýva Beast a pozostáva z dvoch častí :

- BeastServer – základná aplikácia. Obsahuje rozhranie pre internetovú komunikáciu s klientmi a množstvo príkazov pre samotné ovládanie servera.
- BeastApplet – má podobu java appletu. Slúži na prehľadnú vizualizáciu základných funkcií programu.

Špeciálnym dodatkom k celému systému je utilita BeastLearn, ktorá slúži na vypočítavanie štatistík o učení a utilita BeastLexicon, slúžiaca na výpočet závislosti počtu rozpoznaných slov od veľkosti lexikónu.



Obr. 4 Ukážka modulu BeastApplet

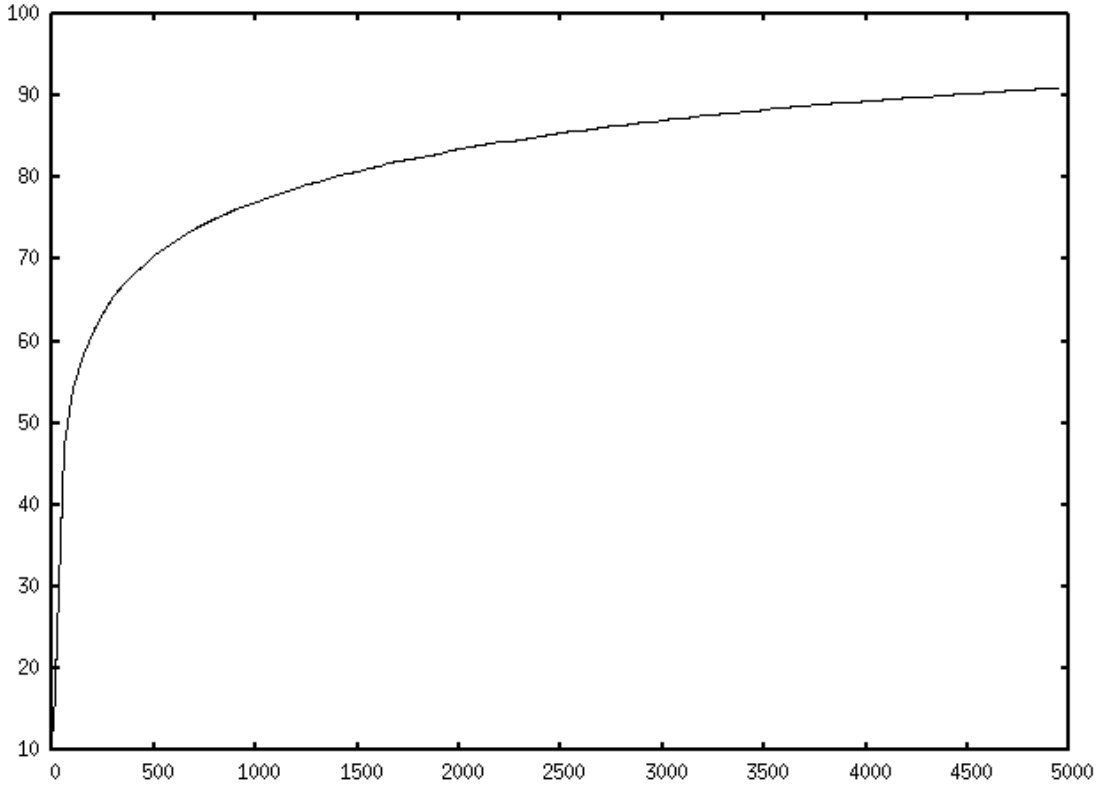
5.1.2 Token

Token je v modeli pre širšie použitie v budúcnosti reprezentovaný ako objekt ľubovoľného typu. Hodnotou tokenu môže byť inštancia ľubovoľnej triedy v jazyku Java. To dáva celému systému možnosť jednoduchej rozšíriteľnosti na inú doménu ako doménu textov. Pre doménu prirodzeného jazyka sme použili na reprezentáciu triedu String, ktorou kódujeme jednotlivé slová. Výnimku tvorí iba tzv. NULL_TOKEN, ktorý označuje token neobsiahnutý v slovníku. Špeciálnou podtriedou Tokenu, je ExcitedToken, ktorý reprezentuje vybudený token, a zaobaľuje okrem jeho logickej hodnoty aj váhu, s ktorou bol, resp. je v danej úlohe vybudený.

Tieto typy hodnôt tokenov slúžia iba na lepšiu čitateľnosť pre užívateľa. Vnútorne systém používa na reprezentáciu hodnoty Tokenu celé čísla, indexy daných tokenov v lexikóne.

5.1.3 Lexikón

Lexikón pozostáva z voliteľne rozsiahlej množiny tokenov a k nim prislúchajúcim ID kľúčov. Každému je priradený osobitný a jedinečný ID kľúč. Veľkosť lexikónu musí byť zadaná ešte pred procesom učenia, lebo lexikón, rovnako ako typ informácie ukladanej v tokenoch zostáva počas celého života (učenia a využívania) programu nemenný. Tokeny sú v lexikóne uložené v podobe zoznamu, v ktorom je každému tokenu priradená hodnota, kvantitatívne vyjadrujúca koľkokrát sa pri budovaní lexikónu model s týmto tokenom stretol. Táto hodnota slúži na zoradenie tokenov podľa ich početnosti, vzhľadom na ďalšie orezanie slovníka, kde sa do nového slovníka dostane voliteľné množstvo najpočetnejších tokenov. Lexikón musí vždy obsahovať NULL_TOKEN, aby bolo možné priradiť index neznámym slovám. Pri experimentoch som použil veľkosť lexikónu 5000 tokenov. K tejto hodnote som dospel experimentálne.

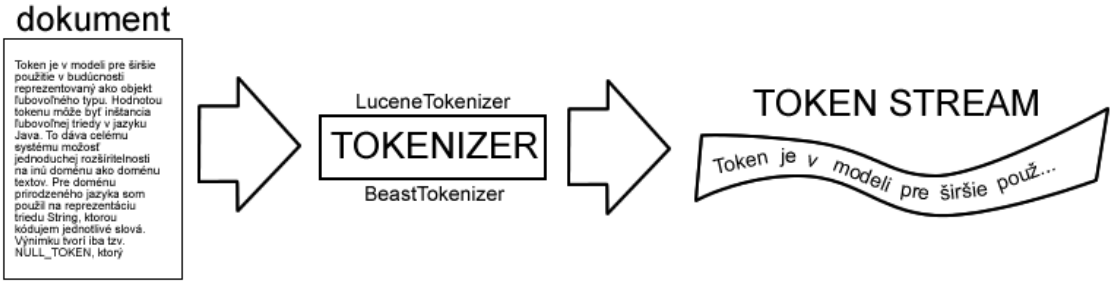


Obr. 5 Závislosť počtu rozpoznaných slov v korpuse od veľkosti lexikónu

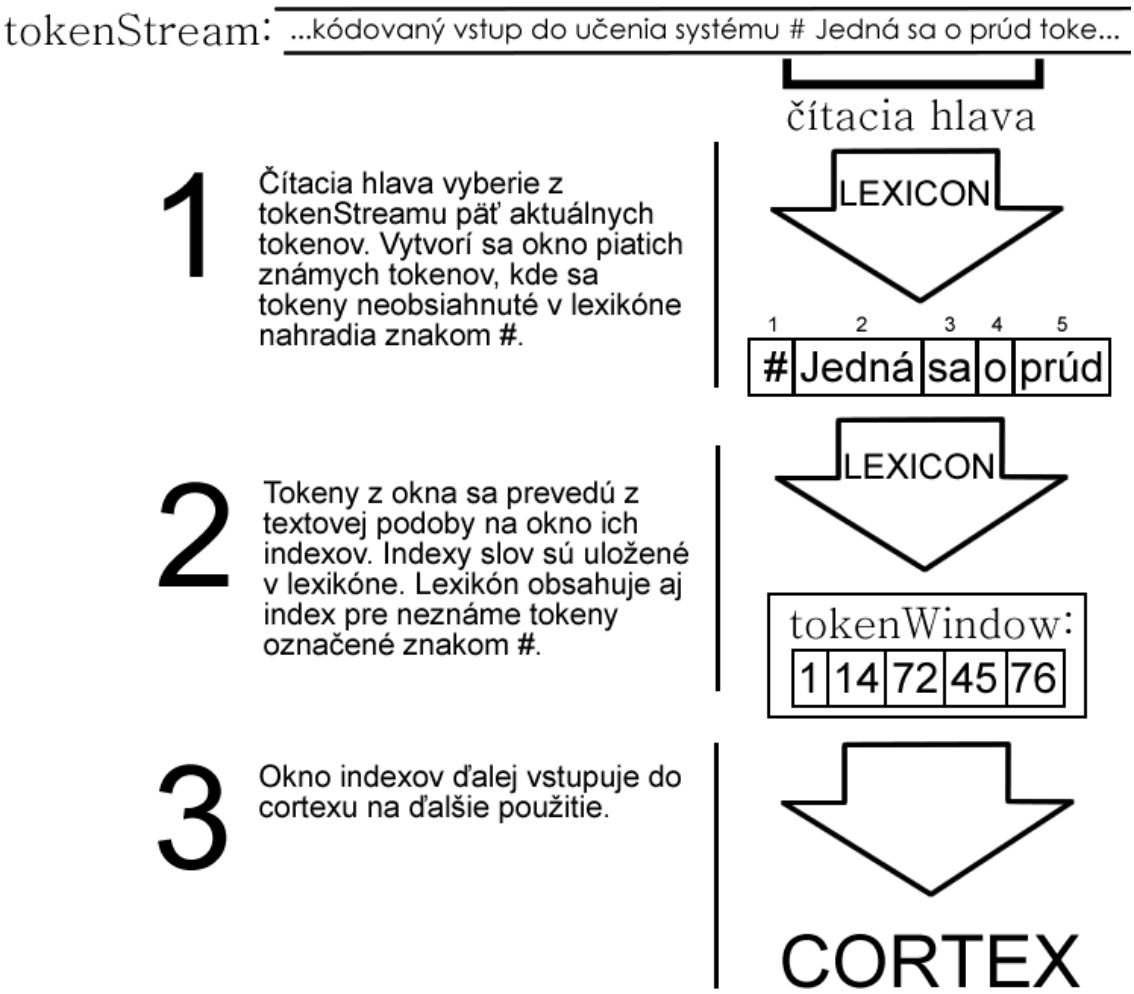
Na obr. 5 je znázornená závislosť percentuálneho počtu rozpoznaných slov v korpuse v závislosti od veľkosti korpusu. Z obrázku jasne vidieť, že už pri lexikóne o veľkosti 2000 slov je počet rozpoznaných slov v texte viac než 80%. Hodnota 5000 bola zvolená preto, aby sa do lexikónu dostalo viac slov a bolo možné vytvoriť viac asociačných spojení.

5.1.4 TokenStream a TokenWindow

TokenStream je trieda, ktorou je kódovaný vstup do učenia systému. Jedná sa o prúd tokenov vyextrahovaných z textového dokumentu, akúsi pásku, ktorá je čítaná čítacou hlavou o šírke piatich tokenov. Kým hlava stojí nehybne, páska sa môže posunúť o jedno políčko doľava. Proces posunutia pásky smerom doľava nazývame „posunom?? okna“, pretože je analogický posunu čítacej hlavy smerom doprava. TokenWindow je okno piatich usporiadaných tokenov prečítaných čítacou hlavičkou, poprípade okno piatich usporiadaných tokenov získaných inou cestou.



Obr. 6 prevod dokumentu na tokenStream

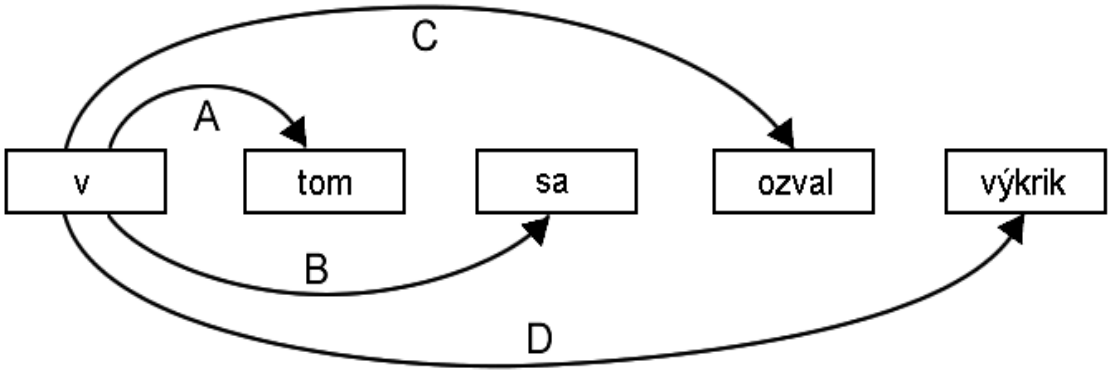


Obr. 7 spracovanie tokenStreamu na okno indexov

5.1.5 Fascikle

Fascikel je základný objekt obsahujúci znalosti systému. Prakticky je fascikel realizovaný ako matica SIZE x SIZE tokenov, obsahujúca počet spoločných výskytov týchto dvojíc tokenov, vo vzdialenosti danej týmto fasciklom, kde SIZE je veľkosť

slovníka. Znalosti sa uchovávajú v štyroch dopredných a štyroch inverzných fascikloch, označených veľkými písmenami (dopredné) A, B, C, D, (a inverzné) Ai, Bi, Ci a Di. Písmená označujú vzdialenosť medzi tokenmi, ktorých spoločné výskyty si fascikel uchováva a to tak, že A značí vzdialenosť jedného tokenu, B dvoch atď. (Obr.8). Všetky fascikle sú uložené v tzv. **regióne**, ktorý tvorí základnú triedu určenú na uchovávanie znalostí.



Obr. 8 znázornenie fasciklov

V tejto konkrétnej implementácii je pre odľahčenie pamäťovej náročnosti a zrýchlenie výkonu prehľadávania vyriešená perzistencia znalostí cez hashovacie tabuľky.

5.1.6 Cortex

Lexikón a región (pokrývajúci všetky fascikle) sú uložené v cortexe. Cortex tvorí základnú funkčnú časť implementácie, poskytuje základné funkcie na ovládanie celého systému.

Kým pri základných vstupoch a výstupoch sa na reprezentáciu hodnôt tokenov používajú reťazce, cortex pracuje len s toknemi, ktoré reprezentujú vnútornú hodnotu celými číslami, indexmi z lexikónu. Tento prístup bol použitý ako na jednoznačné rozlíšenie tokenov, tak na jednoduchší prístup k tokenom.

Základné funkcie cortexu sa dajú rozdeliť do dvoch kategórií. Funkcie pracujúce na nižšej úrovni slúžiace na elementárne operácie s množinami a oknom a na funkcie na vyššej úrovni, ktoré ich využívajú, poskytujúce používateľovi odpovede na jeho otázky.:

Funkcie na nižšej úrovni:

- PresentLearningWindow – základná funkcia pri učení. Zistí, či je dané okno prijateľné a v pozitívnom prípade uloží znalosti z okna do regiónu.
- HasAssociation – testovanie existencie danej asociácie.
- GetConsensusIndex – základná funkcia na testovanie konsenzu.
- LexiconizeWindow – indexácia okna – prevod okna pozostávajúceho z reťazcov na okno ich indexov v lexikóne.
- DelexiconizeWindow – prevod indexov na ich reťazce uložené v lexikóne.
- HoneALL – spojí množiny excitovaných tokenov pomocou metódy min-max.
- Serialize – uloží všetky dáta cortexu do súboru.
- Deserialize – načíta všetky dáta cortexu zo súboru.

Funkcie na vyššej úrovni:

- GetAnswers – doplní posledný token zadaného okna tokenov tak, aby sa splnila podmienka konsenzu.
- GetSynonyms – vráti zoznam tokenov so špecifickými vlastnosťami vzhľadom na zadaný token.
- GetConsensusALLlist – nahradí znaky "#" v zadanom okne tokenov tak, aby spĺňali podmienku konsenzu.
- GetSimilarity – funkcia slúžiaca na kvantitatívne vyjadrenie podobnosti dvoch tokenov na základe ich asociatívneho okolia.

5.2 Proces učenia

Proces učenia je procesom uchovávania znalostí do systému. V tejto implementácii sa jedná iba o štatistické počty nad textom. Vstupom do tohto procesu je jeden alebo viac textových dokumentov vo formáte „plain/text". Ten je pomocou balíka LuceneTokenizer od Apachu prevedený na tokenStream (obr. 6).

1. vyber prvé okno z tokenStreamu,

2. otestuj prípustnosť okna,
3. ak je okno neprípustné, posuň okno, pokračuj krokom 2,
4. pridaj spoločné výskyty tokenov cez príslušné fascikle do regiónu,
5. ak tokenStream pokračuje, posuň okno a pokračuj krokom 2.

TokenStream je ďalej prevedený na okno piatich tokenov, ktoré je neskôr oindexované za pomoci lexikónu na okno piatich indexov. To vstupuje do metódy kortexu PresentLearningWindow.

5.2.1 PresentLearningWindow

Funkcia kontroluje prípustnosť okna a pokiaľ je okno prípustné pridá spoločné výskyty tokenov cez príslušné fascikle do regiónu.

Okno je prípustné práve vtedy, keď sa jeho posledný token nerovná znaku "#", to znamená, má svoj index (odlišný od indexu NULL_TOKENU) v lexikóne.

Učenie – pridávanie spoločných výskytov do regiónu, potom prebieha zprava doľava po najbližší neznámy (NULL_TOKEN) token, alebo v prípade všetkých známych tokenov po začiatok okna.

Príklady prípustných a neprípustných okien:

- a b c d e : okno je prípustné, učíme teda asociácie (a e), (b e), (c e), (d e)
- N a b c d : prípustné je pod-okno je a b c d, učíme (a d) (b d) (c d)
- N N b d N : okno nie je prípustné, posledný token je NULL_TOKEN.
- a N N a b : prípustné pod-okno je (a b) učíme (a b)
- N N a b c : prípustné pod-okno (a b c), učíme (a c) (b c)

5.3 Konsenzus

Konsenzus je základným nástrojom na rozpoznanie frázy. Fráza sa tu chápe ako "syntakticky" správny vstup, ktorý systém rozpozná. Je treba vyzdvihnúť prednosť

systému, ktorý dokáže rozpoznať aj „nové“ frázy, to znamená také, ktoré neboli explicitne zadané v trénovacej množine, práve vďaka tejto funkcii.

5.3.1 HoneAll

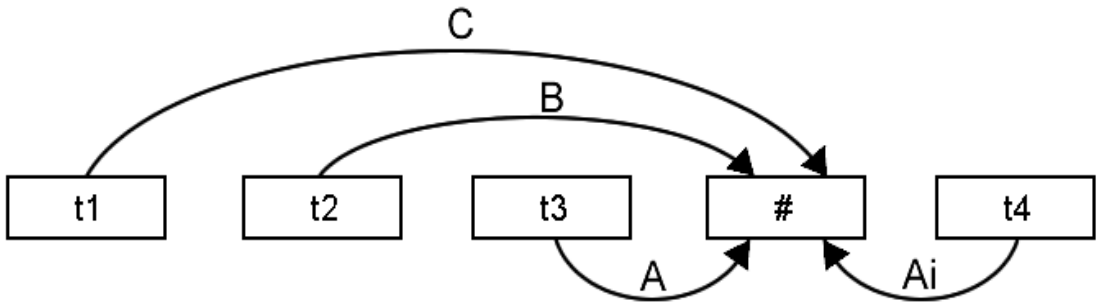
Metóda honeAll je najdôležitejšou funkciou z funkcií nižšej úrovne cortexu. Je to metóda určená na spájanie množín excitovaných tokenov, v podstate sa jedná o rozšírenie operácie prieniku množín. Ak je vykonávaná na okne, tak sa jedná o hľadanie takej odpovede, ktorá čo najviac potvrdzuje najslabšiu hypotézu. Vstupom je ľubovoľný počet množín excitovaných tokenov.

Popis algoritmu:

1. Pomocou funkcie unexciteTokens vytvoríme z množín vstupujúcich do algoritmu množiny obyčajných (neexcitovaných) tokenov.
2. Do množiny Mu-result pridáme zjednotenie týchto množín.
3. Do množiny Mp-result pridáme prienik týchto množín.
4. Každému prvku z množiny Ms-result priradíme váhu, ktorá je určená minimálnou váhou z Mu-result pre excitované tokeny s hodnotou rovnajúcou sa hodnote daného prvku - tokenu. Novovytvorený excitovaný token (token s priradenou váhou) pridáme do množiny result.
5. V množine result sa nachádza výsledok algoritmu.

5.3.2 Tvorba konsenzu

Máme okno piatich blokov, ktoré môžu obsahovať len známe (nachádzajúce sa v slovníku - lexikóne) a nullové tokeny. Tokeny sú navzvájom prepojené asociačnými reláciami cez 4 dopredné?? a 4 inverzné fascikle. Pri tvorbe konsensu chýba v okne jeden alebo viac tokenov, ktoré označíme znakom # a pokladáme za otázku. Každý blok okna, ktorý obsahuje už špecifiký token hlasuje cez príslušný fascikel (daný vzdialenosťou tokenu od otázky v okne) za odpoveď. Výsledkom hlasovania je usporiadaná množina vybudených tokenov pre každý nenullový token. Odpoveďou na otázku je potom zjednotenie týchto množín metódou max, min pomocou funkcie HoneAll v cortexe.



Obr. 9 znázornenie hlasovania fasciklov pri tvorbe "consensu"

V okne sa nachádzajú 4 známe tokeny t1, t2, t3 a t4. Otázka je umiestnená vo štvrtom bloku okna a je označená znakom "#". Na obr. 9 je znázornené aj prepojenie známych tokenov na otázku cez fascikle A,Ai,B a C.

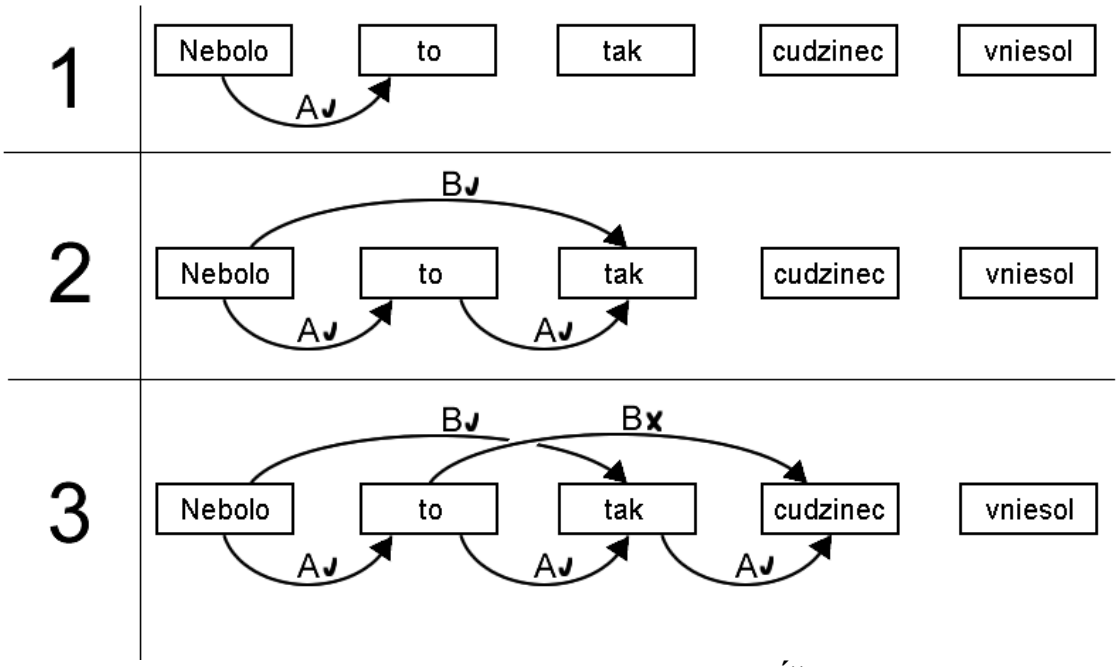
Výsledkom hlasovania známych tonekov cez príslušné fascikle sú usporiadané množiny T1 (výsledok hlasovania tokenu t1 cez fascikel C), T2, T3 a T4. Odpoveďou na otázku konsenzu je potom zjednotenie týchto množín pomocou metódy honeAll.

5.3.3 Kontrola konsenzu

Kontrola konsenzu si vyžaduje jedine znalosť asociácií cez príslušné fascikle v rámci dopytovacieho okna. Postupne sa kontroluje existencia daných asociácií od prvého tokena v okne k tokenu druhému, tretiemu, atď. Výstupom funkcie kontroly konsenzu je dĺžka nájdenej frázy v okne. Tá popisuje najkratší možný úsek okna braný zľava doprava, ktorý je úplne asociačne prepojený.

Popis algoritmu:

1. Vyber prvý token v okne, t1, i = 2.
2. Vyber i-tý token v okne ti.
3. Otestuj existenciu asociácií cez všetky prístupné fascikle medzi t1 a ti.
4. Ak je porušená podmienka existencie asociácie cez niektorý z fasciklov, vráť hodnotu i-1 ako výsledok a ukonči algoritmus.
5. Ak je podmienka existencie všetkých asociácií splnená, tak i = i + 1.
6. Ak i <5 pokračuj na krok 2, inak vráť hodnotu i ako výsledok a ukonči algoritmus.



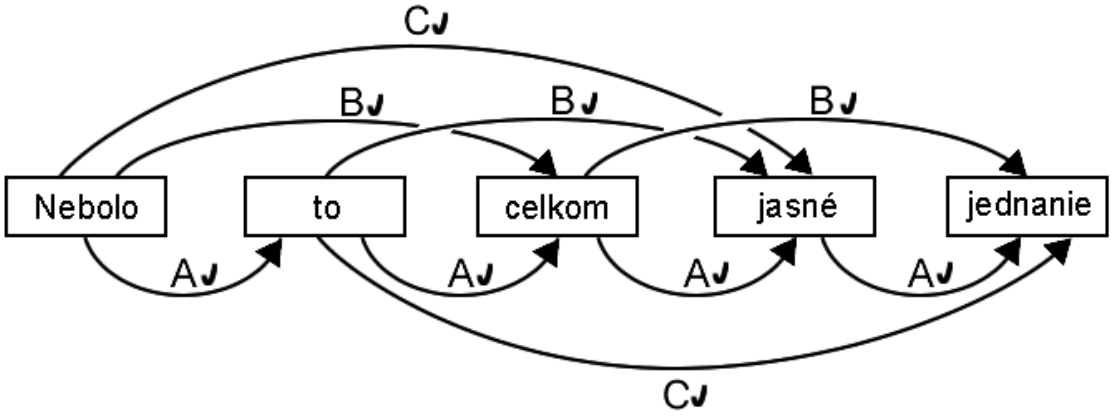
Obr. 10 znázornenie kontroly konsenzu pri okne nespĺňajúcom podmienky konsenzu

Na obr. 10 je znázornený proces kontroly konsenzu na okne tokenov: (Nebolo, to, tak, cudzinec, vniesol). Kroky na obrázku sa líšia od krokov algoritmu.

1. Vybraný prvý token “Nebolo”, i = 2, čiže ti = t2 = “to”. Medzi tokenmi sa nenachádzajú žiadne iné tokeny a existuje medzi nimi jediný druh asociačného prepojenia, a to cez fascikel A. Keďže medzi nimi existuje cez tento fascikel asociácia, algoritmus pokračuje ďalej s i = i + 1 = 3.

2. V tomto prípade je ti = t3 = “tak”. Medzi tokenmi “Nebolo” a “tak” sa nachádza token “to”. Na prešetrenie všetkých asociácii je teda potrebné prešetriť existenciu troch asociácií. Keďže asociácie sú vytvorené, algoritmus pokračuje ďalej.

3. Algoritmus narazil na absenciu asociácie medzi tokenmi “to” a “cudzinec” cez fascikel B. Nie je teda splnená podmienka asociácie a algoritmus nám vráti hodnotu 3 ako výslednú dĺžku frázy v okne.



Obr. 11 znázornenie kontroly konsenzu pri okne spĺňajúcom podmienky konsenzu

Kontrola konsenzu je potrebná ako podporná funkcia pri tvorbe synoným a rozpoznávania fráz v texte.

5.4 Ropoznávanie fráz

Rozpoznávanie fráz je založené na kontrole konsenzu. Text, v ktorom frázy hľadáme, vstupuje ako parameter do funkcie rozpoznávania fráz v čistej "plain/text" podobe a na výstupe dostaneme HTML dokument s vyznačenými frázami.

Spracovanie vstupu je veľmi podobné samotnému učeniu systému, avšak namiesto učenia jednotlivých znalostí získaných zo vstupných okien vstupujú tieto okná do funkcie kontroly konsensu. Algoritmus:

1. vyber okno z tokenStreamu,
2. otestuj konsenzus,
3. ak je dĺžka konsenzu kratšia ako 5, označ frázu a presuň okno na koniec konsenzu,
4. inak posuň okno o jedno políčko,
5. ak textStream neskončil, skoč na krok 2.

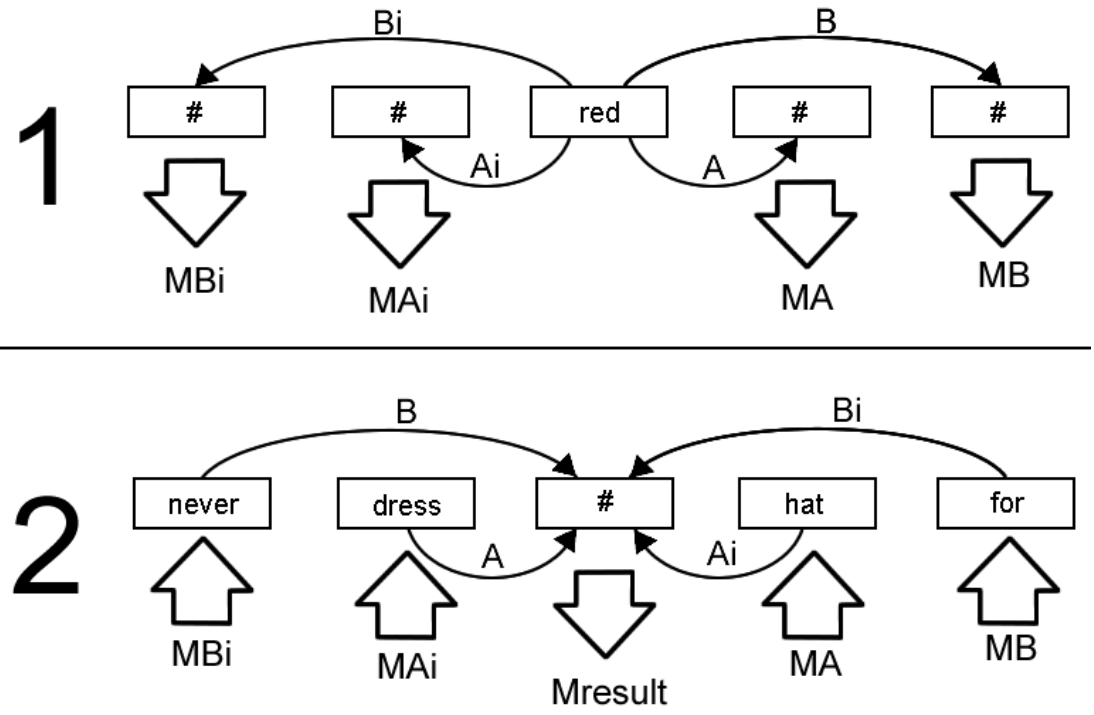
Výstupný HTML súbor obsahuje aj štatistickú informáciu – percentuálny podiel počtu všetkých rozpoznaných slov k počtu slov v označených frázach.

5.5 Tvorba synoným

Funkcia cortexu GetSynonyms je jednou z najzaujímavejších. Keďže v jazyku sú synonymá chápané ako množina slov, ktoré inak znejú, ale majú rovnaký význam., v tomto prípade sa nejedná o synonymá v pravom slova zmysle. Tento pojem bol použitý len kvôli najbližšej nájdenej aproximácii k už existujúcim pojmom. V ponímaní tejto implementácie sa jedná o tokeny - slová patriace do jednej triedy. Vstupom do funkcie je token. Výstupom je množina tokenov, ktoré sú tomuto tokenu syntakticky blízke, rozšírená o tokeny z jedného sémantického poľa, čo implicitne danú triedu definuje. V ďalšom texte je teda pod pojmom synonymum chápaná príslušnosť k triede.

Algoritmus pre získanie množiny synoným (popis funkcie GetSynonyms):

1. Pre zadaný token T vyber všetky asociácie a fascikla A, Ai, B, Bi a ulož ich do množín MA, MAi, MB a MBi.
2. Nájdi konsenzus pre všetky kombinácie MBi x MAi x T x MA x MB. Výsledky konsenzov ulož do výslednej množiny M.
3. Vráť množinu M ako výsledok algoritmu.



Obr. 12 hľadanie synoným pre token „red“

Na obr. 12 je zobrazený príklad hľadania množiny synoným pre token „red“. Kroky nezodpovedajú krokom algoritmu. Popis krokov na obrázku:

1. Cez príslušné fascikle sa naplnia množiny MBi, MBi, Ma a Mb.
2. Postupne sa pridávajú do Mresult množiny slov získaných funkciu GetConsensus z okna tvoreného všetkými kombináciami MBi x MAi x „#” x MA x MB. V tomto konkrétnom príklade sa do Mresult pridá výsledok konsezu pre okno („never“, „dress“, „#“, „hat“, „for“), čo môže byť napríklad slovo „yellow“.
3. V množine Mresult sa nachádza výsledná množina synoným.

Uvádzam príklady synoným, naučených na 200 MB korpuse. Korpus pochádza z elektronickej knižnice www.gutenberg.org a pozostáva z náhodne vybraných publikácií v anglickom jazyku pri rôznych prahoch. Vľavo sa nachádza token, ktorý bol vstupom funkcie GetSynonyms, vpravo za dvojbodkou je výstup danej funkcie.

Threshold 5: **animal**: beast, animal, deer, song
Threshold 5.5: **love**: love, own, friendship
Threshold 6: **boat**: canoe, vessel, boats, ships, boat, ship
Threshold 7: **music**: voices, singing, sing, cheerful, voice, song, music
king: king, emperor
palace: palace, temple
Threshold 8: **eye**: vision, skin, eye, eyes
sun: moon, stars, sun, star, fires, fire, splendour, blazing, slowly, color, bright
Threshold 9: **war**: alliance, war, estabilished
moon: sun, moon
Threshold 10: **grass**: corn, tobacco, grass, tall
warrior: partha, warrior, warriors, duryodhana, bhima, arjuna, bhishma, hero
Threshold 11: **bow**: mace, shafts, weapons, arrows, bow, cars
street: street, road
Threshold 18: **red**: gray, grey, red, white, brown, blue, green
Threshold 20: **krishna**: arjuna, satyaki, bhimasena, krishna, duryodhana, yudhishthira

Príslušnosť do triedy pekne znázorňuje napríklad posledný príklad s tresholdom 20, ktorý vyjadruje veľmi silné asociácie. Pri otázke tokenom „Krishna“, nám poskytne funkcia popis triedy, ktorú by sme mohli nazvať „indické božstvá“, vymenovaním jej prvkov. Zo syntaktického hľadiska sa jedná o podstatné mená a zo sémantického hľadiska sa jedná o mená indických bohov. Tento exotický príklad sa dostal do cortexu vďaka textovému dokumentu Máhabharata, ktorý sa nachádza v množine dokumentov, z ktorej sa systém učil.

5.6 Matica podobností

Pri tvorbe matice podobnosti sa využíva funkcia cortexu GetSimilarity. Jej vstupom sú dva tokeny a index fasciklu, ktorý má byť použitý pri prehľadávaní asociatívneho okolia.

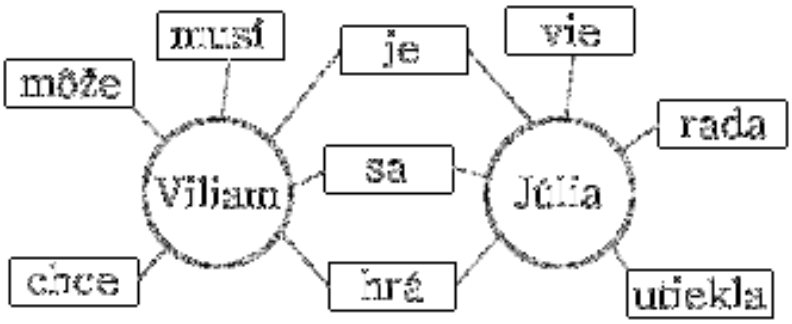
GetSimilarity

Funkcia GetSimilarity slúži na kvantitatívne vyhodnotenie príbuznosti (podobnosti) dvoch tokenov na základe ich asociatívneho okolia cez príslušný fascikel. Vyjadruje pomer počtu spoločných asociácií k počtu všetkých asociácií daných tokenov. Jedná sa o sémantickú podobnosť pojmov definovanú vyššie.

Príklad

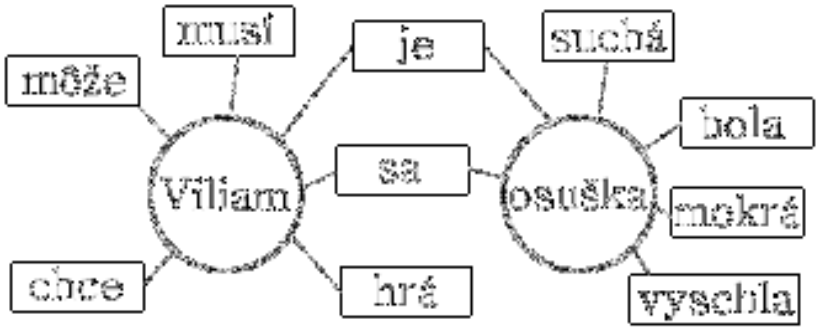
Majme tokeny „Viliam“, „Júlia“ a „osuška“ také, že poznáme ich asociatívne okolie. Chceme vypočítať sémantickú podobnosť tokenu „Viliam“ k ostatným tokenom. Predpokladajme, že asociácie vyzerajú takto:

- Viliam => môže, musí, chce, je, sa, hrá
- Júlia => vie, rada, utiekla, je, sa hrá
- Osuška => je, sa, suchá, bola, mokrá, vyschla



Obr. 13 zobrazenie asociačného okolia pre tokeny „Viliam“ a „Júlia“

Počet spoločných asociácií tokenov Viliam a Júlia je 3, počet všetkých rôznych tokenov asociovaných s týmito tokenmi je 9. Kvantitatívne vyjadrenie ich sémantickej podobnosti je teda 3 / 9 = 0,333.



Obr. 14 zobrazenie asociačného okolia pre tokeny „Viliam“ a „osuška“

Počet spoločných asociácií tokenov Viliam a Júlia je 2, počet všetkých rôznych tokenov asociovaných s týmito tokenmi je 10. Kvantitatívne vyjadrenie ich sémantickej podobnosti je teda 2 / 10 = 0,2.

Matica podobností by potom vyzerala takto:

	Viliam	Júlia	osuška
Viliam	1	0.333	0.2
Júlia	0.333	1	0.2
osuška	0.2	0.2	1

Obr. 15 Matica podobností

Z čoho je možné vyvodiť záver, že pojem „Viliam“ má vzhľadom na teóriu asociatívneho učenia pojmov bližšie k pojmu „Júlia“ ako k pojmu „osuška“.

V modeli Beast sa používa pri počítaní podobnosti dvoch pojmov váhovaný súčet podobností cez všetky dvojice fasciklov, kde váha klesá priamo úmerne so vzdialenosťou fascikla. Matica sa generuje príkazom learnRelatives v modeli Beast a uloží maticu podobnosti aj so zonamom tokenov do súboru „relationSet.dat“.

6 Experimenty

6.1 Experimenty na umelej gramatike

Nad danou gramatikou vytvorenou pomocou programu SLG (The Simple Language Generator 2.1) som vygeneroval 10000 viet o celkovej veľkosti 342 kB. Pomocou modulu BeastLearn som vytvoril nad týmto súborom serializovanú triedu Cortex spolu s lexikónom, pričom som zaznamenával rast počtu asociácií každých 100 tokenov.

6.1.1 Popis gramatiky

```
S  : SP VI . (.25) | SP VT OP . |
 {sub-intr, SP NP N, VI} | {sub-trns, SP NP N, VT} |
 {trns-obj, VT, OP NP N} | {sub-obj, SP NP N, OP NP N} |
 {intrans-ref, VI, SP RC VI};

SP | OP : NP | NP RC (.3) |
 {sub-intr, NP N, RC VI} | {sub-trns, NP N, RC VT} |
 {trns-obj, RC VT2, NP N};

RC : who VI | who VT OP | who SP VT2 | {trns-obj, VT, OP NP N} |
{sub-trns, SP NP N, VT2};

NP : ART ADJ N | {noun-art, N, ART} | {noun-adj, N, ADJ};

ART: "" | the | a;

ADJ: "" (0.6) | quick | happy | hungry | nasty | mangy | crazy |
sleazy;

N  : boy | boys | girl | girls | Mary | John | cat | cats | dog |
dogs;

VI : walks | walk | bites | bite | eats | eat | barks | bark;

VT | VT2 : chases | chase | feeds | feed | sees | see | walks |
walk | bites | bite;

sub-intr {
 boy  | girl  | Mary  | John  : walks | eats;
 boys | girls : walk  | eat;
 cat  | dog   : walks | bites | eats  | barks;
 cats | dogs  : walk  | bite  | eat   | bark;
 cat  | cats  ! bark  | barks;
}

sub-trns {
 boy  | girl  | Mary   | John : chases | feeds | sees(.1) |
walks;
```

```
 boys | girls : chase  | feed     | see(.1) | walk;
 cat  | dog   : chases | sees(.2) | bites;
 cats | dogs  : chase  | see(.2)  | bite;
}

trns-obj {
 walk | walks : cat | cats | dog | dogs;
 see  | sees  : cat | cats;
}

sub-obj {
 Mary ! Mary;
 John ! John;
}

intrans-ref {
 walks | walk ! walks | walk;
 bites | bite ! bites | bite;
 eats  | eat  ! eats  | eat;
 barks | bark ! barks | bark;
}

noun-adj {
 boy  | boys | girl | girls | Mary | John ! mangy;
 John | cat  | cats | dog   | dogs ! sleazy;
}

noun-art {
 Mary | John : "";
 boys | girls | cats | dogs ! a;
 boy  | girl  | cat  | dog  ! "";
}
```

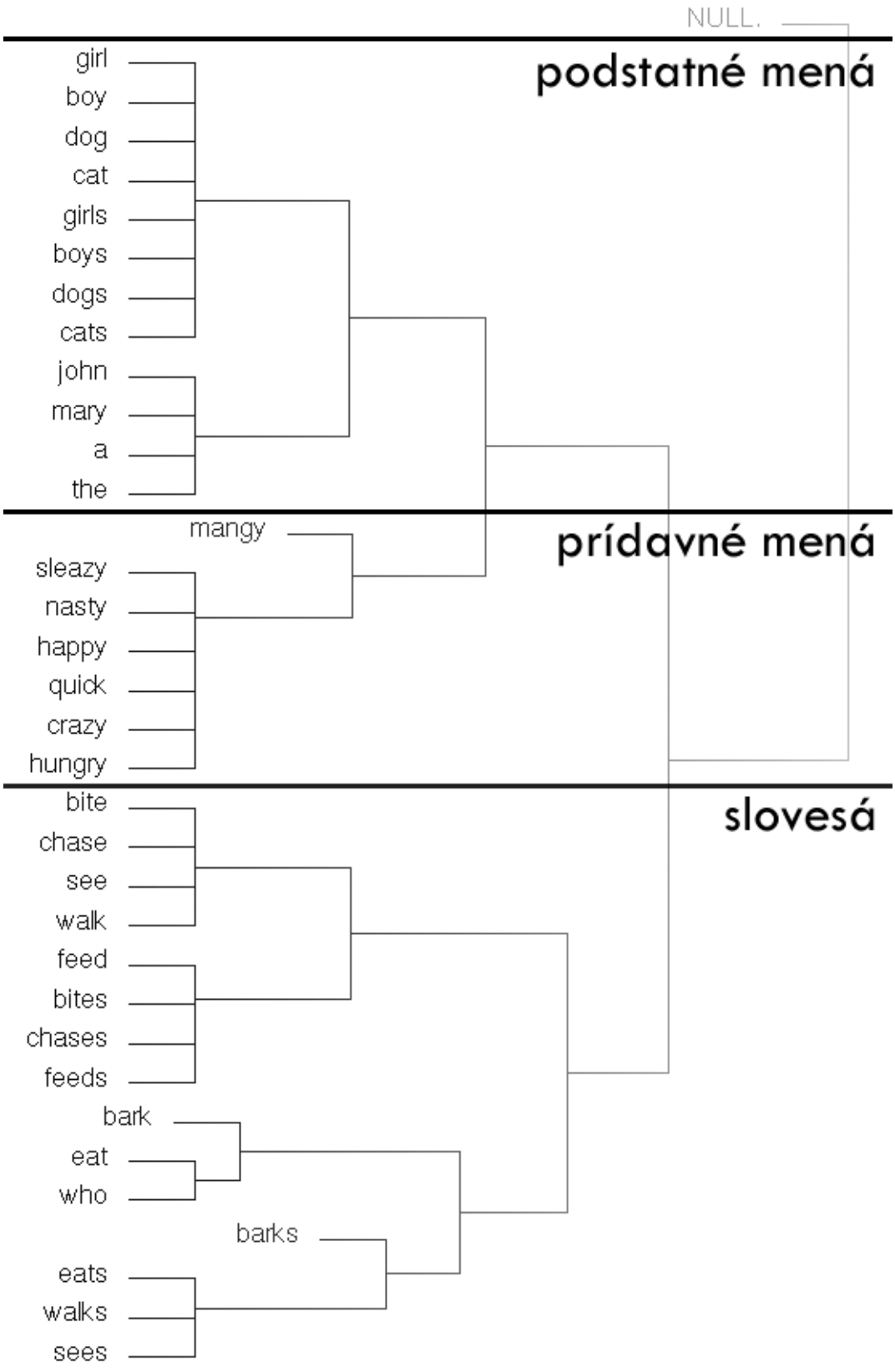
6.1.2 Príklad trénovacej množiny

```
John walks.
mangy dogs bite.
a mangy cat walks.
the nasty dog bites the girls who walk.
the boy who eats walks.
the boy who walks hungry dogs who chase John walks a cat who
eats.
the dogs who Mary walks bark.
a crazy cat chases John.
girls walk hungry dogs who chase a cat.
John who feeds Mary who cats who the cats see bite feeds dogs.
quick dogs bite happy cats.
the mangy dogs eat.
a girl who walks the hungry dog who sees the cat walks the dogs.
the cats who chase John who feeds mangy cats bite happy Mary who
chases John.
```

Pomocou modulu Beast som vygeneroval maticu podobnosti. Tú som pozhlukoval algoritmom hclust() v systéme R, voľne prístupnom jazyku a prostredí pre štatistické počty.

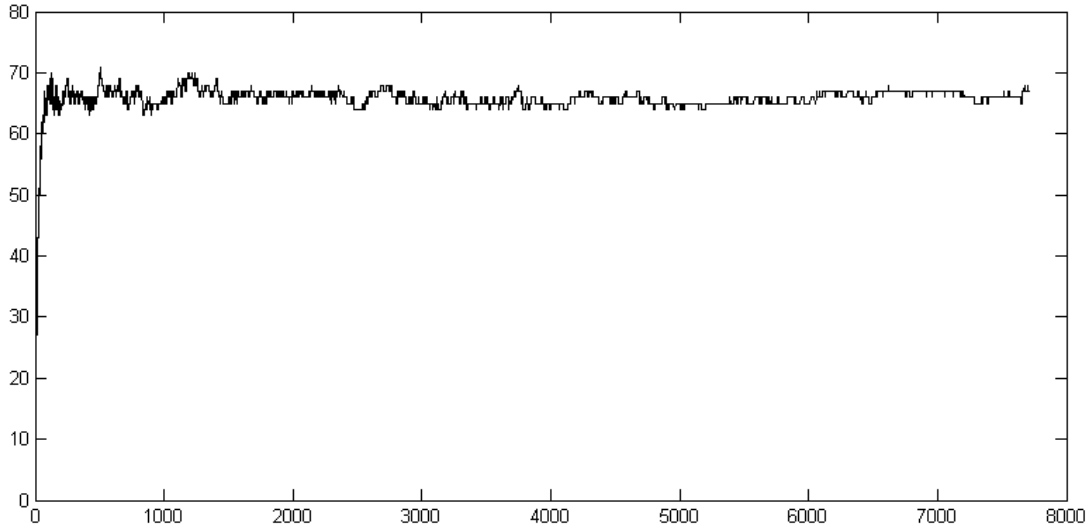
6.1.3 Algoritmus hclust(), hierarchické zhlukovanie programu R

Funkcia slúži na hierarické zhlukovanie pomocou matice odlišností pre rôzny počet objektov. Na začiatku je každý objekt priradený svojmu vlastnému zhluku, potom algoritmus iteratívne, v každom kroku spája dva najpodobnejšie zhluky, pokiaľ ich nespojí do jediného. V každom kroku sa vzdialenosti (podobnosti) zhlukov objektov prepočítavajú pomocou Lance – Williamsovej rovnice na aktualizáciu odlišnosti v závislosti od použitej metódy, v tomto konkrétnom prípade úplnej metódy, ktorá hľadá najpodobnejšie zhluky..



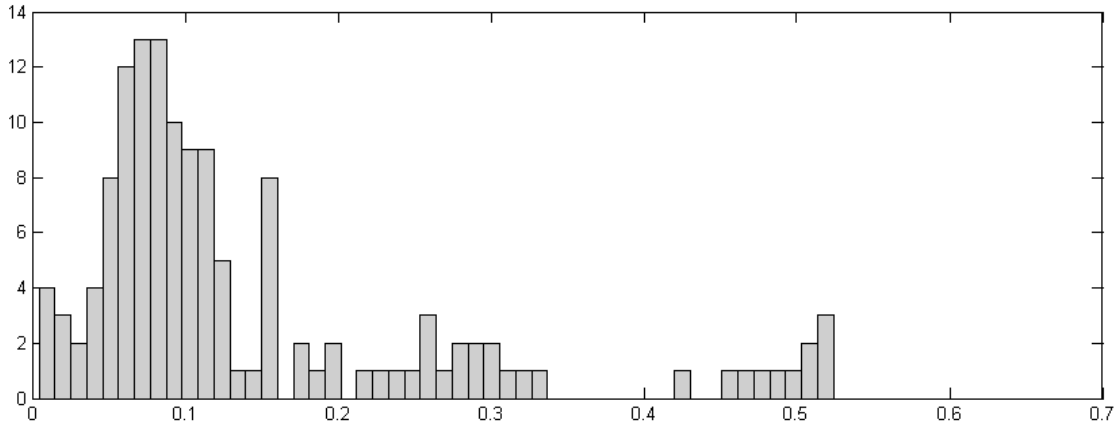
Obr. 16 Pozhlukovaná matica podobnosti (umelá gramatika)

Z dendogramu na obr. 16 vidno, že systém pozhlukoval tokeny do morfologických skupín so skoro 100% úspešnosťou. Výnimku tvoria tokeny „a“ a „the“, ktoré sú v texte veľmi časté a vytvárajú si mnoho asociácií. Ďalšou odchylkou od 100% úspešnosti je token „who“, ktorý nepatrí k slovesám a bol zaradený do zlého zhluku najskôr kvôli definícii gramatiky, ktorá tiež nepopisuje pravidlá anglického jazyka úplne. Token NULL tvorí osobitný zhluk, pretože sa so žiadnymi tokenmi nespája.



Obr. 17 Priebeh rastu asociácií počas učenia pri prahu 3

Priebeh rastu asociácií počas učenia je dobrým ukazovateľom toho, nakoľko je systém danú „gramatiku“ naučený. Kým na začiatku učenia prudko rastie a zmena počtu asociácií je veľká, po prejdení väčšej časti korpusu sa počet asociácií stabilizuje, ich zmena je relatívne malá.



Obr. 18 Histogram distribúcie váh pri prahu 3 po naučení vo fascikli A

6.1.4 Rozpoznávanie fráz

Po vložení korpusu, z ktorého sa systém učil do funkcie rozpoznávania fráz, dokázal systém rozpoznať väčšinu textu. Zo všetkých slov v korpuse sa nachádzalo 80.6746% vo frázach.

6.2 Experimenty na korpuse z prirodzeného jazyka

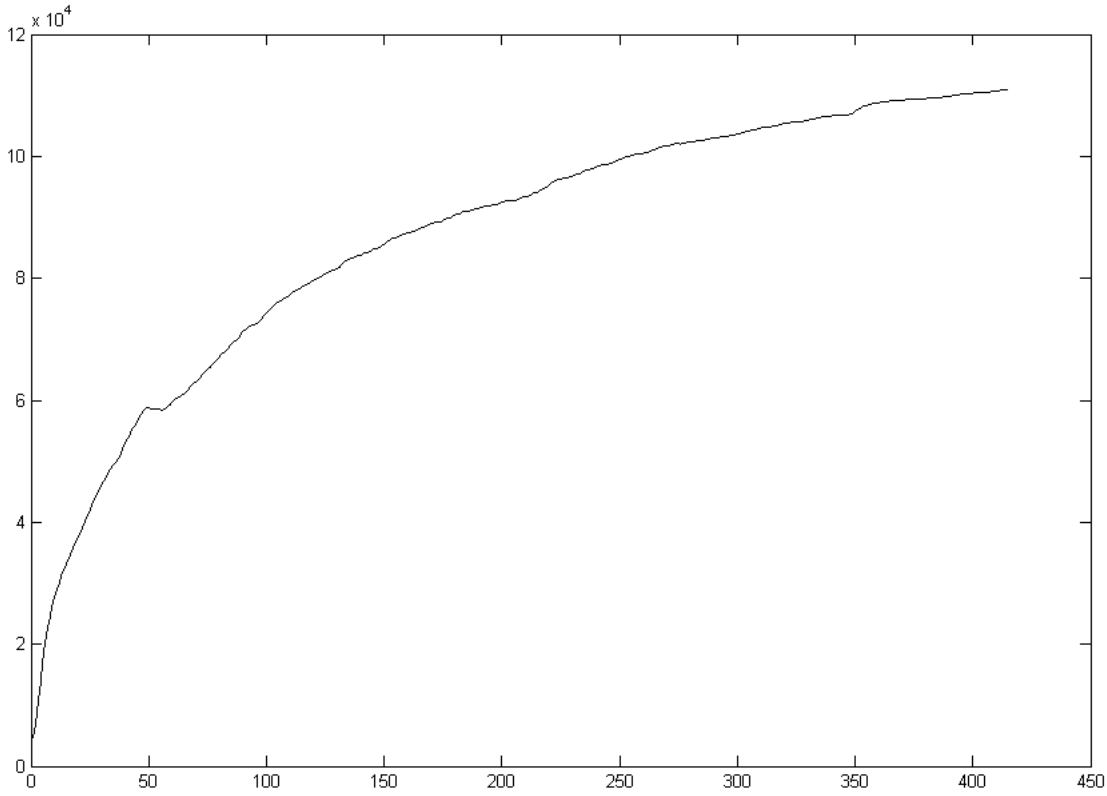
Korpus zložený z textov v prirodzenom jazyku tvorí množina náhodne vybraných anglických publikácií z elektronickej knižnice www.gutenberg.org . Celková veľkosť korpusu je 200 mB. Pomocou modulu BeastLearn som vytvoril nad týmto súborom serializovanú triedu Cortex spolu s lexikónom, pričom som zaznamenával rast počtu asociácií každých 100 000 tokenov (z celkového počtu 42 000 000 tokenov).

6.2.1 Príklad textu z korpusu

.....But at the foot of this cliff grew a tree, gnarled and stunted, the which, as Beltane watched, Black Roger began to climb, until, being some ten feet from the ground, he, reaching out and seizing a thick vine that grew upon the rock, stepped from the tree and vanished into the face of the cliff. But in a moment the leaves were parted and Roger looked forth, beckoning Beltane to follow. So, having climbed the tree, Beltane in turn seized hold upon the vine, and stumbling amid the leaves, found himself on his knees within a small cave, where Roger's hand met his....

6.2.2 Priebeh učenia

Učenie som realizoval s lexikónom o veľkosti 3500 tokenov. K tomuto kompromisu som bol nútený prístupiť po absencii výpočtovej sily, ktorú som mal k dispozícii. Z grafu na obr. 5 ale vyplýva, že 3500 tokenov pokrýva až vyše 85% celkového počtu roznych slov korpusu v prirodzenom anglickom jazyku. Kritickejšia bola veľkosť korpusu, 200 mB, ktorá nepostačuje na dokonalé naučenie asociácií v prirodzenom anglickom jazyku.



Obr. 19 Priebeh rastu asociácií počas učenia pri prahu 8

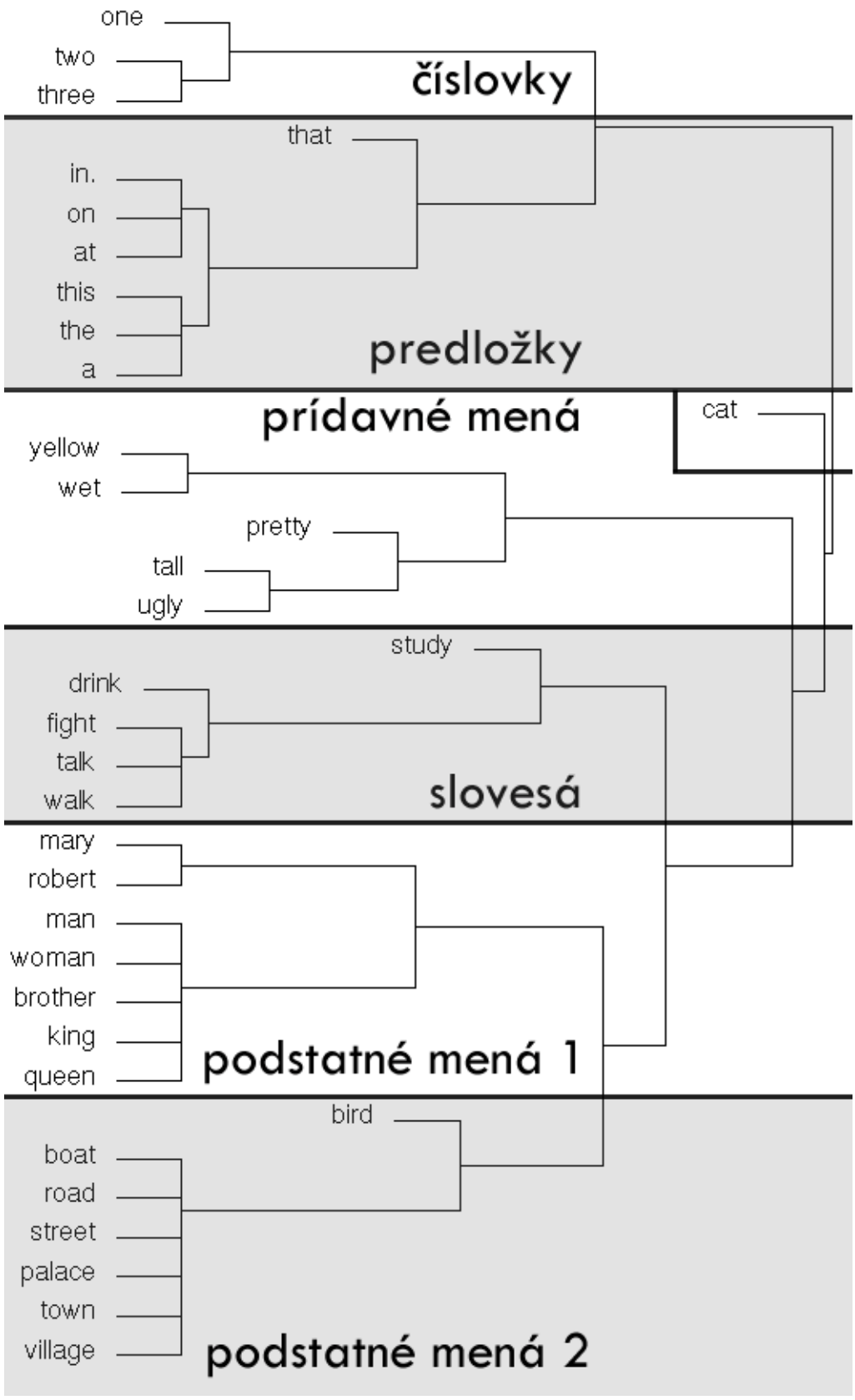
Z obr. 19 znázorňujúceho priebeh nárastu počtu asociácií počas učenia vidno, že pri skončení učenia nebol počet asociácií stále stabilizovaný, rástol. Nebolo však možné kvôli pamäťovej náročnosti, vzhľadom na podmienky naučiť systém na väčšom korpuse.

Obr. 20 Histogram distribúcie váh pri prahu 8 po naučení vo fascikli A

6.2.3 Matica podobnosti

Vzhľadom na toľkokrát spomínanú pamäťovú a výpočtovú náročnosť bolo vytvorenie matice podobnosti pre všetkých 3500 tokenov v slovníku nereálne. Jednou z možností ako odprezentovať hierchachické zhlukovanie na danom korpuse je vytvorenie matice z vybraných tokenov.

Množina tokenov, z ktorých je matica podobnosti vytvorená je vybraná tak, aby sa dalo dopredu predpokladať vytvorenie zhlukov, ktoré sa od seba budú významnejšie odlišovať.



Obr. 21 Pozhlukovaná matica podobnosti (prirodzený jazyk)

Výsledný graf, vytvorený rovnako algoritmom hclust, ukazuje, že množina zadaných slov sa rozdelila do siedmich významnejších zhlukov. Jediným nesprávne zaradeným slovom, je „cat“, ktoré sa nachádza vo svojom vlastnom zhluku.

Číslovky a predložky spolu s členmi sú členmi jedného zhluku.

Podstatné mená sa rozdeľujú na dva zhluky, z ktorých prvý obsahuje len slová označujúce osoby, a druhý obsahuje ostatné podstatné mená.

6.2.4 Synonymá

Jedným z možných spôsobov ilustrácie zhlukovania pojmov sú synonymá. Predpokladáme teda, že odpoveďou na otázky, budú tokeny, ktoré budú mať jasnú morfologickú štruktúru a navyše budú inštanciami jednej sémantickej triedy. Keďže kvalita tejto triedy je silne závislá od prahu, nejedná sa o experiment v pravom slova zmysle, ale iba o ukážku systému, o ktorej možno trdiť, že nemôže byť náhodná.

Pri opýtaní sa slovom „**five**“ je výslednou množinou 29 slov, ktoré popisujú triedu a zároveň morfologickú skupinu „číslovky“ na 86.206%.

five: thirty, fifty, twenty, forty, eighty, fifteen, fourteen, sixteen, sixty, six, five, seventy, eight, ten, three, four, dollars, seven, nine, twelve, thousand, eleven, half, eighteen, thirteen, per, two, several, cost.

Pri opýtaní sa slovom „**robert**“ je výslednou množinou 14 slov, 92,857% z nich tvoria podstatné mená, 85,714% z nich patrí do triedy „krstné mená“.

robert: francis, william, peter, walter, richard, george, james, henry, anthony, edward, john, iii, sir, charles.

Pri opýtaní sa slovom „**green**“ je výslednou množinou 21 slov, z toho 85,751% z nich tvoria prídavné mená, a z toho 52,38% z nich patrí do triedy „farba“.

green: yellow, green, red, blue, black, white , gray, brown, grey, cotton, tall, sky, thick, corn, cultivated, flat, grown, wild, golden, plain, dark.

Treba podotknúť, že výsledné tokeny sú usporiadané podľa minimálnej váhy, s akou sa spájajú vo frázach tvorených kombináciou slov asociačného okolia pýtaného tokenu. Vhodným orezaním je tak možné dosiahnuť oveľa lepšie výsledky. Napríklad stanovenie prahu na váhu s akou bol pri dotaze „green“ vrátený token „grey“ a následným orezaním podľa tohto prahu by sme dostali množinu tokenov, ktoré by patrili do triedy farba so 100% úspešnosťou.

Tieto výsledky sú viac ilustračné ako smerodajné, neposkytujú žiaden dókaz, môžu byť však dobrou motiváciou (rovnako ako príklady synoným z predchádzajúcej kapitoly) pre ďašiu optimalizáciu, testovanie a rozširovanie systému.

6.2.5 Rozpoznávanie fráz

Po vložení 10 mB podvzorky korpusu, z ktorého sa systém učil do funkcie rozpoznávania fráz, dokázal systém rozpoznať len veľmi malú časť textu. Zo všetkých slov v korpuse sa nachádzalo vo frázach len 6,807%. Tento slabý výsledok mohol byť spôsobený buď nedostatočnou veľkosťou lexikónu, keďže každé neznáme slovo mohlo veľmi ľahko „rozbiť“ frázu, tak aj malým korpusom, ktorý vstupoval do učenia ako trénovacia množina.

Keďže hustota fráz bola 5x až 6x väčšia pri štadardnej hlavičke gutenberg projektu, ktorá je pri každej publikácii rovnaká, dá sa predpokladať, že práve malá trénovacia množina tvorí hlavnú príčinu slabého rozpoznania.

7 Záver

V tejto práci sa nám podarilo pozhlukovať pojmy na základe ich podobností vypočítanej z ich asociatívneho okolia, a tak oddemonštrovať emergenciu sémantiky. Pomocou funkcie GetSynonyms sa nám podarilo z textu vydolovať významovo podobné objekty.

Experimenty na korpuse z prirodzeného jazyka naznačili, že systém má mnoho zaujímavých vlastností pri automatickom budovaní ontologií z textov v prirodzenom jazyku a je veľmi pravdepodobné, že sa v budúcnosti uplatní pri automatickom budovaní obrovských lexikónov, akým je napríklad WordNet. Experimenty nad umelou gramatikou poskytujú už hmatateľné dôkazy, že systém je schopný zhlukovať slová do morfologických skupín, ak má k dipozícii dostatočne veľkú trénovaciu množinu. Výsledky testov nad gramatikou by mohli byť spolu s testami nad korpusom z prirodzeného jazyka dosatočnou motiváciou na ďalšie skúmanie tejto teórie.

V budúcnosti by sa mohol výskum nadväzujúci na túto prácu zamerať na optimalizáciu programovej implementácie. Je smutné, že na Fakulte elektrotechniky a informatiky nemajú študenti k dispozícii stroj schopný pracovať naraz s veľkými kvantami dokumentov, potrebnými na kvalitné experimentovanie. Pri tvorbe tejto práce neuspel ani projekt Grid, ktorý sa z môjho pohľadu zaoberá skôr distribúciou procesorového času ako systémových prostriedkov, ku ktorým patrí aj pamäť počítača.

Zameranie sa na vlastnosť systému rozpoznávať frázy by mohla byť veľmi prospešná. Pri kvalitnejšom rozpoznávaní, čiže hľadaní zmysluplných úsekov by bolo možné systém využiť na filtrovanie zmysluplných informácií od šumu, tak ako aj možnú opravu zašumených dát.

Veľmi zaujímavým námetom na ďalšiu prácu by bolo nezastaviť sa pri rozpoznávaní fráz, ale pokračovať ďalej a snažiť sa z textov vydolovať vzťahy medzi frázami a potom aj medzi celými vetami. Systém by mohol poskytnúť veľmi zaujímavý prístup pri prekladoch dokumentov porovnávaním ontologických štruktúr.

1. Zoznam použitej literatúry

[1] Alberts, L. K.: YMIR: an Ontology for Engineering Design. University of Twente, PhD thesis, 1993

[2] Csontó, J. - Sabol, T.: Umelá inteligencia, TU Košice, 1991

[3] Ferguson, C. J., Goldie, S.: Applied Artificial Intelligence and the Management of Knowledge. Bristol Business School Teaching and Research Review Issue 3, Summer 2000, ISSN 1468-4578

[4] Gruber, T. R.: Towards principles for the design of ontologies used for knowledge sharing. In Guarino, N. & Poli, R., editors, Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer, 1994.

[5] Guarino, N., Giaretta, P.: Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N.J.I. Mars (ed.), Towards Very Large Knowledge Bases, IOS Press 1995, Amsterdam, pp. 25 – 32

[6] Guarino, N.: Understanding, Building, and Using Ontologies. LANDSEB–CNR, National Research Council, Padova

[7] KAON - an overview [online]. [cit. 2005-04-20]. Dostupné na internete: http://kaon.semanticweb.org/main_kaonOverview.pdf

[8] R. Hecht-Nielsen, A Theory of Cerebral Cortex, Proceedings of the 1998 International Conference on Neural Information Processing (ICONIP98), 1998

[9] Robert Kende, "Generovanie konceptuálnych popisov textových dokumentov použitím všeobecnej ontológie". Znalosti 2005. Stara Lesna, Slovensko, 9.-11.2.2205. Zbornik.

[10] Robert Kende, "Knowledge Modelling in Support of Knowledge Management". 107-112, Engineering of Intelligent Systems, 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2001, Budapest, Hungary, June 4-7, 2001, Proceedings.

[11] Schreiber, G., Wielinga, B., and Jansweijer, W.: The KAKTUS View on the 'O' Word. In Proceedings of IJCAI95 Workshop on Basic Onto-logical Issues in Knowledge Sharing. Montreal, Canada, 1995.

[12] Sowa, J., F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., ISBN 0-534-94965-7, 16 August 1999

[13] The Cyc knowledge server [online]. [cit. 2005-04-21]. Dostupné na internete: http://www.cyc.com/cyc/technology/whatiscyc

[14] Wikipedia, the free encyclopedia [online].[cit. 2005-04-21]. Dostupné na internete: http://en.wikipedia.org/wiki/Ontology

[15] Wikipedia, the free encyclopedia [online].[cit. 2005-04-21]. Dostupné na internete: http://en.wikipedia.org/wiki/Wordnet

2. Zoznam príloh

1. CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe, programy Beast, BeastLexicon, BeastApplet a BeastLearn, korpusy textov, nástroje na experimentovanie.
2. Používateľská príručka.
3. Systémová príručka.

3. Zoznam obrázkov