Marián Šimko

Jednoduché dolovanie v slovenskom texte

Ontožúr, 5.12.2008

Lexikálna analýza

Odstránenie stop slov

Lematizácia/Stemming

Zostavenie vektorovej reprezentácie

„Advanced“ techniky

Lexikálna analýza

- Vstup: sekvencia znakov
- Výstup: sekvencia lexém (tokenov, slov)
- Oddeľovače
  - úroveň viet: . ? ! ...
  - úroveň slov: , ; : ( ) -

Odstránenie tzv. stop slov

- Sémanticky nevýznamné slová
  - predložky, spojky, častice, ...
  - pozor, ak sledujeme kolokácie!
- Krátke slová
- Číslice
- Alfanumerický miš-maš
  - nbu123

Lematizácia, Stemming

- Lemma – slovníkový tvar
  - rybárovej -> rybár
- Stem – koreň slova
  - rybárovej -> ryb
- Angličtina
  - algoritmicky (Porter, 1980)
- Flexívne jazyky
  - inak...

Slovenčina

- JÚĽŠ SAV lematizátor
  - databáza lem a možných tvarov
  - ~ 25MB
  - hovorová slovenčina, chýbajú termíny
- Tvaroslovník
  - algoritmický prístup
  - porovnávanie na základe koncovky
  - pomalšie, nižšia úspešnosť

Vektorová reprezentácia

- Variant: tzv. Bag of Words (BOW) model
  - nezáleží na poradí
- Dimenzie vektora – slová/výrazy
- Hodnoty – frekvencie výskytu
- Príklad:
  „Ema má mamu. Mala Ema veľkého psa?”

term	Ema	mama	mať	veľký	pes
weight	2	1	2	1	1

TF-IDF

- Term Frequency – Inverse Document Frequency

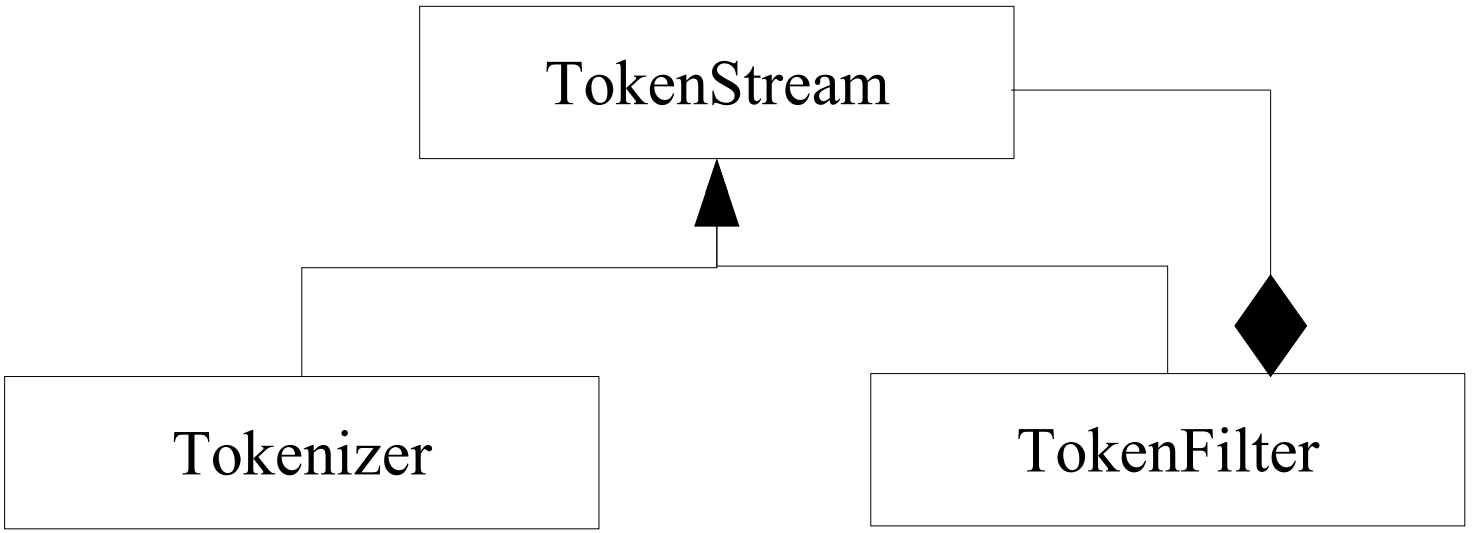
$$tf(t_i, d_j) = \frac{n(t_i, d_j)}{\sum_{t_k \in d_j} n(t_k, d_j)}$$

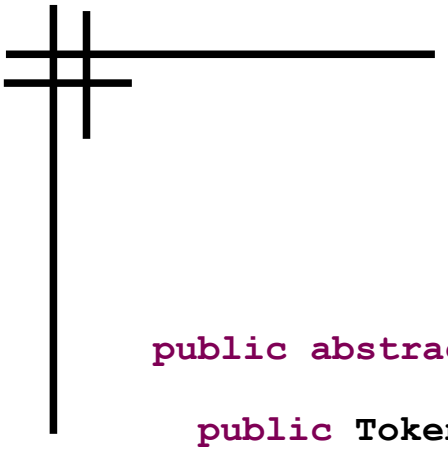
$$idf(t_i) = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

Apache Lucene

- Open source IR library
- **Java**, C#, Ruby, PHP, Perl, ...
- Fulltextové indexovanie a vyhľadávanie
- PDF, HTML, XML, MS Word, ...

org.apache.lucene.analysis.Analyzer





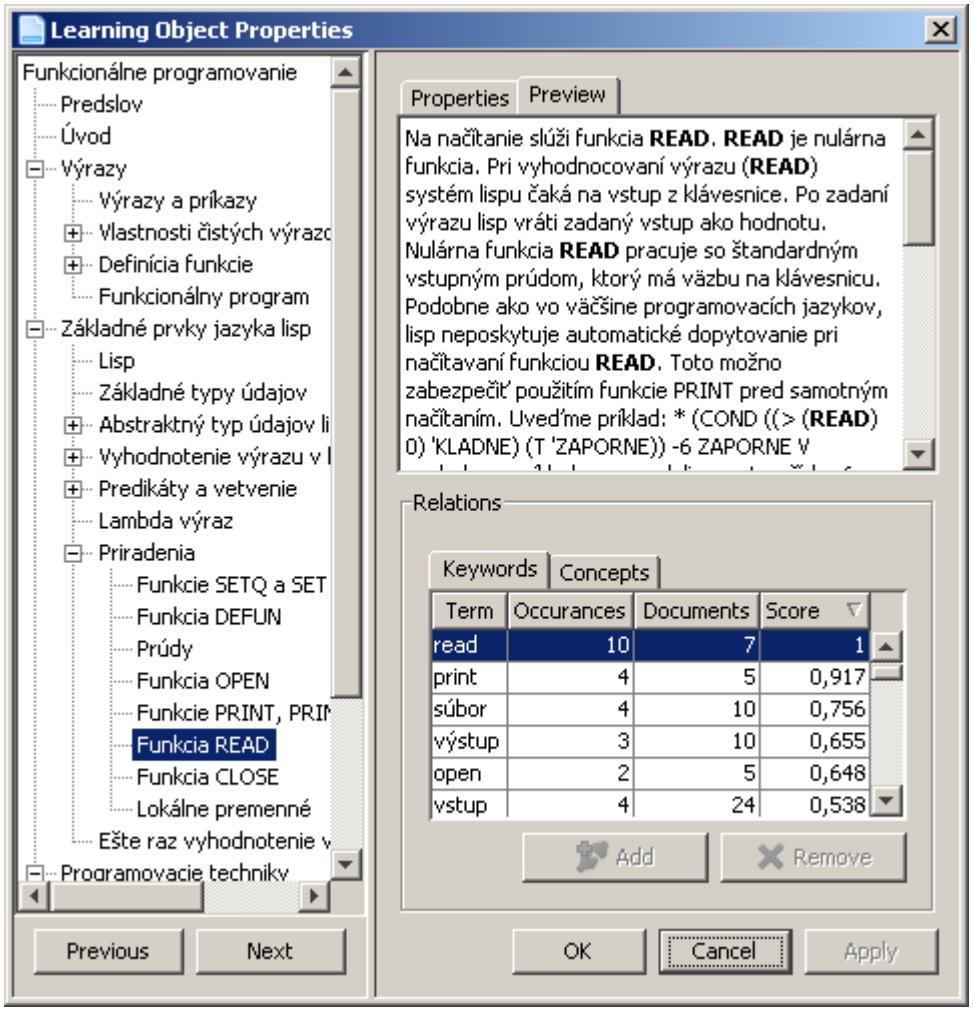
```
public abstract class TokenStream {
  public Token next() throws IOException {
    Token result = next(new Token());
    return result;
  }
}
```

```
public abstract class Analyzer {
  public abstract TokenStream tokenStream(String fieldName, Reader reader);
}
```

```
public final class SlovakLemmatizerFilter extends TokenFilter {

    public static final String TOKEN_TYPE_LEMMA = "LEMMA";

    private CdbLemmatizerSk lemmatizer;


    public SlovakLemmatizerFilter(TokenStream input) {
        super(input);
        lemmatizer = new CdbLemmatizerSk();
    }


    @Override
    public final Token next() throws IOException {
        token = input.next();
        List<String> lemmas = lemmatizer.lemmatize(token.termText());

        for (String lemma : lemmas) {
            Token lemmaToken = new Token(lemma, token.startOffset(),
                                         token.endOffset(), TOKEN_TYPE_LEMMA);
            return lemmaToken;
        }
    }
}
```

```
public class SlovakAnalyzer extends Analyzer {
    private Set<String> stopWords = new HashSet<String>();

    public SlovakAnalyzer() {
        setStopWordList(new File("resources/sk_stop.txt"));
    }

    @Override
    public TokenStream tokenStream(String fieldName, Reader reader) {
        return new TrashWordPositionalFilter(
                new SlovakLemmatizerFilter(
                new PositionalStopFilter(
                new LowerCaseTokenizer(reader),
                stopWords), false)));
    }
    ...
}
```

```
public class SlovakAnalyzer extends Analyzer {
    private Set<String> stopWords = new HashSet<String>();

    public SlovakAnalyzer() {
        setStopWordList(new File("resources/sk_stop.txt"));
    }

    @Override
    public TokenStream tokenStream(String fieldName, Reader reader) {
        return new TrashWordPositionalFilter(
                new ManualLemmatizerFilter(
                new SlovakLemmatizerFilter(
                new PositionalStopFilter(
                new MyLowerCaseTokenizer(reader),
                stopWords), false)));
    }
    ...
}
```


Learning Object Properties
Funkcionálne programovanie
Predslov
Úvod
Výrazy
Výrazy a príkazy
Vlastnosti čistých výrazc
Definícia funkcie
Funkcionálny program
Základné prvky jazyka lisp
Lisp
Základné typy údajov
Abstraktný typ údajov li
Vyhodnotenie výrazu v l
Predikáty a vetvenie
Lambda výraz
Priradenia
Funkcie SETQ a SET
Funkcia DEFUN
Prúdy
Funkcia OPEN
Funkcie PRINT, PRI
Funkcia READ
Funkcia CLOSE
Lokálne premenné
Ešte raz vyhodnotenie v
Programovacie techniky
Previous
Next
Properties
Preview
Na načítanie slúži funkcia READ. READ je nulárna funkcia. Pri vyhodnocovaní výrazu (READ) systém lispu čaká na vstup z klávesnice. Po zadaní výrazu lisp vráti zadaný vstup ako hodnotu. Nulárna funkcia READ pracuje so štandardným vstupným prúdom, ktorý má väzbu na klávesnicu. Podobne ako vo väčšine programovacích jazykov, lisp neposkytuje automatické dopytovanie pri načítavaní funkciou READ. Toto možno zabezpečiť použitím funkcie PRINT pred samotným načítaním. Uveďme príklad: * (COND ((> (READ) 0) 'KLADNE) (T 'ZAPORNE)) -6 ZAPORNE V
Relations
Keywords
Concepts
Term
Occurances
Documents
Score
read 10 7 1
print 4 5 0,917
súbor 4 10 0,756
výstup 3 10 0,655
open 2 5 0,648
vstup 4 24 0,538
Add
Remove
OK
Cancel
Apply

Odkazy

- JÚĽŠ SAV lematizátor
  GARABÍK, R. 2006. Slovak morphology analyzer based on levenshtein edit operations. In *Proceedings of 1st Workshop on Inteligent and Knowledge Oriented Technologies*, WIKT 2006, Bratislava, Slovakia, 2006, pp. 2-5.
- Tvaroslovník
  KRAJČI, S., NOVOTNÝ, R. Hľadanie základného tvaru slovenského slova na základe spoločného konca slov. In *Proceedings of 1st Workshop on Inteligent and Knowledge Oriented Technologies*, WIKT 2006, Bratislava, Slovakia, 2006, pp. 99-101.
- http://cr.yp.to/cdb.html
- http://lucene.apache.org