Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Tomáš V e s e l ý

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Automatic Synonyms Creation

Guidelines:

Design an algorithm for finding synonyms from the open sources (Internet) for the purpose of a query rewrite in a full-text search engine. More precisely, a system for replacing the original word/term without changing original meaning. Order the list of synonyms from the closest to the most distant words. Suggest a criteria for setting a usibility threshold of selected synonyms for the purpose or a query rewrite. Design criteria for scoring the quality of the selected synonyms. Use test data to estimate the algorithm performance.

Bibliography/Sources:

[1] Xing Wei, Fuchun Peng, Huihsin Tseng, Yumao Lu, Benoit Dumoulin - Context Sensitive Synonym Discovery for Web Search Queries - Proceedings of the 18th ACM Conference on Information and Knowledge Management (1585 - 1588) – New York, USA - 2009.
[2] John A. Bullinaria, Joseph P. Levy – Extracting semantic representations from word co-occurrence statistics stop-lists, stemming, and SVD - 2012.
[3] Ronen Feldman, James Sanger - The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data - Cambridge University Press - 2006.

Bachelor Project Supervisor: Ing. Jan Šedivý, CSc.

Valid until: the end of the summer semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 10, 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Tomáš V e s e l ý

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Automatické vytváření synonym

Pokyny pro vypracování:

Navrhněte přístup, jak využít volně dostupné zdroje pro vyhledávání synonym pro účely rozšíření dotazu do fulltextového vyhledávače. Přesněji, pro libovolné slovo/slovní spojení a kontext vygenerujte seznam slov/slovních spojeních, kterými může být původní slovo/slovní spojení v daném kontextu nahrazeno při zachování stejného významu. Seznam uspořádejte od nejsilnějších synonym po nejslabší. Navrhněte kritérium, podle kterého se zvolí hranice použitelnosti pro účely fulltextového vyhledávání. Na testovacích datech změřte kvalitu navrženého systému.

Seznam odborné literatury:

[1] Xing Wei, Fuchun Peng, Huihsin Tseng, Yumao Lu, Benoit Dumoulin - Context Sensitive Synonym Discovery for Web Search Queries - Proceedings of the 18th ACM Conference on Information and Knowledge Management (1585 - 1588) – New York, USA - 2009.
[2] John A. Bullinaria, Joseph P. Levy – Extracting semantic representations from word co-occurrence statistics stop-lists, stemming, and SVD - 2012.
[3] Ronen Feldman, James Sanger - The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data - Cambridge University Press - 2006.

Vedoucí bakalářské práce: Ing. Jan Šedivý, CSc.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

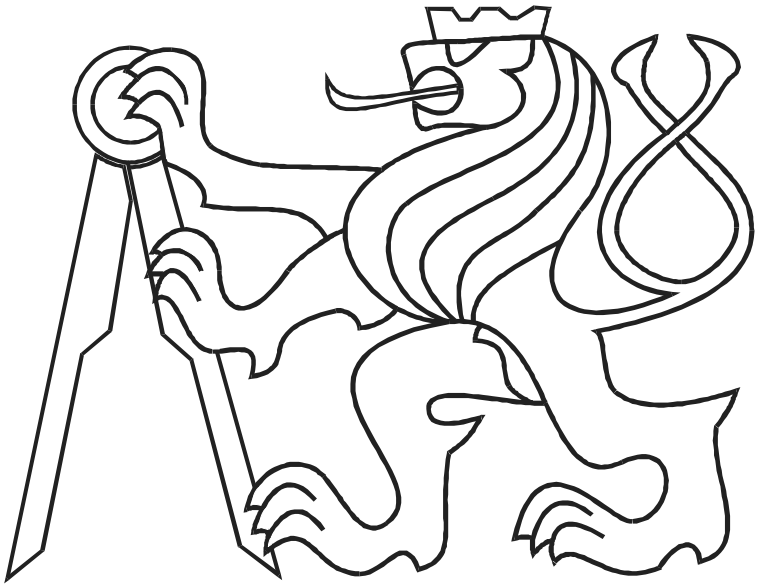
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Bachelor project



Tomáš Veselý

Automatic Synonyms Creation

Department of Cybernetics

Project supervisor: **Ing. Jan Šedivý, CSc.**

Declaration

I declare that I worked out the presented thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis.

In Prague ....

Acknowledgements

I would like to express my deep gratitude to Ing. Jan Šedivý, CSc., my supervisor for his professional guidance and valuable support.

Access to the CERIT-SC computing and storage facilities provided under the programme Center CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, reg. no. CZ. 1.05/3.2.00/08.0144, is greatly appreciated.

Abstract

Synonyms are words with the same or similar meaning. They can be any part of speech, if both words are the same part of speech. Synonyms are important for Natural Language Processing. For example in full-text search one of the relevant signals for finding a relevant web page is the existence of the query words or similar words in the document. To achieve this, the search query is expanded using all possible synonyms to make the search as broad as possible. In this work, all kinds of methods for automatic synonyms discovery are presented and compared. Then some evaluation methods are proposed for evaluation of the discovered synonyms. Last, one method is thoroughly tested on the proposed evaluation methods.

Abstrakt

Synonyma jsou slova se stejným nebo podobným významem. Mohou být jakéhokoli slovního druhu, ale obě slova musí mít stejný slovní druh. Synonyma jsou důležitá pro zpracování přirozeného jazyka. Například ve fulltextovém vyhledávání je jedním z kriterií pro nalezení relevantního dokumentu existence slov z dotazu nebo podobných slov v dokumentu. Aby vyhledáváni bylo co nejvíce generalizováno je vyhledávací dotaz rozšířen pomocí všech možných synonym. V této práci, jsou popsány a porovnány různé druhy metod pro automatické vyhledávání synonym. Dále jsou navrženy metody pro testovaní kvality těchto nalezených synonym. Na závěr je navrženými testovacími metodami důkladně otestován jeden z algoritmů.

Contents

1 Problem formulation **1**

1.1 Query expansion 2

2 Algorithms **3**

2.1 Dictionary based algorithms 3

2.1.1 Generalized Kleinberg's method 4

2.1.2 Problems 4

2.2 Web based methods 5

2.2.1 Pointwise mutual information - information retrieval 5

2.2.2 Synonyms discovery from full-text search log 6

2.2.3 Problems 7

2.3 Corpora based algorithms 7

2.3.1 Positive Pointwise Mutual Information with Cosine 8

3 Evaluation **11**

3.1 TOEFL synonym questions 11

3.1.1 Sample question 11

3.2 ESL synonym questions 12

3.3 Thesaurus evaluation 12

3.4 Human evaluation 12

3.5 Our approach 13

3.5.1 Application for creating test dataset 14

4 Experiments **17**
4.1 Data 17
4.1.1 Data preprocessing 19
4.2 Collecting evaluation data 19
4.3 Results 19
4.3.1 Testing parameters of the PPMIC 20
4.3.2 Size of the corpus 22
4.3.3 Generalization of the corpus 23

5 Conclusion **25**

A List of Abbreviations **29**

B Additional figures **31**

List of Figures

3.1 Screenshot of web application. 15

4.1 Performance of the PPMIC algorithm depending on *Caron* P value and number of dimensions used. Dataset A is used. 20
4.2 Performance of the PPMIC algorithm depending on *Caron* P value and number of dimensions used. Dataset B is used. 21
4.3 Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 1$. Dataset A is used. 22
4.4 Performance of the PPMIC algorithm depending on *Caron* P and context window with 4000 dimensions. Dataset A is used. 24

B.1 Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 0.15$. Dataset A is used. 31
B.2 Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 1$. Dataset B is used. 32
B.3 Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 0.15$. Dataset B is used. 33
B.4 Performance of the PPMIC algorithm depending on *Caron* P and context windows with 4000 dimensions. Dataset B is used. 34
B.5 Performance of the PPMIC algorithm depending on number of dimensions and context windows used with *Caron* $P = 0.15$. Dataset A is used. 35
B.6 Performance of the PPMIC algorithm depending on number of dimensions and context windows used with *Caron* $P = 0.15$. Dataset B is used. 36

List of Tables

2.1 Comparison of results for target word *velký* without SVD and with SVD with 1000 Principal components 10

4.1 Corpuses characteristic 18

Chapter 1

Problem formulation

Synonyms are words with the same or similar meaning. They can be any part of speech, if both words are the same part of speech. Synonyms can be used to improve performance in many applications. For example synonyms can be used in query expansion for full-text search engine to make the search more general. Another usage is in translation, natural language generation and after all in communication, where we can look on synonyms as spice and flavor of language. However, language is evolving rapidly in this age and new words are being invented as new products are released every day. People on the web are distorting words, hence static synonyms lists which can be found in thesauri[1] are not enough. The language is evolving on every day basis and this bachelor project is focusing on automatic generation of synonyms. The remainder of this document will address the synonyms discovery for query expansion. For query expansion we need synonyms that are:

- **Context sensitive** synonyms depends on the context in which they are used. For example in sentence *"Dentist removes scale from the teeth"* we can change the word *scale* with synonym *tartar*. But in this context it is not possible use the word *measure* as synonym to word *scale* without changing the sentence meaning.

- **Domain specific** synonyms depends on the domain of the usage. For example for word *CTU* is synonym term *Czech Technical University* in the university domain but in this domain *Counter Terrorist Unit*[2] is not synonym to word *CTU*. On the other hand in some domain about TV shows term *Counter Terrorist Unit* is synonym for word *CTU*.

- **Time sensitive** synonyms are evolving over time such as term *new Xbox* which today in 2014 is synonym to term *Xbox one* but in 2005 synonym to *new Xbox* was term *Xbox 360* and in 2001 it was term *Xbox*. As another example we can use elected officials such as president, senators, etc where their name is the synonym to

[1] Thesaurus is dictionary publication where words are grouped by their synonymity or antonymity
[2] Counter Terrorist Unit is an intelligence agency from TV show 24

the function they hold. For example today is *president Obama* synonym to *Barack Obama* but 10 years ago it would not be synonym.

1.1 Query expansion

Query expansion is a technique used mainly in full-text search for improving search performance and results relevance. When user enters query into search engine this query is expanded into multiple queries and the actual search is done on all these queries. Results are then combined and shown on the search engine result page (SERP) .

There are multiple ways how to expand query. Query can be expanded by spelling correction, synonyms, adding accent to words, stemming, lemmatization, etc. This work is focusing on enriching the query from set of the automatically generated synonyms. This work is not focusing on linguistic synonyms we are rather looking for similar word with the aim to provide desired search intent.

Chapter 2

Algorithms for synonyms discovery

In this chapter are presented some algorithms for synonyms discovery. They are divided to the three categories based on the data they are using to discover synonyms. Few representative algorithms are described briefly in each category and some very interesting algorithms are described thoroughly to provide better insight into problematic.

2.1 Dictionary based algorithms

Dictionary based algorithms are using monolingual or bilingual dictionaries or combination of both to find synonyms.

One of the methods to find synonyms is the ArcRank [1] method. This method is based on PageRank algorithm. Oriented graph constructed from monolingual dictionary, where words are nodes and there is edge from word w_i to w_j if w_j is in definition of w_i, is used as input for this algorithm.

Second method is called Distance method [2]. This method is using the same graph as the ArcRank as an input. From this graph adjacency matrix $\mathbf{M}$ is created and the distance between word w_i and w_j is defined as:

$$d(w_i, w_j) = \|\mathbf{M}_{i,\cdot} - \mathbf{M}_{j,\cdot}\| + \|(\mathbf{M}_{\cdot,i} - \mathbf{M}_{\cdot,j})^T\| \tag{2.1}$$

where $\|.\|$ is l_1 norm. To obtain synonyms, distances between the target word and all other words are calculated and synonyms to the target word are the words with the smallest distance.

Next method is Generalized Kleinberg's method [2]. This method is described more thoroughly in the section 2.1.1 because it is very interesting algorithm. Generalized Kleinberg's method is also algorithm which was the most successful in the comparison made by Blondel [2]. Comparison was done on the four carefully selected word. These words are described in chapter 3.4. Of course we can argue that comparison done on the four words is not very

meaningful but because of the problems with these methods described in 2.1.2 we won't try to compare them more meaningfully.

2.1.1 Generalized Kleinberg's method

This method is based on assumption that synonyms have many words in common in their definition and appear together in the definition of many words. Dictionary is used to construct graph G, where words are vertices and there exists edge from word u to word v if word is in definition of word u.

For given word w we create subgraph G_w of G, where vertices of G_w are those pointed by w or pointing to w. We can say that word w represents vertex 2 in the structure graph 2.2 and synonyms to w should also look like vertex 2 in the structure graph 2.2.

$$1 \rightarrow 2 \rightarrow 3 \tag{2.2}$$

For each vertex i in subgraph G_w we create three scores (x_i^1, x_i^2, x_i^3). Then algorithm 1 is run to generate scores for each vertex. Synonyms for word w are the words with the highest x_i^2 score. These x_i^2 scores are converging to the normalized principal eigenvector of the matrix $\mathbf{A}$. Matrix $\mathbf{A}$ is defined as:

$$\mathbf{A} = \mathbf{M}_w\mathbf{M}_w^T + \mathbf{M}_w^T\mathbf{M}_w \tag{2.3}$$

where $\mathbf{M}_w$ is adjacency matrix of G_w.

$x_{i,0}^1 = 1$;
$x_{i,0}^2 = 1$;
$x_{i,0}^3 = 1$;
repeat
  $x_{i,t+1}^1 = \sum x_{j,t}^2$, where j are vertices pointed by i;
  $x_{i,t+1}^2 = \sum x_{j,t}^1 + \sum x_{k,t}^3$, where j are vertices pointing to i and k are vertices pointed by i;
  $x_{i,t+1}^3 = \sum x_{j,t}^2$, where j are vertices pointing to i ;
  $x_{i,t+1}^k = \frac{x_{i,t+1}^k}{\|x_{i,t+1}^k\|}$, where $k \in \{1, 2, 3\}$
until *convergence*;

Algorithm 1: Kleinberg's algorithm

2.1.2 Problems

Dictionary based methods in general have many problems. As noted at the beginning of this text in chapter 1, time sensitive synonyms are needed for our application but due to usage of dictionaries, synonyms lists can not be updated very often. It is obvious that

maintaining monolingual dictionary updated would be very effortful. Also creating domain specific synonyms from these methods would be difficult as dictionaries are usually very general.

2.2 Web based methods

One of the method which uses web as input is Pointwise mutual information - information retrieval (PMI-IR) algorithm [3]. This algorithm is using web as large corpus. PMI-IR is in detail described in chapter 2.2.1.

The other type of the web based methods is using users of full-text search to find synonyms. These methods are using click logs from full-text search engine with stored information about user behavior during searching. Cheng et al. [4] suggest algorithm for discovering entity synonyms. By entity we mean things like products, movies, songs, etc. This algorithm is using click logs to accomplish this task.

The next method from Wei et al. [5] is also using click logs but instead of finding entity synonyms it is used to find regular synonyms. This method is in detail described in chapter. 2.2.2

2.2.1 Pointwise mutual information - information retrieval

The PMI-IR is co-occurrence based algorithm. Instead of using corpus to find similar words it is using advanced search capabilities of full-text search engine. The motivation to use full-text web search engine is that full-text web search engines are indexing much more text than any corpus so it could perform better. Consider target word t is given and synonyms are selected from set of choices $\{c_i\}$. The PMI-IR is using Pointwise Mutual Information (PMI) as a measure of the co-occurrence of two words. PMI is defined as:

$$I\left(c_i; t\right) = \log\left(\frac{P\left(c_i, t\right)}{P\left(t\right) P\left(c_i\right)}\right) \tag{2.4}$$

Synonyms are the words from set $\{c_i\}$ with the highest $I\left(c_i; t\right)$. Due to maximization equation 2.4 can be simplified to:

$$I\left(c_i; t\right) = \frac{P\left(c_i, t\right)}{P\left(c_i\right)} \tag{2.5}$$

PMI-IR uses full-text advanced search capabilities to estimate PMI. Some full-text search functions have to be defined. Let x and y be two phrases. Full-text search engine returns document if both phrases x and y are present in the document if we use operator AND as follows x AND y. Operator OR as is used as follows x OR y and engine returns document if x or y or both is present in document. NOT operator is used as NOT x and

engine returns documents which does not contain phrase x. The last operator is NEAR operator which returns documents which contain x and y in document and these phrases are specified within 10 words in the document. Operator NEAR is used as follows $x \,\mathrm{NEAR}\, y$. Last thing we need is function which indicates number of returned documents for the given query. This function is defined as:

$$hits\,(q) = \text{number of documents which search engine returns for query } q \tag{2.6}$$

Turney [3] presents four different versions of PMI-IR:

1. The simplest case is when two words co-occur when they appear in the same document:

$$I\,(c_i; t) = \frac{hits\,(t \,\mathrm{AND}\, c_i)}{hits\,(c_i)} \tag{2.7}$$

2. Another case is when documents contain both target t and c_i close together:

$$I\,(c_i; t) = \frac{hits\,(t \,\mathrm{NEAR}\, c_i)}{hits\,(c_i)} \tag{2.8}$$

   Where NEAR operator constraints search to documents where target t and c_i are used within 10 words of one another.

3. This version reduces problem with antonyms:

$$I\,(c_i; t) = \frac{hits\,((t \,\mathrm{NEAR}\, c_i) \,\mathrm{AND\,NOT}\, ((t \,\mathrm{OR}\, c_i) \,\mathrm{NEAR}\, \text{"}not\text{"}))}{hits\,(c_i \,\mathrm{AND\,NOT}\, (c_i \,\mathrm{NEAR}\, \text{"}not\text{"}))} \tag{2.9}$$

4. This version takes context into account:

$$I\,(c_i; t) = \frac{hits\,((t \,\mathrm{NEAR}\, c_i) \,\mathrm{AND}\, context \,\mathrm{AND\,NOT}\, ((t \,\mathrm{OR}\, c_i) \,\mathrm{NEAR}\, \text{"}not\text{"}))}{hits\,(c_i \,\mathrm{AND\,NOT}\, (c_i \,\mathrm{NEAR}\, \text{"}not\text{"}))} \tag{2.10}$$

2.2.2 Synonyms discovery from full-text search log

Wei et al. [5] describes algorithm how to use Web search query log with clicked URL to find context sensitive synonyms.

This algorithm uses queries with relevant documents and number of views and clicks for each of them as input.

First of all, similar queries based on their clicked distribution are necessary. To find clusters of similar queries Jensen-Shanon divergence (JSD) is used as distance measure. For each pair of queries JSD between their clicked distributions P and Q is computed. JSD is defined as:

$$JSD\,(P \| Q) = \frac{1}{2} D\,(P \| M) + \frac{1}{2} D\,(Q \| M) \tag{2.11}$$

where

$$M = \frac{1}{2}(P + Q) \tag{2.12}$$

$$D(P\|Q) = \sum_i \ln\left(\frac{P(i)}{Q(i)}\right) P(i) \tag{2.13}$$

Subsequently clustering of the queries is performed as follows: Each query will start in their own cluster and clusters will be merged greedily. In the clusters, queries with the same number of words and which differs only in one word, are aligned. From these aligned queries probability of reformulating word w_i to w_j can be defined as:

$$P(w_j|w_i) = \frac{\sum_k \text{sim}_k(w_i \to w_j)}{\sum_{w_j}\sum_k \text{sim}(w_i \to w_j)} \tag{2.14}$$

where $\text{sim}_k(w_i \to w_j)$ is similarity score of query q_k which reformulates w_i to w_j. This similarity is JSD between aligned queries. Probability of reformulating w_i to w_j in query q_k is defined as:

$$P(w_j|w_i, q_k) = sim_k(w_i \to w_j) \tag{2.15}$$

The probability $P(w_j|w_i)$ represents all-purpose synonyms and the $P(w_j|w_i, q_k)$ represents context sensitive synonyms for the given query q_k. To combine these concepts linear combination in log scale is used. The final equation is:

$$\log P_{q_k}(w_j|w_i) = \lambda \log P(w_j|w_i) + (1-\lambda)\log P(w_j|w_i, q_k) \tag{2.16}$$

where $\lambda \in (0, 1)$ is mixture weight.

2.2.3 Problems

There is few problem with the web based methods. PMI-IR brings interesting concept of using web as corpus. For solving TOEFL test (more about TOEFL test can be found in chapter 3.1) as described in [3] this method is successful. However, for usage for finding synonym for arbitrary word this concept is nearly impossible to use. Huge amount of queries into the search engine would need to be performed and that would be very time consuming.

Algorithms which are using click logs as input are not very convenient because getting hands onto click logs is nearly impossible. Only large search engines companies have access to these data.

2.3 Corpora based algorithms

Corpora based algorithms are using large amount of text to extract synonyms. The key idea is that synonyms are used in the roughly same way and on the similar places in the

text. One of the corpora based algorithms is Positive Pointwise Mutual Information with Cosine (PPMIC) algorithm which uses word to word co-occurrence to find synonyms. This algorithm is in detail described in chapter 2.3.1.

2.3.1 Positive Pointwise Mutual Information with Cosine

Bullinaria & Levy [6] suggest algorithms for finding similar words. It is word to word co-occurrence based algorithm where each word is represented as a vector.

Context window and some normalization function is required to calculate these vectors. Normalization function is used to normalize co-occurrence counts to be independent on the corpus size. Context window can be any size and shape but experiments done by Bullinaria & Levy [6] shows that rectangular window of small size performs best together with Positive PMI (eq. 2.4) as normalization function.

First, co-occurrence counts are computed. Counts can be represented as a matrix:

$$\mathbf{M}_{counts} = \{m_{ij}\} \tag{2.17}$$

where rows represents words, columns represents contexts and m_{ij} is number of times word j is in context of word i.

Co-occurrence counts are shown in matrix $\mathbf{M}_{counts}$ for corpus c with three sentences and context window of size ± 1.

$$\begin{aligned} c = \{&\text{sentence with four words,} \\ &\text{another sentence with five words,} \\ &\text{another term}\} \end{aligned}$$

$$\mathbf{M}_{counts} = \begin{array}{c} \\ \text{sentence} \\ \text{with} \\ \text{four} \\ \text{words} \\ \text{another} \\ \text{five} \\ \text{term} \end{array} \begin{array}{c} \begin{array}{ccccccc} \text{sentence} & \text{with} & \text{four} & \text{words} & \text{another} & \text{five} & \text{term} \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Then these counts are normalized to be independent on the corpus size by PMI. All negative values of PMI are set to zero. This will happen if co-occurrence number will

be smaller than the expected number. For the example normalized counts of the matrix $\mathbf{M}_{counts}$ are shown in matrix $\mathbf{M}_{pmi}$:

$$\mathbf{M}_{pmi} = \begin{array}{c} \\ \text{sentence} \\ \text{with} \\ \text{four} \\ \text{words} \\ \text{another} \\ \text{five} \\ \text{term} \end{array} \begin{array}{c} \begin{array}{ccccccc} \text{sentence} & \text{with} & \text{four} & \text{words} & \text{another} & \text{five} & \text{term} \end{array} \\ \begin{pmatrix} 0 & 1.70 & 0 & 0 & 1.01 & 0 & 0 \\ 1.70 & 0 & 1.70 & 0 & 0 & 1.70 & 0 \\ 0 & 1.70 & 0 & 1.70 & 0 & 0 & 0 \\ 0 & 0 & 1.70 & 0 & 0 & 1.70 & 0 \\ 1.01 & 0 & 0 & 0 & 0 & 0 & 1.70 \\ 0 & 1.70 & 0 & 1.70 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.70 & 0 & 0 \end{pmatrix} \end{array}$$

Because each word is represented as vector obtaining synonyms is very simple. For target word t for which we are searching synonyms, distance between t and all other words is calculated and synonyms are the words with the smallest distance. This distance can be any measure but experiments done by Bullinaria & Levy [6] shows that cosine distance $d_{cos}(\mathbf{a}, \mathbf{b})$ yields best results. Cosine distance $d_{cos}(\mathbf{a}, \mathbf{b})$ is defined as:

$$d_{cos}(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{2.18}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors.

To improve performance [7], Singular Value Decomposition (SVD) can be used to perform data smoothing. By SVD every matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ can be decomposed as

$$\mathbf{M} = \mathbf{USV}^T \tag{2.19}$$

where

1. $\mathbf{S} \in \mathbb{R}^{m \times n}$ is diagonal and contains singular values of $\mathbf{M}$ in decreasing order
2. $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$
3. $\mathbf{V} \in \mathbb{R}^{n \times n}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$

SVD can be used to find the nearest matrix with lower rank in terms of Frobenion norm. Where Frobenion norm of matrix $\mathbf{M}$ is defined as:

$$\|\mathbf{M}\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right) \tag{2.20}$$

Solution to finding matrix with lower rank is matrix:

$$\mathbf{M}' = \mathbf{US}'\mathbf{V}^T \tag{2.21}$$

where $\mathbf{S}'$ is $\mathbf{S}$ but with $r - r'$ smallest singular values set to zero.

Cosine distance is invariant to the orthogonal rotation, hence matrix $\mathbf{US}'$ can be used instead of matrix $\mathbf{M}'$. Moreover, final matrix can be changed into $\mathbf{US}'^P$, where P is so called *Caron P* value. Singular values are raised to the *Caron P* power. *Caron P* is smaller that 1, hence large singular values will have the smaller impact. This modification also yields better results as is noted in [7].

In table 2.1 are top 10 similar words for word *velký* with SVD and without SVD.

w/o SVD	with SVD
obrovský	veliký
malý	obrovský
a	ohromný
takový	značný
i	takový
jeho	větší
další	pořádný
tento	obří
který	malý
nebo	nesmírný

Table 2.1: Comparison of results for target word *velký* without SVD and with SVD with 1000 Principal components

Chapter 3

Evaluation

In this section we discuss different types of evaluation for automatic synonyms acquisition. First, commonly used evaluation methods are presented. After that some new evaluation methods will be suggested.

3.1 TOEFL synonym questions

Test Of English as a Foreign Language (TOEFL) [8] is a test which contains synonyms questions among many other things. These synonyms questions are often used to compare different algorithms for synonym acquisition. The big comparison table can be found here [8]. This test consist of 80 questions. Each question contains one target word, for which we are selecting synonym, and four possible answers, one of them is correct. Algorithms are compared by percentage of correctly answered questions. Average non-English US college applicant can solve this test with 64.5% correctly answered questions. Data are available on request by contacting LSA Support at CU Boulder [9].

3.1.1 Sample question

Stem:
levied

Choices:

- imposed
- believed
- requested
- correlated

Correct answer:
imposed

3.2 ESL synonym questions

English as a Second Language (ESL) [10] is a test which consist of 50 synonyms questions. Each question includes a sentence, providing context. There is one marked word in sentence which represents a word for which we are selecting synonym from four choices, one of them is correct. Data are available on request from Peter Turney [11].

Sample question

Stem:
A **rusty** nail is not as strong as a clean, new one.

Choices:

- corroded
- black
- dirty
- painted

Correct answer:
corroded

3.3 Thesaurus evaluation

This method is used in [12]. Three online thesauri are consulted for evaluation synonyms. At first target word is used to obtain synonyms from online thesauri and then union of the obtained synonyms is used as answer set of synonyms for target word. Words marked as "informal", "slang", "idiom" and multi word phrases were taken out from the answer sets.

3.4 Human evaluation

This method is used in [2]. They have carefully chosen 4 words for their variety.

- "**disappear** a word with various synonyms such as vanish" [2]

- "**parallelogram** a very specific word with no true synonyms but with some similar words: quadrilateral, square, rectangle, rhomb..." [2]
- "**sugar** a common word with different meanings (in chemistry, cooking, dietetics ...). One can expect glucose as a candidate." [2]
- "**science** a common and vague word. It is hard to say what to expect as synonym. Perhaps knowledge is the best option." [2]

They asked 21 people to rank the list of 10 synonyms for each word and rank the list from 0 to 10 (10 being best one). Order of the words was randomly chosen. Then average mark of the lists was used to compare four methods for synonyms discovery.

3.5 Our approach

There are problems with the methods described above. TOEFL and ESL is selecting only one synonym from four choices. Human evaluation as described above is almost impossible to do for large amount of data. We are not looking for synonyms from language point of view so thesaurus evaluation is also not possible. Also thesauri are of very poor quality for Czech language. Because of the problems described we propose better approach for testing automatic sensitive synonyms acquisition.

First is the Normalized Discounted Cumulative Gain (NDCG).

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

where $IDCG_p$ is the maximum possible DCG_p and

$$DCG_p = \sum_{i=0}^{p} \frac{2^{rel_i} - 1}{log_2(i+2)}$$

where rel_i is relevance of the i-th word. There is a problem with NDCG, it does not penalize bad words in the result set and does not penalize missing words in the result set.

The alternative metric $R_{precision}$ is solving the problem with bad and missing words in the result set. It is defined as:

$$R_{precision} = \frac{|R_{|Rel|} \cap Rel|}{|Rel|}$$

where Rel is a set of synonyms for the target word w and $R_{|Rel|}$ is a set of top $|Rel|$ similar words of the target word w.

Because $R_{precision}$ does not say anything about order of the synonyms, both metricies will be used in the experiments. Testing data for these metricies are generated by application described in chapter 3.5.1

3.5.1 Application for creating test dataset

Web application for creating testing dataset was created. The application backend is written in Python and is using MySQL database. The frontend is written in HTML and JavaScript and it is dynamically generated by PHP.

Once the user logs-in, randomly chosen words are picked from the database (Picture 3.1) and displayed. The selected synonyms are displayed in a random order in three frames. In fact each word is a button. The user may by pressing the button chose the order of relevance in the green frame or can label it as an antonym in the red frame. The grey frame offers the option to remove the word completely. The last frame can be used for suggesting more synonyms.

The application offers keyboard shortcuts. Each keyboard row control one category (synonyms, antonyms, inappropriate). The *esc* key refreshes the page and shows another word. In case of an error the button *Something wrong with this word, inform admin* can be used to inform admin.

A simple gamification features are improving the app. Users are honored by one point for rating a word or three points for creating a new synonym or antonym. Top 10 most successful users are promoted on a list. All users actions are logged and saved for future processing. This data might be useful in the future.

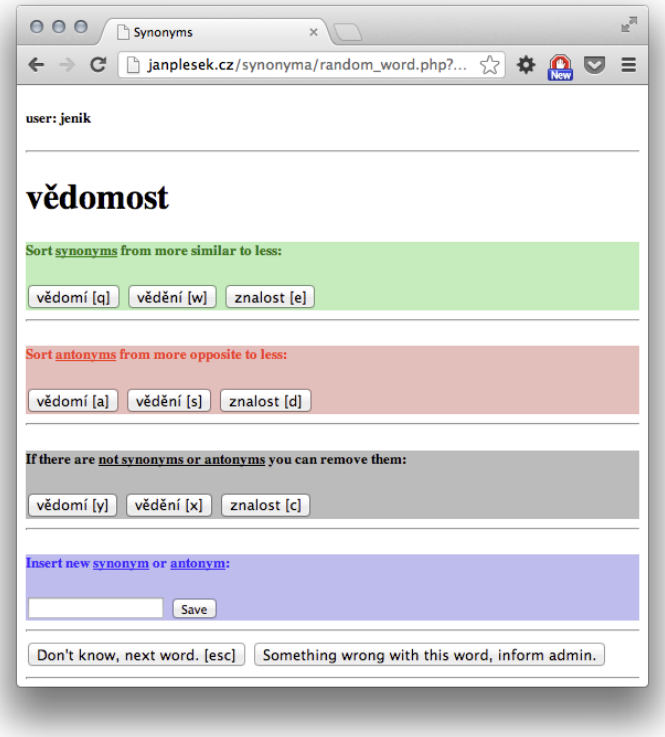


Figure 3.1: Screenshot of web application.

Chapter 4

Experiments

We have done theoretical comparison of different types of methods and came to the conclusion that corpus based methods have the biggest potential. Corpus based method eliminates dictionary based problem such as maintaining synonyms updated also obtaining input data is extremely easy and computational complexity is reasonable.

From the corpus based methods we decided to attempt to apply the PPMIC algorithm to find synonyms for query expansion. In this chapter are described experiments done on our implementation of the PPMIC algorithm. First, used corpus is described together with used preprocessing. Second, acquisition of testing data is depicted. Last, completed experiments are presented.

4.1 Data

As input to test the PPMIC algorithm corpus SYN version 2 is used. This corpus is developed at the Institute of the Czech National Corpus and available from http://korpus.cz/. It is non reference[1] corpus created by merging all reference SYN corpuses. The version 2 is created from SYN2000, SYN2005, SYN2006PUB, SYN2009PUB and SYN2010. At the time of writing there was released version 3 of SYN corpus which additionally contains SYN2013PUB. SYN corpus is lemmatized and contains morphological tags. This corpus is not representative[2], and is mainly created by news and magazines. The size of the corpuses from which is SYN created is shown in table 4.1.

[1] reference corpus is corpus which is static, it does not evolve over time.
[2] representative corpus is corpus which is genre balanced.

Corpus	Word count	Characteristic
SYN2010	100M	genre balanced, mostly from 2005-2009
SYN2009PUB	700M	news and magazines from 1995-2007
SYN2006PUB	300M	news and magazines from 1989-2004
SYN2005	100M	genre balanced, mostly from 2000-2004
SYN2000	100M	genre balanced, mostly from 1990-1999

Table 4.1: Corpuses characteristic

Due to licensing we obtained corpus in shuffled form. Corpus was divided into blocks and these blocks were shuffled. Size of the block was at most 100 words. In the listing 4.1 you can see example from this corpus showing one sentence.

```
<opus autor="Mlčoušek, Jiří" nazev="Hajný Vítězslav a fořt Bořivoj" naklad
<doc id="1">
<block>
<s id="1">
Každý                každý                PLMS1-------------
si                   se                   P7-X3-------------
dovede               dovést               VB-S---3P-AA---P
představit           představit           Vf---------A---P
,                    ,                    Z:----------------
jak                  jak                  Db----------------
důležitá             důležitý             AAFS1-----1A-----
byla                 být                  VpFS---3R-AA---I
Bořivojova           Bořivojův            AUFS1M-------------
dohlížecí            dohlížecí            AAFS1-----1A-----
činnost              činnost              NNFS1-----A-----
.                    .                    Z:----------------
</s>
...
</block>
...
</doc>
</opus>
...
```

Listing 4.1: Example of text from syn v2 corpus

4.1.1 Data preprocessing

Plain raw text is created from the provided corpus. Information about used morphological analysis tool, it's model version, etc wasn't provided, therefore the lemmatized version of words were not used. MorphoDiTa [13], tool for morphological analysis was used instead. This tool uses state of the art algorithms for Czech language and is published as open source.

Dictionary is created from tokenized and lemmatized text obtained by MorphoDiTa from the raw text. All the words used less than 350 times in corpus were removed. This word removal causes decreasing number of unique words from order of millions to approximately 71 thousands. Threshold 350 can be seen as too much but in context that corpus is lemmatized and corpus size is 1300M words it doesn't seem too much. This way size of the matrix is not extremely large and computation is feasible on normal hardware.

4.2 Collecting evaluation data

To create evaluation data we needed prepopulate database of the application for creating evaluation data. This application is described in chapter 3.5.1. Data we used is online Czech thesaurus available at http://www.slovnik-synonym.cz/. Then users were asked to verify already presented synonyms, add new synonyms and sort them according their synonymity. Users also obtained advice how to proceed in case of uncertainty. They should think up example with full-text search when they can replace the words and expect the same results. This way the first evaluation dataset A was created. This dataset A contains 285 test where each test consists of target word and correct synonyms ordered by their synonymity.

The second dataset B was created from the data created by this algorithm. Five thousands words were randomly picked and for each word 10 most similar words were generated by PPMIC algorithm. These data were entered to the application and again people were asked to process them with the same informations. The second dataset B contains 600 tests. More than 20 people have participated in the ranking synonyms.

4.3 Results

In this chapter are shown results for three experiments. First experiment is testing how parameters can change accuracy for our task. Second experiment is conducted to get some sense how size of the corpus can affect performance. Third experiment is focused how corpus generalization can improve results.

4.3.1 Testing parameters of the PPMIC

The first experiment was conducted to get information how PPMIC can perform on our task. We will use $R_{precision}$ and $NDCG_{10}$ measures to test precision of PPMIC, these measures are explained in chapter 3.5. In the rest of this document, $NDCG_{10}$ will be signed as $NDCG$ for convenience. Figure 4.1 shows the dependency of performance on the number of dimensions with the changing value of $Caron\ P$. It can be seen from figure 4.1 that for $Caron\ P$ smaller than 1 we are getting the best results. We are selecting synonyms from the set of words with size of the dictionary (71 thousand words as explained in chapter 4.1.1), which is much harder task than TOEFL test (selecting one synonym from 4 choices) but results seem to be consistent with the TOEFL task as reported by Bullinaria & Levy [7].

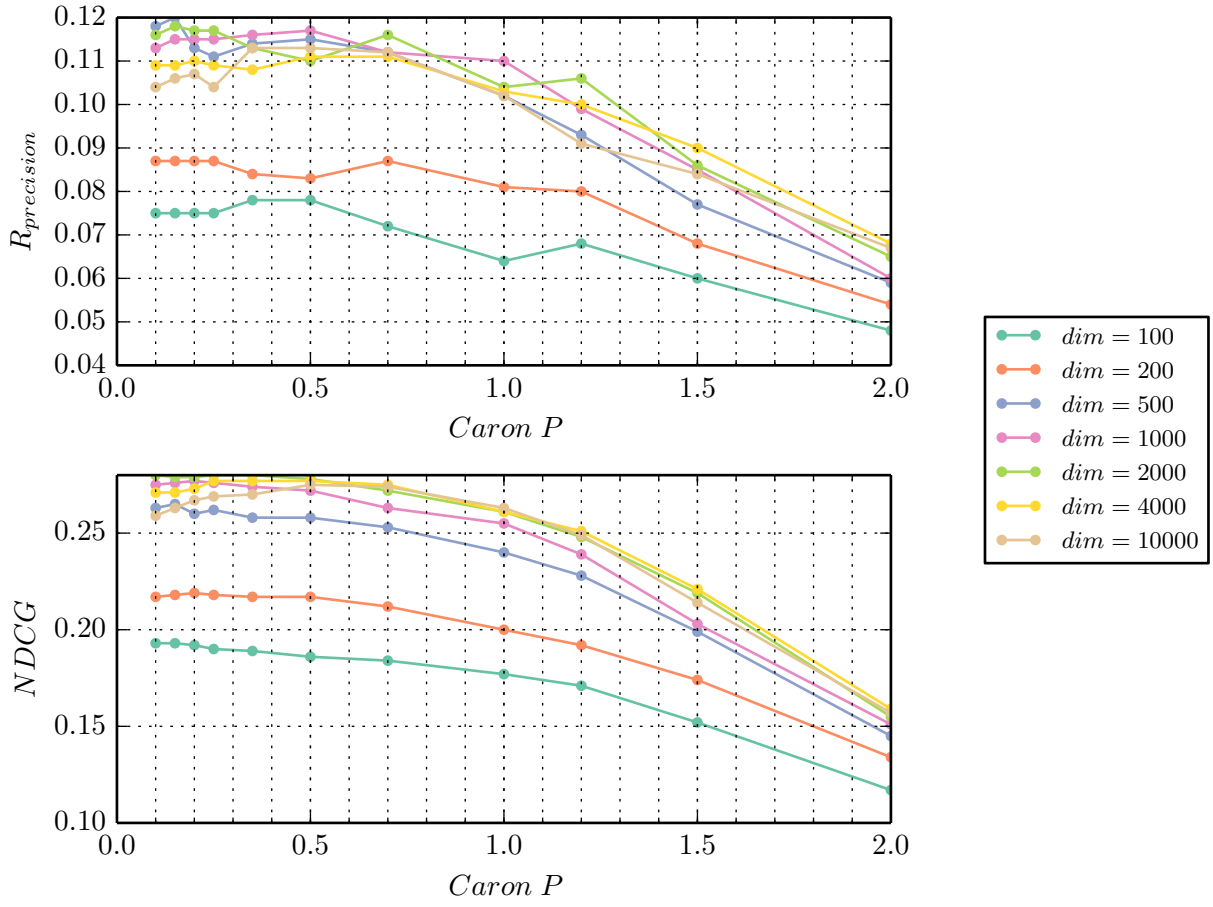


Figure 4.1: Performance of the PPMIC algorithm depending on $Caron\ P$ value and number of dimensions used. Dataset A is used.

For $R_{precision}$ best results are achieved with $Caron\ P = 0.15$ and 500 dimension.

However, the results for 500 dimensions are very unstable. Using the $NDCG$ measure we are receiving more stabler results. The Best performance is at 2000 and 4000 dimension where results do not depend so much on $Caron\ P$ value and are almost constant for $Caron\ P$ smaller than 0.5.

$NDCG$ and $R_{precision}$ numbers seem to be rather low so we inspected synonyms sets returned by PPMIC by hand. We noticed that PPMIC results are much better than our numbers from the dataset A. Specifically in full-text search we are not looking for synonyms in linguistic point of view but rather words which are similar in some sense. One of the explanations why our numbers seem to be lower than the subjective feeling can be that we don't have complete set of correct synonyms. To deal with this problem we decided to create dataset B. How this dataset B was created is described in chapter 4.2.

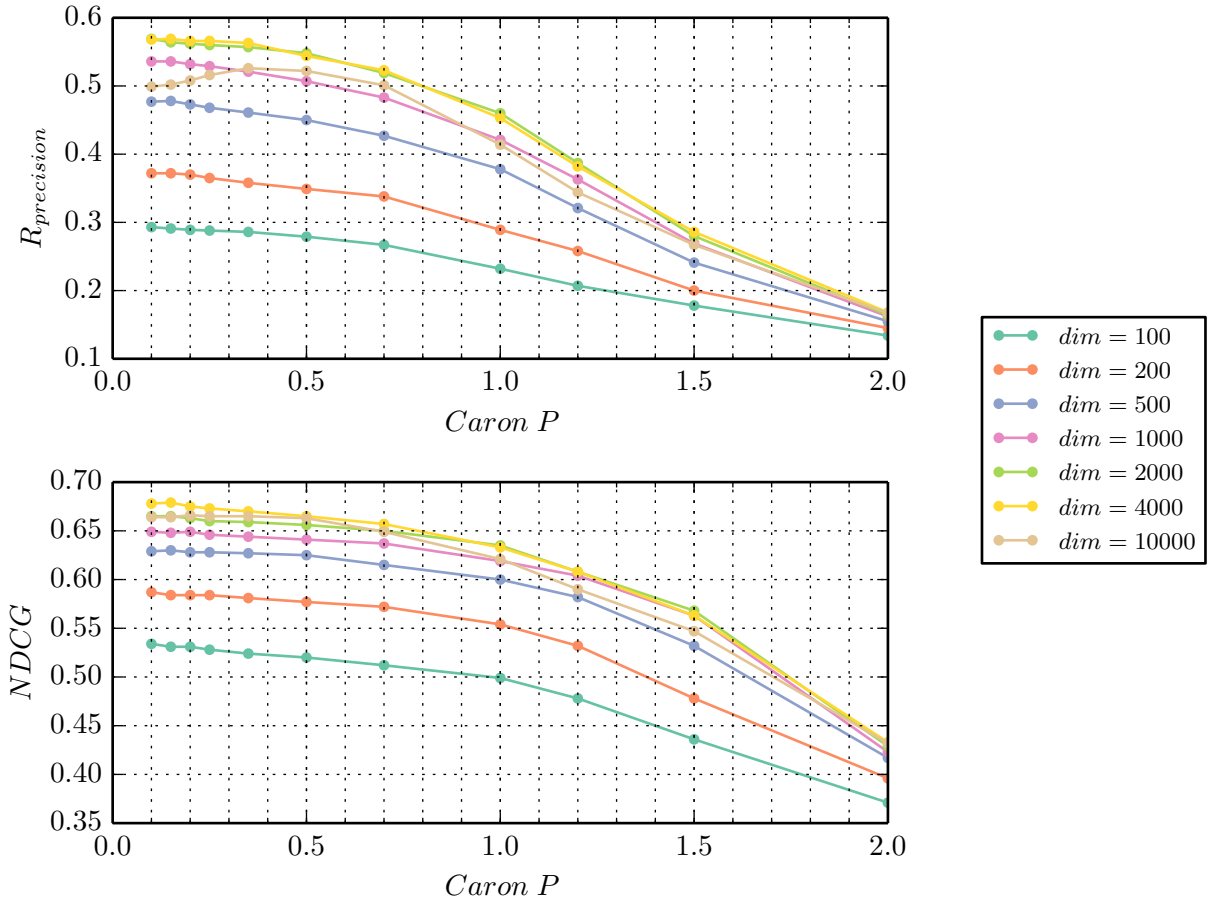


Figure 4.2: Performance of the PPMIC algorithm depending on $Caron\ P$ value and number of dimensions used. Dataset B is used.

Consequently, we repeated the experiment for dataset B. Results for the dataset B are in figure 4.2. Here we can see that results are much better. All the curves seem to be much

smoother. Best results are obtained for 4000 dimension with *Caron* $P = 0.1$ for $R_{precision}$ metric and *Caron* $P = 0.15$ for $NDCG$. And again it confirms that *Caron* P smaller than 1 improves results.

4.3.2 Size of the corpus

At the second experiment, we divided corpus into chunks and tested how does performance change with different sized corpus.

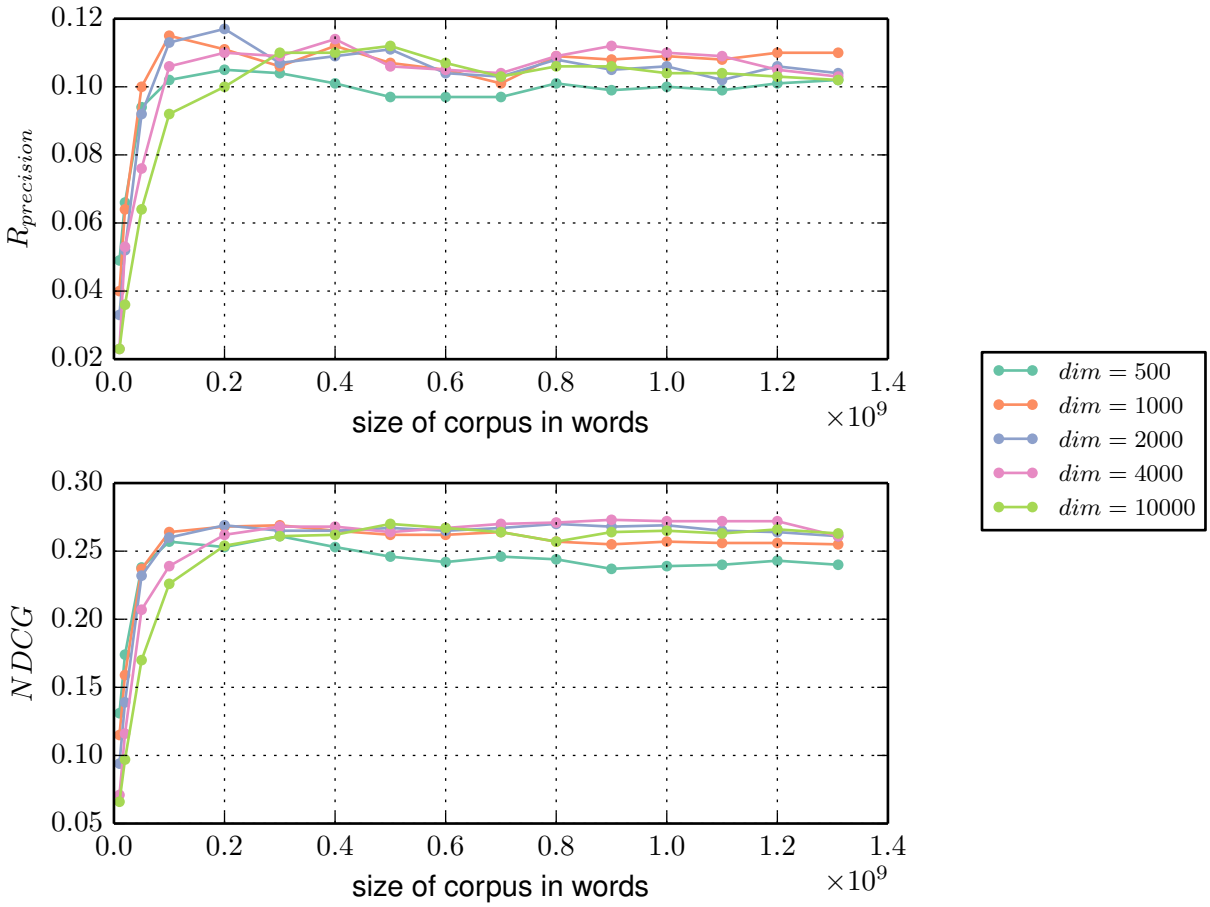


Figure 4.3: Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 1$. Dataset A is used.

In figure 4.3 is depicted how performance changes by increasing the size of the corpus. It is tested on the dataset A with *Caron* $P = 1$. Next comparisons for different *Caron* P and dataset B are in figures B.1, B.2 and B.3 in appendix. As can be seen performance almost hits ceiling at 300M words for the dataset A and 800M words for the dataset B. For

larger corpuses performance is almost constant. We can observe very small improvement for second dataset for larger corpus. This is quite in contradiction to what Bullinaria & Levy in [7] found out. Their performance was increasing with larger corpus significantly. It can be caused by many reasons.

The first reason is that the TOEFL test they used consists of rather rare synonyms. In other words, these synonyms have low occurrence counts in the corpus. Certainly, the low statistical significance of the synonyms can be solved by increasing the size of the corpus. However, our dataset contains synonyms with good counts in the corpus and by increasing it we do not gain much of the new information about these synonyms.

The second reason could be that corpus, which was used by Bullinaria & Levy is created by crawling the web. This corpus quality will be probably lower that SYN corpus (chapter 4.1 created from news, magazines and books. By using a better quality corpus, in other words, with a less noise present, the representation of the words could be more precise and by increasing the size of the corpus we don't gain much of the new information.

4.3.3 Generalization of the corpus

In the next experiment we tried to improve results by making text more general. To make text more general, the first thing that comes into considerations are entities. By entities we mean things like names, rivers, cities, etc. Other things that could be used to make the text more general are word classes, part of the speech, morphological tags, etc.

In our experiments we replaced all entities of the same type with unique term. Corpus with replaced entities then can be used in the same way as the original corpus. To find entities, tool called NameTag [14] is used. This tool achieves state of the art performance for Czech language.

In figure 4.4 is depicted the PPMIC performance depending on *Caron* P for modified corpus, original corpus SYN and various context windows (±1, ±2 and ±3). There can be seen improvements in $R_{precision}$ measure on the dataset A with context window ±1. This experiment also confirms that using the context window ±1 performs best, as was shown by Bullinaria & Levy in [7]. By using context window ±2 or ±3 performance drops significantly for both performance measures.

For the dataset B results are in figure B.4 and there is very small improvement in performance for modified corpus with *Caron* $P > 0.7$. Performance results depending on number of dimensions are in figure B.5 for dataset A and in the figure B.6 for the dataset B. These results are consistent with the results obtained for original corpus.

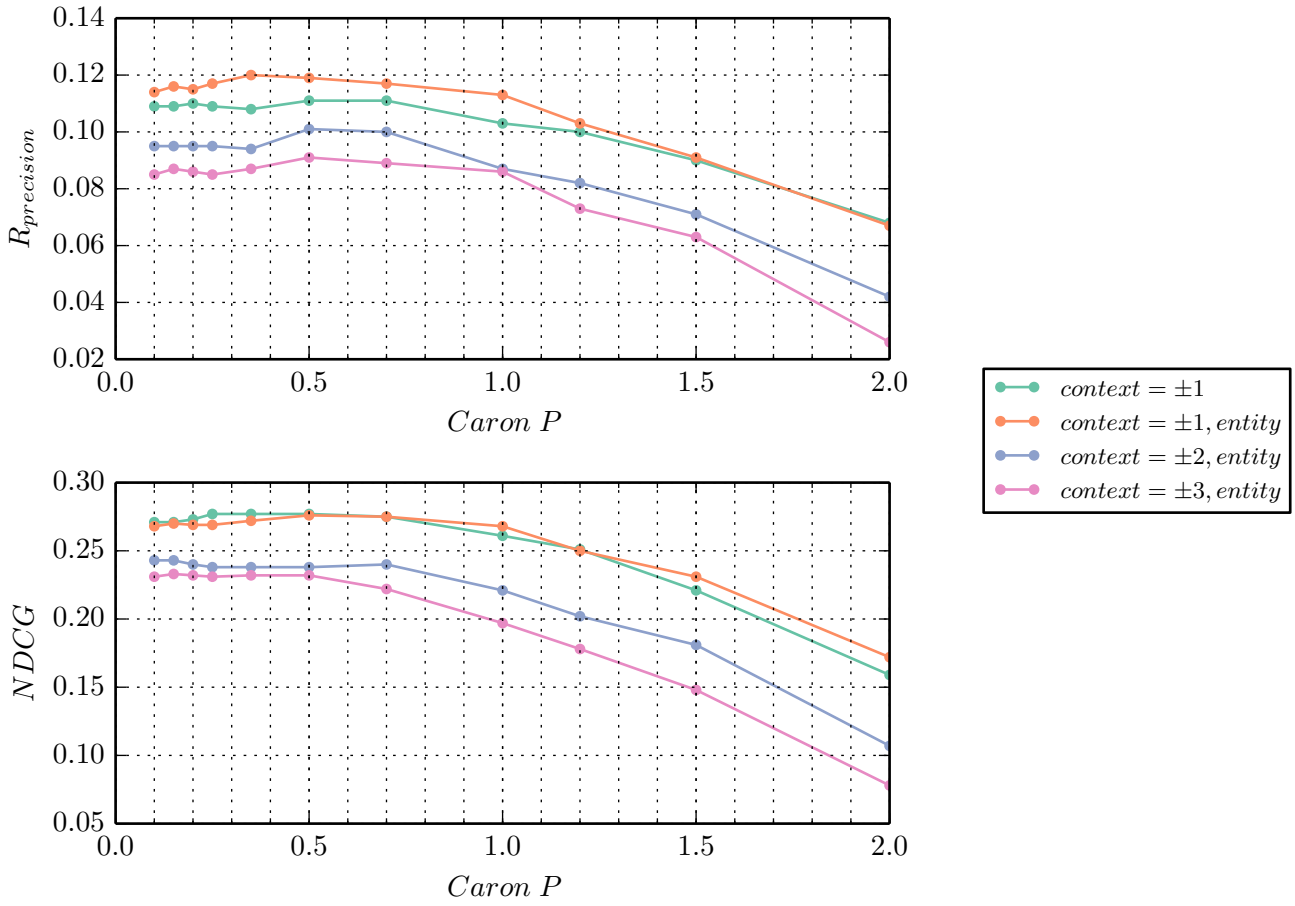


Figure 4.4: Performance of the PPMIC algorithm depending on *Caron P* and context window with 4000 dimensions. Dataset A is used.

Chapter 5

Conclusion

In this work we compared some of the algorithms, which could be used for synonyms discovery. From our comparisons it is clear that the most promising are the corpus based methods. These algorithms require large amount of data sets. Currently we can collect large enough data sets from our target domain internet by crawling the web.

We created a system for automatic synonyms discovery for query expansion based on the PPMIC algorithm. For the purpose of query expansion we can precompute set of synonyms for every word in the dictionary. Hence we can retrieve synonyms during query expansion in constant time. This is very convenient, especially in query expansion where we need expand queries extremely fast.

Also evaluation method for testing performance of this system was created. To measure performance two criteria were chosen $NDCG$ and $R_{precision}$. Unfortunately in the experiments we have proved that creating a testing data set is very difficult. To make the test as precise as possible we created web based application where users can improve the test. Even human can't think of every possible form of word which algorithm can return and which can be marked as correct. To really evaluate quality of this system, real application test will be needed to find out where to cut the list of the correct synonyms.

Nevertheless, this system yields performance for $R_{precision}$ metric 57.1% on our dataset B. We have also shown that generalization of the corpus by replacing entities can improve performance. Other generalization techniques can be explored in the future. Such as use word classes, morphological tags, part of the speech, etc. Generally speaking, returned set of synonyms by the algorithm could be used for query expansion but we can't be sure about real performance until the real performance will be measured by performing A/B testing.

We would like to thank to the people from Seznam.cz for their guidance and their help with creating testing dataset. To run the experiments presented in this work, more than 700 days of CPU time was needed. This would not be possible without access to the CERIT-SC computing and storage facilities. Also we would like to thank Ing. Jan Plešek for programming web application for creating evaluation data.

Bibliography

[1] Jan Jannink. *Thesaurus Entry Extraction from an On-line Dictionary*. 1999.

[2] Vincent D. Blondel. "Automatic extraction of synonyms in a dictionary". In: *in Proceedings of the SIAM Text Mining Workshop*. 2002.

[3] Peter Turney. "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL". In: (2001).

[4] Tao Cheng, Hady W. Lauw, and Stelios Paparizos. "Entity Synonyms for Structured Web Search". In: *Knowledge and Data Engineering, IEEE Transactions on* 24.10 (2012), pp. 1862–1875. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.168.

[5] Xing Wei et al. "Context sensitive synonym discovery for web search queries". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 1585–1588. ISBN: 978-1-60558-512-3. DOI: 10.1145/1645953.1646178. URL: http://doi.acm.org/10.1145/1645953.1646178.

[6] John A. Bullinaria and Joseph P. Levy. "Extracting semantic representations from word co-occurrence statistics: A computational study". In: *Behavior Research Methods* (2007), pp. 510–526.

[7] John A. Bullinaria and Joseph P. Levy. "Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD". English. In: *Behavior Research Methods* 44.3 (2012), pp. 890–907. DOI: 10.3758/s13428-011-0183-8. URL: http://dx.doi.org/10.3758/s13428-011-0183-8.

[8] *ACL - TOEFL Synonym Questions*. May 15, 2014. URL: http://aclweb.org/aclwiki/index.php?title=TOEFL_Synonym_Questions_(State_of_the_art).

[9] *LSA Support at CU Boulder*. URL: http://lsa.colorado.edu/mail_sub.html.

[10] *ACL - ESL Synonym Questions*. May 15, 2014. URL: http://aclweb.org/aclwiki/index.php?title=ESL_Synonym_Questions_(State_of_the_art).

[11] *Peter Turney*. URL: http://www.apperceptual.com/.

[12] Masato Hagiwara, Yasuhiro Ogawa, and Katsuhiko Toyama. "Supervised Synonym Acquisition Using Distributional Features and Syntactic Patterns". In: *Information and Media Technologies* 4.2 (2009), pp. 558–582.

[13] Jana Straka Milan; Straková. *MorphoDiTa: Morphological Dictionary and Tagger*. eng. 2014. URL: http://hdl.handle.net/11858/00-097C-0000-0023-43CD-0.

[14] Jana Straka Milan; Straková. *NameTag*. eng. 2014. URL: http://hdl.handle.net/11858/00-097C-0000-0023-43CE-E.

[15] M. Křen. "The SYN Concept: Towards One-Billion Corpus of Czech". In: *Proceedings of the Corpus Linguistics Conference CL2009, Liverpool*. 2009.

[16] Jan Hajič. *Disambiguation of Rich Inflection (Computational Morphology of Czech). Vol. 1*. Karolinum Charles University Press, Praha 2004.

[17] Tomáš Jelínek. "Nové značkování v Českém národním korpusu". In: *Naše řeč, 91, 1*. 2008, pp. 13–20.

[18] Drahomíra Spoustová et al. "The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech". In: *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing. ACL 2007, Praha*, pp. 67–74.

[19] Vladimír Petkevič. "Reliable Morphological Disambiguation of Czech: Rule-Based Approach is Necessary". In: *Insight into the Slovak and Czech Corpus Linguistics (Šimková M. ed.). Veda, Bratislava*. 2006, pp. 26–44.

[20] Jana Straková, Milan Straka, and Jan Hajič. "A New State-of-The-Art Czech Named Entity Recognizer". In: *Text, Speech and Dialogue: 16th International Conference, TSD 2013. Proceedings*. (Plzeň, hotel Angelo). Ed. by Ivan Habernal and Václav Matoušek. Vol. 8082. Lecture Notes in Artificial Intelligence. Západočeská univerzita v Plzni. Berlin / Heidelberg: Springer Verlag, 2013, pp. 68–75. ISBN: 978-3-642-40584-6.

Appendix A

List of Abbreviations

ESL English as a Second Language

JSD Jensen-Shanon divergence

NDCG Normalized Discounted Cumulative Gain

PMI Pointwise Mutual Information

PMI-IR Pointwise mutual information - information retrieval

PPMIC Pointwise Mutual Information with Cosine

SERP Search engine result page

SVD Singular Value Decomposition

TOEFL Test Of English as a Foreign Language

Appendix B

Additional figures

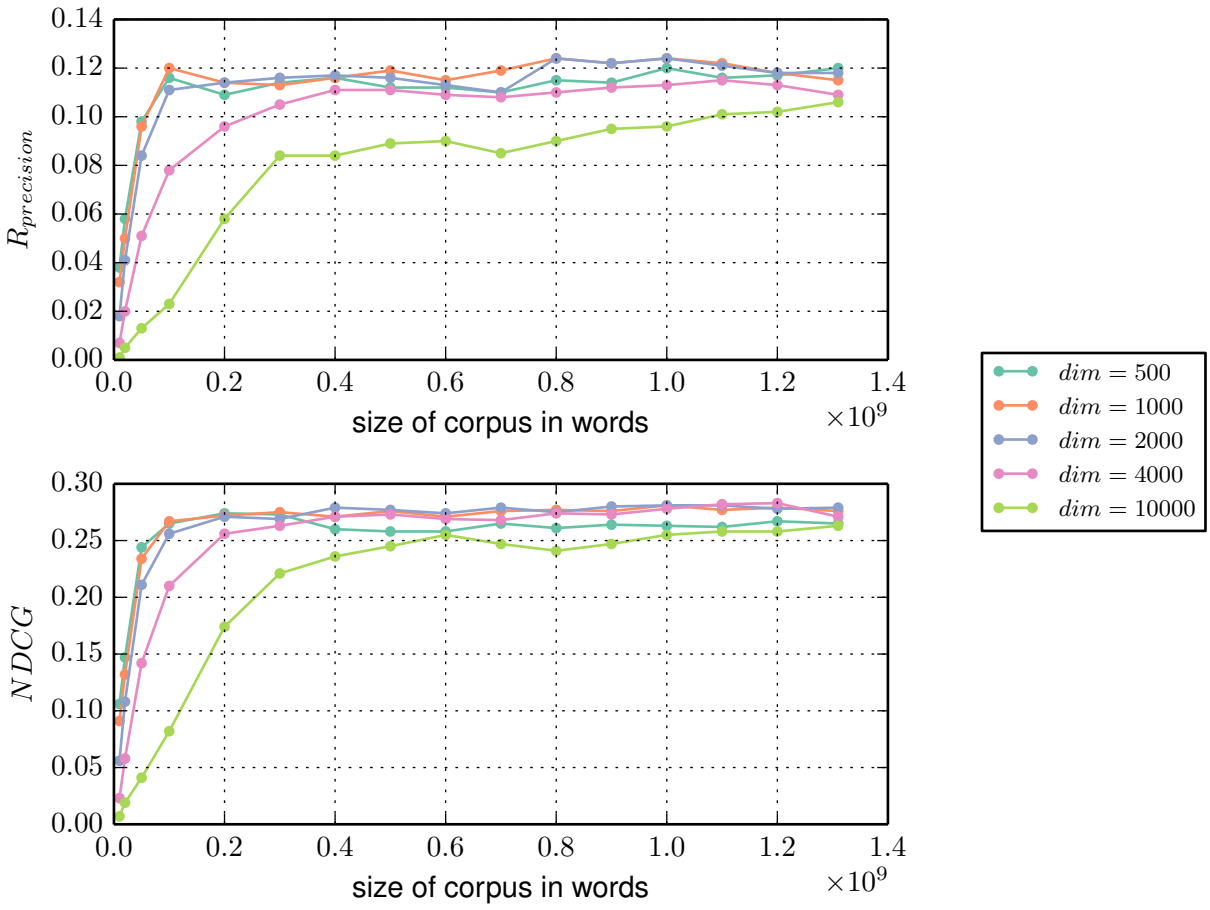


Figure B.1: Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 0.15$. Dataset A is used.

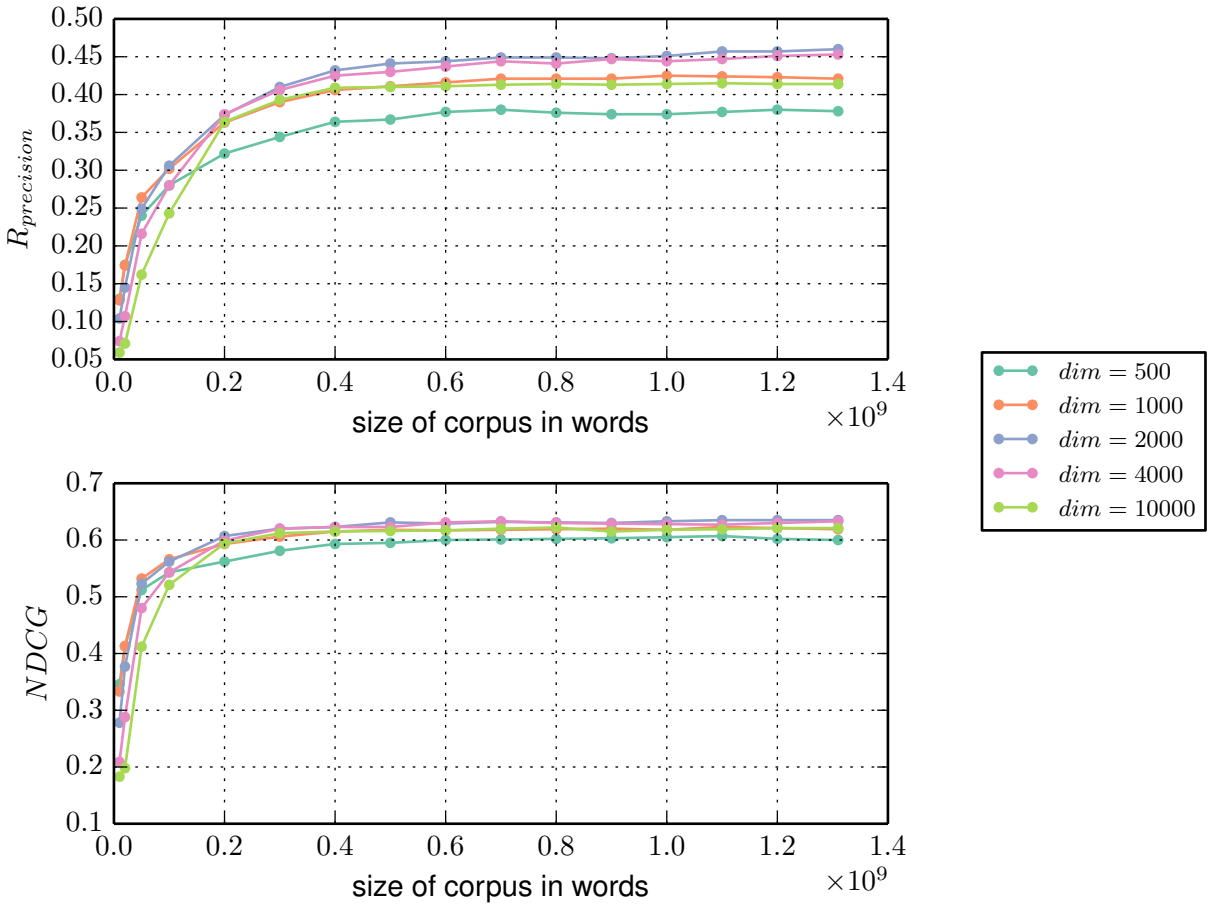


Figure B.2: Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 1$. Dataset B is used.

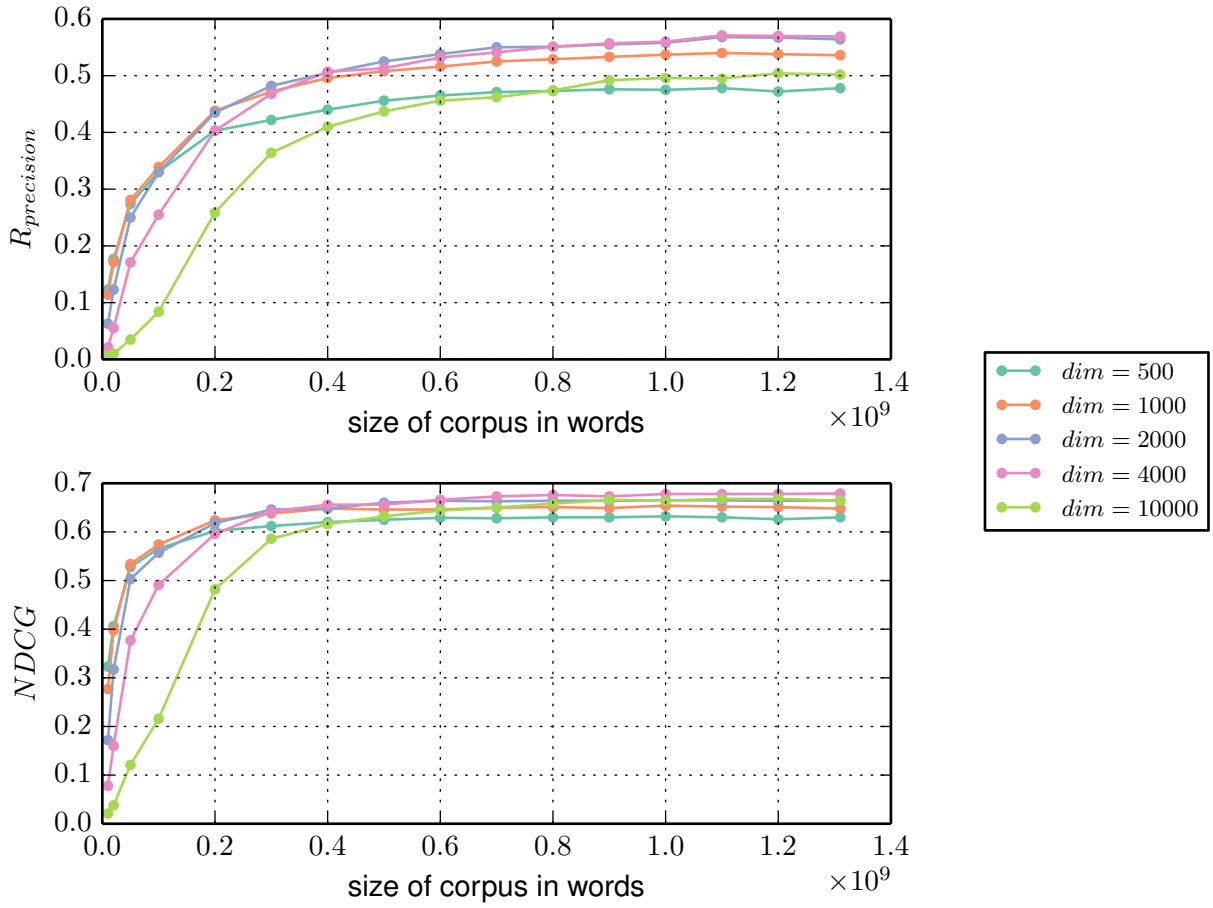


Figure B.3: Performance of the PPMIC algorithm depending on size of the corpus with *Caron* $P = 0.15$. Dataset B is used.

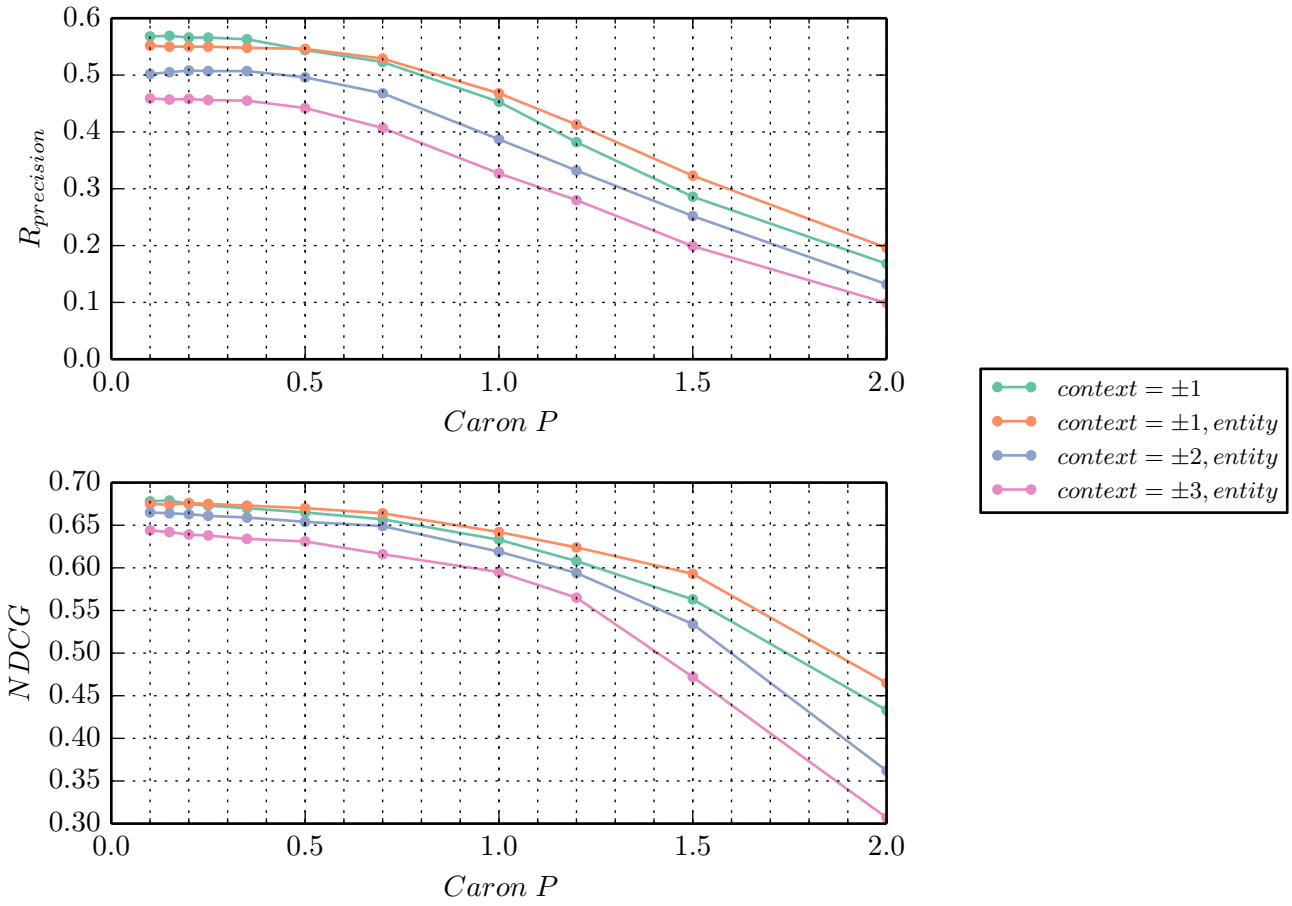


Figure B.4: Performance of the PPMIC algorithm depending on *Caron P* and context windows with 4000 dimensions. Dataset B is used.

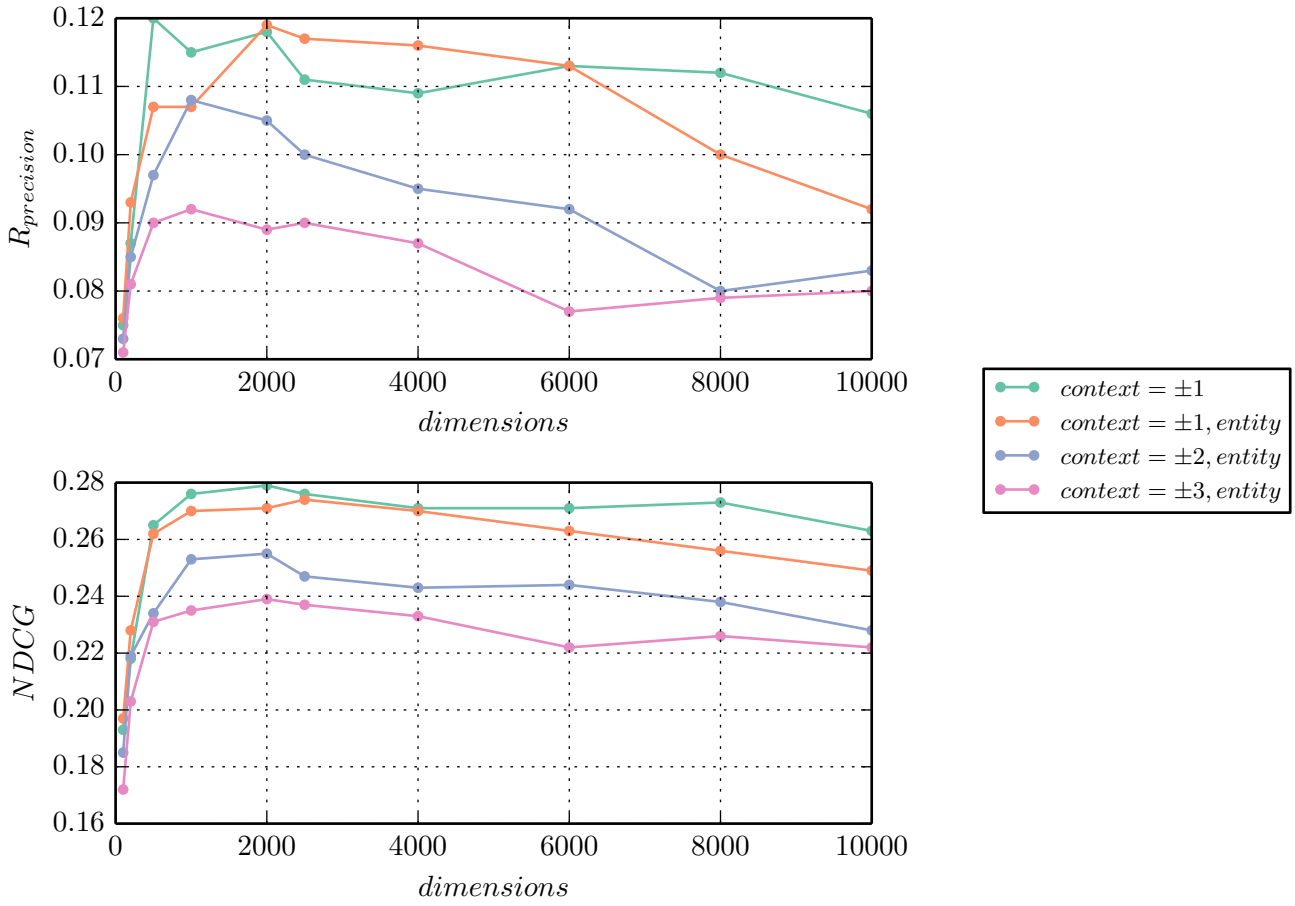


Figure B.5: Performance of the PPMIC algorithm depending on number of dimensions and context windows used with *Caron* $P = 0.15$. Dataset A is used.

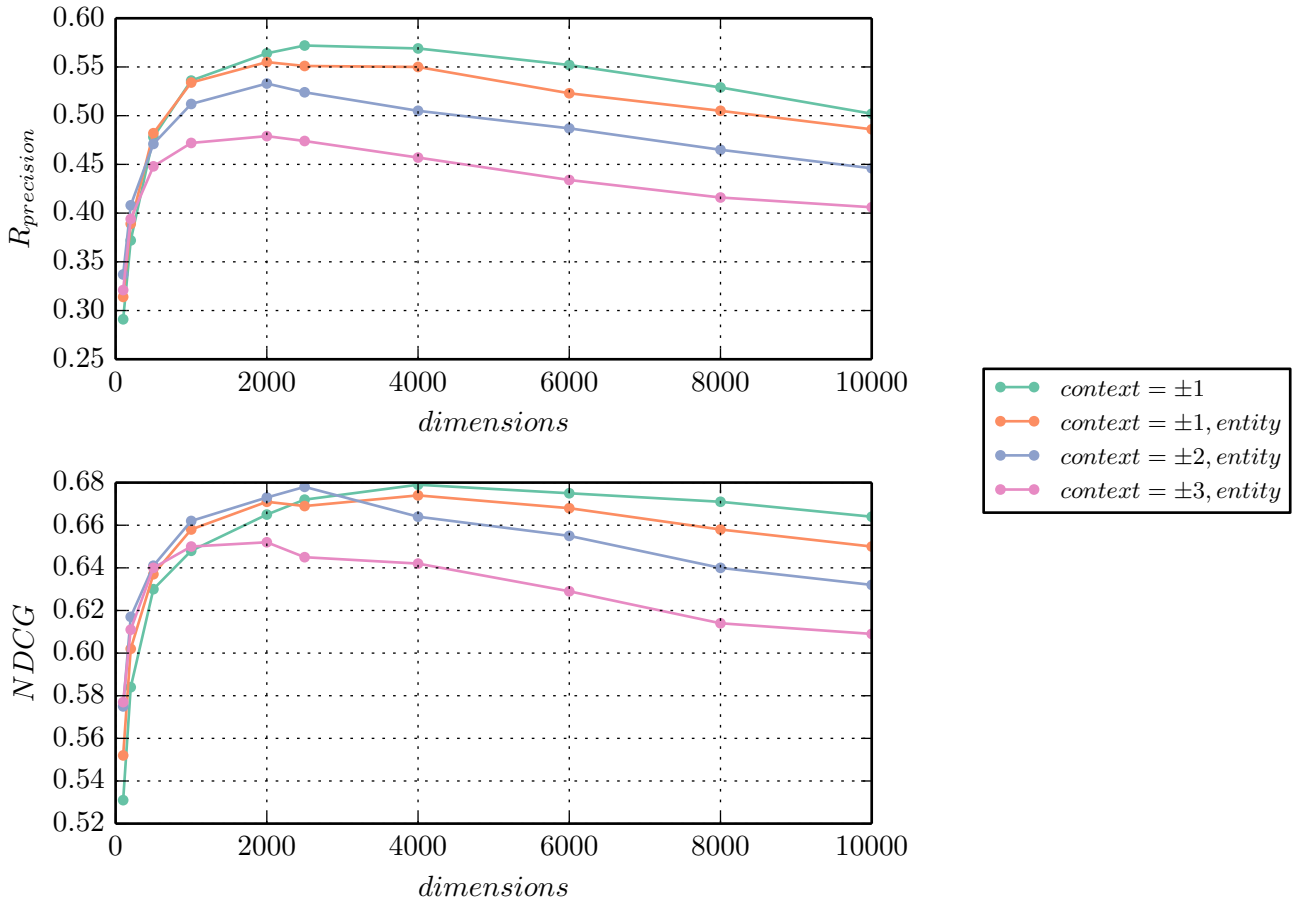


Figure B.6: Performance of the PPMIC algorithm depending on number of dimensions and context windows used with *Caron* $P = 0.15$. Dataset B is used.