Exploring
Latent Semantic Vector Models
Enriched With N-grams

Leif Grönqvist

Växjö University Press

Series editors: Tommy Book and Kerstin Brodén
ISSN: 1404-4307
ISBN: 91-7636-519-0
Printed by: Intellecta Docusys, Göteborg 2006
Exploring
Latent Semantic Vector Models
Enriched With N-grams

Leif Grönqvist
School of Mathematics and Systems Engineering
Växjö University, Sweden

The Graduate School of Language Technology
Göteborg, Sweden

October 4, 2006
Abstract

This thesis deals with a kind of vector space model called “Latent Semantic Vector Model”, or LSVM, calculated by the technique “Latent Semantic Indexing”. An LSVM can be used for many things, but I have mainly looked at one direct application: document retrieval. What we can gain from an LSVM is the possibility of searching for content rather than specific keywords. Using an LSVM in a document retrieval system has been shown to improve the quality of the returned document lists, which makes it easier for the user to find the information he or she wants. The problem attacked in this thesis is that an LSVM in the normal case contains just single words, while the terms one searches for in many cases are multi-word expressions.

LSVMs have been trained with various parameter settings for training data, vocabulary, matrix size, context size, and last but not least, different ways to include multi-word expressions directly into the models. The aim has been to determine how the performance of an LSVM changes when we go from a word-based model to a model containing both words and multi-word expressions. To be able to measure the changes, two evaluation methods have been used: synonym tests and document retrieval. Synonym testing has been performed for Swedish and document retrieval for both Swedish and English. The results are improved when multi-word expressions are added for the synonym test task, but change for the worse for document retrieval. For English, the latter change is not significant.

This work has also resulted in two new resources, well suited for evaluation of various models: the evaluation set SweHP560, containing 560 Swedish synonym test queries from "Högskoleprovet", and the new metrics RankEff and WRS for document retrieval evaluation, which handle the problem of an incomplete gold standard in a better way than existing metrics like MAP and bpref.

Acknowledgements

A big thanks to Joakim Nivre, Jussi Karlgren, Per Ahlgren, and Anna Byczynska! Thank you also, all people at School of Mathematics and Systems Engineering at Växjö University (MSI), the National Graduate School of Language Technology (GSLT) and Department of Linguistics at Göteborg University, and all other people that deserve a thanks—especially those who actually are planning to read this thesis.
## Contents

1 Introduction .................................. 1
   1.1 Latent semantic vector models ............ 1
   1.2 Research questions ........................ 2
   1.3 The evaluation experiments ............... 3
   1.4 Choice of languages ....................... 4
   1.5 Outline of the thesis ..................... 4

I Background .................................. 7

2 Models for information retrieval .......... 9
   2.1 The traditional vector space model ....... 9
      2.1.1 Terms and words ...................... 10
      2.1.2 Term weighting ........................ 12
      2.1.3 The Okapi BM25 term weighting scheme ........ 13
      2.1.4 Similarity calculation ................. 13
   2.2 Other models for information retrieval ... 14
      2.2.1 The generalized vector space model .... 14
      2.2.2 Probabilistic information retrieval .... 14
      2.2.3 The extended Boolean model ............ 15
      2.2.4 The neural network model ............... 17
   2.3 Latent semantic indexing ................. 17
      2.3.1 Structure of the training data ........ 19
      2.3.2 Singular value decomposition .......... 21
      2.3.3 Other methods to calculate an LSVM ..... 23

3 The nature of a latent semantic vector model 24
   3.1 Concepts versus dimensions ............... 24
   3.2 How much meaning can be fit into a bag-of-words model? ... 25
   3.3 Multiple meanings of terms in an LSVM .... 26
   3.4 MWUs in an LSVM ........................... 26
   3.5 MWUs in information retrieval ............ 27

4 Latent semantic vector models and applications 30
   4.1 Synonym tests .............................. 30
   4.2 Information retrieval ..................... 31
7.4 Comparison of metrics using made-up examples .................. 67
  7.4.1 The tested metrics ........................................... 67
  7.4.2 Results for made-up examples ............................... 68
7.5 Comparison of metrics using real examples ....................... 70
  7.5.1 Examples with few relevant documents ...................... 70
  7.5.2 Examples with a higher number of relevant documents ...... 70
  7.5.3 Stability for each metric .................................... 70
  7.5.4 Combining the result from several topics .................. 72
7.6 Experimental comparison using log-files from TREC .............. 72
  7.6.1 Test collections ............................................. 74
  7.6.2 Error rates .................................................. 77
  7.6.3 Correlations between system-rankings with
        full pools of relevance data ................................ 79
  7.6.4 Effects of gradually reducing relevance data .............. 81
        7.6.4.1 Change in absolute average values ................... 82
        7.6.4.2 Correlations between system-rankings ............... 83
7.7 Conclusion ........................................................ 86
  7.7.1 Results ...................................................... 86
  7.7.2 Discussion .................................................. 90

8 Interpretation of evaluation results .............................. 91
  8.1 Scale of improvement ........................................... 91
  8.2 Significance testing of document retrieval results ........... 91
  8.3 Significance testing of synonym test results ................. 92
    8.3.1 The \( \chi^2 \) -test ...................................... 93
    8.3.2 McNemar’s test ............................................ 93
    8.3.3 Summary .................................................. 94

III N-grams in Latent Semantic Vector Models ........................ 95
9 Latent semantic vector models enriched with n-grams .............. 97
  9.1 Multi-word units .............................................. 97
    9.1.1 Linguistic multi-word units ............................... 98
    9.1.2 Operationalization of MWUs ................................ 99
      9.1.2.1 Syntactic phrases .................................... 99
      9.1.2.2 Statistical phrases .................................. 100
      9.1.2.3 N-grams ............................................. 101
    9.1.3 Choosing which MWUs to add to the model .................. 101
  9.2 Models .......................................................... 103
    9.2.1 Latent semantic vector models .............................. 104
    9.2.2 Adding n-grams to an L SVM ................................ 104
    9.2.3 Using the enriched L SVM .................................. 105
      9.2.3.1 Lookup in an L SVM enriched with n-grams ........... 107
      9.2.3.2 Lookup in a model composed by L SVMs
            containing n-grams of different lengths ............... 107
11.2.3.1 Degrees of relevance .............................................. 134
11.2.3.2 Statistics for the relevance-judgements .................. 134
11.2.4 Training of new models for document retrieval ........ 135
11.2.5 Overall evaluation results ........................................ 136
11.2.5.1 Corpus choice ............................................... 137
11.2.5.2 Upper and lower case ...................................... 138
11.2.5.3 Adding bigrams and trigrams .............................. 138
11.2.6 Detailed observations ............................................. 139
  11.2.6.1 Comparing a keyword model and the word-based LSVM .... 140
  11.2.6.2 Comparing the word-based LSVM and an LSVM enriched with n-grams \((M_{T3sep})\) .......................... 141
11.2.6.3 Results for different degrees of relevance ............. 142
11.2.7 Conclusion of the Swedish document retrieval results .... 142

11.3 The English test suite for document retrieval .............. 144
11.3.1 The document database ........................................ 145
11.3.2 The topics and relevance-judgements ...................... 145
11.3.3 Training of new LSVMs .......................................... 146
11.3.4 Overall evaluation results ..................................... 147
  11.3.4.1 Corpus choice and number of dimensions ............. 147
  11.3.4.2 Adding bigrams and trigrams ............................ 147
11.3.5 Conclusion of the English document retrieval results . 148

11.4 Summary of the evaluation results for document retrieval . 148

IV Conclusion .......................................................... 149

12 Contributions .......................................................... 151
  12.1 Synonym test evaluation ......................................... 151
    12.1.1 The SweHP560 synonym test evaluation set ............ 151
    12.1.2 Results .................................................... 151
  12.2 Document retrieval evaluation ................................ 152
    12.2.1 New evaluation metrics .................................. 152
    12.2.2 Evaluation results ....................................... 153
  12.3 N-grams in LSVMs ................................................ 154

13 Future research ...................................................... 156
  13.1 Evaluation metrics ............................................... 156
    13.1.1 Different information needs .............................. 156
    13.1.2 Calculating average over RankEff ...................... 156
    13.1.3 Significance testing ...................................... 157
  13.2 LSVMs enriched with n-grams .................................. 157
    13.2.1 Different ways to add MWUs .............................. 157
    13.2.2 Different ways to train the LSVMs ........................ 157
    13.2.3 More testing is needed .................................... 157
List of Tables

2.1 A toy document collection to use to exemplify the co-occurrence data used to build a traditional vector model or an LSVM. 10
2.2 The document×term-matrix extracted from the document collection in table 2.1. 11
2.3 The term×term-matrix extracted from the document collection in table 2.1 using a context size of ±2 words. 21
4.1 Some example documents used in section 4.4 35
4.2 Top-10 words closest to the example document $d_1$ 36
4.3 Top-10 words and bigrams closest to $d_2$ in a mixed model of words and $n$-grams 36
4.4 Some main words from an example thesaurus automatically generated from TREC-data 37
5.1 Ranking lists from four hypothetical systems that demonstrate one problem with $R$-precision 44
6.1 Share of queries and alternatives with different lengths in SweHP560 51
6.2 Overview of query and alternative lengths in SweHP560 52
7.1 Illustrative example of a $RankEff$ calculation 60
7.2 Rankings associated with two hypothetical systems where the resulting ranking lists are identical except for one relevant document 63
7.3 Rankings associated with two hypothetical systems with the same number of relevant and irrelevant documents 64
7.4 Some made-up examples that should test the behaviour for the different evaluation metrics 68
7.5 Results for the made-up examples 69
7.6 Some example topics with very few relevant documents 71
7.7 Results for the real example topics 72
7.8 Some example topics with relatively few relevant documents 73
7.9 Some example topics with a high percentage of relevant documents 74
7.10 Overview of the TREC document collections and topic sets 75

ix
7.11 Averages for the number of retrieved, relevant, and unjudged documents over all systems and TREC 77
7.12 Averages for the top-100 retrieved documents over all systems and topics 78
7.13 Experimentally extracted significant differences for each metric in TREC-8 calculated for $z = 25$, i.e. 50 topics. 78
7.14 Experimentally extracted significant differences for each metric in TREC-10 calculated for $z = 25$, i.e. 50 topics. 79
7.15 Experimentally extracted significant differences for each metric in TREC-12 calculated for $z = 50$, i.e. 100 topics. 79
7.16 Kendall correlations between system-rankings for each metric 80

8.1 Answers from two hypothetical LSVMs $M_1$ and $M_2$ for $n$ synonym queries 92
8.2 Score and expected value for two hypothetical LSVMs $M_1$ and $M_2$, based on the 1120 observations 93
8.3 $\chi^2$-values for some useful significance levels 93
8.4 Number of correct answers for two hypothetical LSVMs $M_1$ and $M_2$, based on the 560 queries 94

9.1 The different LSVMs used to find out how $n$-grams change the performance 105
9.2 The preprocessed version of the document in figure 9.1 for each model sent to the Infomap NLP System 106

10.1 Results for the two case options 115
10.2 Results for different corpora 116
10.3 Results for different $Vocab$-values 117
10.4 Results for different $CoVoc$-values 118
10.5 Results for different $Dim$-values 119
10.6 Results for different $Con$-values 120
10.7 Results after preprocessing for the Bring+Lexin corpus 121
10.8 $p$-values from the McNemar test between $M_W$, $M_{T2}$, and $M_{T3}$ 121
10.9 The share of each difference type between $M_W$ and $M_{T3}$ 122
10.10 The cases of single word queries and answers where the answers from $M_W$ and $M_{T3}$ differ 124
10.11 The cases of single word queries and MWU answers where the answers from $M_W$ and $M_{T3}$ differ 125
10.12 Table 10.11, continued 126
10.13 The cases of MWU and single word queries answers where the answers from $M_W$ and $M_{T3}$ differ 127
10.14 The cases of MWU queries and MWU answers where the answers from $M_W$ and $M_{T3}$ differ 128
10.15 Number of improvements for each model depending on queries/answer type, comparison of $M_W$ and $M_{T3}$ 129
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.16</td>
<td>Number of improvements for each model depending on queries/answer type, comparison of $M_{T2}$ and $M_{T3}$</td>
<td>129</td>
</tr>
<tr>
<td>11.1</td>
<td>Overview of the Swedish document database</td>
<td>132</td>
</tr>
<tr>
<td>11.2</td>
<td>Four level relevance scale (Sormunen, 2002)</td>
<td>135</td>
</tr>
<tr>
<td>11.3</td>
<td>Average number of relevant or irrelevant documents per topic</td>
<td>136</td>
</tr>
<tr>
<td>11.4</td>
<td>Overview of relevance-judgements for the GPHD data</td>
<td>137</td>
</tr>
<tr>
<td>11.5</td>
<td>Overview of relevance-judgements for the TT data</td>
<td>137</td>
</tr>
<tr>
<td>11.6</td>
<td>Overview of relevance-judgements for the full data set where GPHD and TT are merged together</td>
<td>138</td>
</tr>
<tr>
<td>11.7</td>
<td>Overview of subcorpora for Swedish document retrieval</td>
<td>138</td>
</tr>
<tr>
<td>11.8</td>
<td>Results for subcorpora of various sizes</td>
<td>139</td>
</tr>
<tr>
<td>11.9</td>
<td>Comparison of results for lower case and case sensitive models</td>
<td>139</td>
</tr>
<tr>
<td>11.10</td>
<td>Comparison of results for word-based and n-gram enriched models</td>
<td>140</td>
</tr>
<tr>
<td>11.11</td>
<td>Evaluation of the models based on separate LSVMs for each n-gram length</td>
<td>140</td>
</tr>
<tr>
<td>11.12</td>
<td>$RankEff$ difference between two models calculated per topic</td>
<td>141</td>
</tr>
<tr>
<td>11.13</td>
<td>Individual document movement for $M_W$ compared to $M_{Key}$, for relevant documents in topic T20</td>
<td>141</td>
</tr>
<tr>
<td>11.14</td>
<td>Individual document movement for $M_W$ compared to $M_{Key}$, for all documents in all pools</td>
<td>142</td>
</tr>
<tr>
<td>11.15</td>
<td>$RankEff$ difference between $M_W$ and $M_{T3sep}$ calculated per topic, sorted by relative difference</td>
<td>144</td>
</tr>
<tr>
<td>11.16</td>
<td>Average scores for each degree of relevance</td>
<td>145</td>
</tr>
<tr>
<td>11.17</td>
<td>Overview of the English document database TREC812</td>
<td>145</td>
</tr>
<tr>
<td>11.18</td>
<td>Overview of subcorpora for English document retrieval</td>
<td>146</td>
</tr>
<tr>
<td>11.19</td>
<td>Results for subcorpora of various sizes</td>
<td>147</td>
</tr>
<tr>
<td>11.20</td>
<td>Results for $M_W$, $M_{T2}$, and $M_{T3}$</td>
<td>148</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Two terms in the extended Boolean model 16
2.2 Layout of a neural network IR-model 18
4.1 The hand of God goal 37

7.1 Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-12 55
7.2 Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-8 56
7.3 Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-10 57
7.4 Translated version of topic C137 from CLEF 65
7.5 Scores for a reduced ranking list of topic T72—a topic with a medium share of relevant documents 75
7.6 Normalized scores for a reduced ranking list of topic T72—a topic with a medium share of relevant documents 76
7.7 Normalized scores for a reduced ranking list of topic T7—a topic with a high share of relevant documents 76
7.8 Normalized scores for a reduced ranking list of topic T58—a topic with a few known relevant documents 77
7.9 Average values for each metric and system in TREC-8, ordered by MAP values 81
7.10 Average values for each metric and system in TREC-10, ordered by MAP values 82
7.11 Average values for each metric and system in TREC-12, ordered by MAP values 83
7.12 Average values for each metric over all systems in TREC-8 when relevance data are reduced 84
7.13 Average values for each metric over all systems in TREC-10 when relevance data are reduced 85
7.14 Average values for each metric over all systems in TREC-12 when relevance data are reduced 86

xii
7.15 Kendall correlation values for system-rankings between full relevance pools and reduced pools for TREC-8

7.16 Kendall correlation values for system-rankings between full relevance pools and reduced pools for TREC-10

7.17 Kendall correlation values for system-rankings between full relevance pools and reduced pools for TREC-12

9.1 A tokenized version of the example document presented in section 2.1.1

10.1 Four answer combinations between $M_W$ and $M_{T3}$

11.1 Topic C098 from CLEF

11.2 Topic C102 from CLEF

11.3 Topic C118 from CLEF

11.4 RankEff for different degrees of relevance for each topic

11.5 $bpref$-10 for different degrees of relevance for each topic
Chapter 1

Introduction

Latent semantic indexing (LSI), presented in Deerwester et al. (1990) and Dumais et al. (1988), is a statistical method for extracting semantic information hidden in the structure of a document collection, using co-occurrence statistics. The result is a vector representation for all the words and documents, which may be used to calculate similarities between any documents or words. In this thesis, I perform experiments on various methods to include multi-word units (MWUs) during the training of the latent semantic vector model (LSVM), to make it possible to look up not just single words but also MWUs in the LSVM. Evaluation is a very important part of this investigation, not only as a tool during the experiments, but also in the sense that I present new resources and metrics for evaluation of LSVMs and information retrieval systems in general. I use two different applications for evaluation: synonym tests and document retrieval.

1.1 Latent semantic vector models

The LSVM is calculated from co-occurrence statistics for a document collection. This can be done in many different ways, but the main idea is to think of the original co-occurrence representation as an extremely high-dimensional vector space, just as the traditional vector model used in information retrieval. Each word stands for one dimension and all dimensions are orthogonal to each other. The vector for a document is created as the vector sum of the words it contains. The problem with this representation is that it contains one dimension for each word, even if some different words may have a very similar meaning. So what matters in the traditional vector model is the exact set of keywords a query or document contains. To go from a traditional vector space model to an LSVM is to create a projection from the extremely high number of dimensions to a number that better fits the conceptual space. The more narrow space, i.e. the lower number of dimensions, in the LSVM will make it possible to see similarities between different keywords, since they are no more orthogonal, based on the document structure created by the co-occurrences of terms.

Note that what we find in the documents are sequences of words (or tokens) that sometimes are equivalent to terms, but in many other cases
are just parts of multi token terms.

The normal way to train an LSVM is to use a co-occurrence matrix containing just single words as input. This means that the contexts are seen as bags of words, and a lookup in the word-based vector space for an MWU is actually a set of lookups of each word included in the MWU, which are then added together. The vector addition of semantic vectors is often not able to catch the real meaning of MWUs. Idioms, such as “kick the bucket” and “Big Apple”, are good examples of difficult cases. The motivation for including MWUs in an LSVM is simply to get a model, where we directly can search for compounds, multi-word full names and other MWUs, which may give better results for some of the applications using the LSVM, while we retain all the positive properties. In document retrieval, for example, an LSVM has been shown to improve both recall and precision compared to a keyword search baseline (Ding, 2005). But since there are no natural ways to index MWUs in an LSVM, the user loses the possibility to search for them, which is possible in most traditional search engines by using quotation marks around the MWU. Another case where MWUs may improve results is a keyword extractor based on an LSVM. If the model contains words and MWUs, the extractor will also find multi-word terms and not just single words. Hulth (2004) shows that less than 14% of all manually selected keywords contain just one token. The same thing applies if we are using the LSVM to automatically build a domain specific thesaurus based on a corpus.

1.2 Research questions

There are many ways to include MWUs in an LSVM. One of the most straightforward is to just create a new version of the training corpus that contains the single words but also $n$-grams, i.e. word sequences of length up to some bound $n$. In this thesis I will investigate to what extent MWUs can be captured by $n$-grams in this sense. The basic question I will pose is:

How will the performance of an LSVM change when MWUs are added?

There is no easy answer to this question since we have to consider many preconditions and options:

- How should we add the MWUs?
- In what sense or in which applications do we want to measure the changes in performance and how should they be measured?
- What is the scale of improvement?

Let me comment on these points briefly. I have tried to build the MWU-enriched LSVMs in basically two ways:
1.3 The evaluation experiments

- Preprocess the corpus to include both words and MWUs and train an LSVM including all of them, i.e. single words, word pairs, and word triples.

- Train separate sub-LSVMs for single words, word pairs, and possibly triples. Similarities are then calculated for each sub-model and weighted together to one value.

The way I select which MWUs to add to a model is investigated more deeply later in the thesis, but I am trying to keep things simple, in order to hold down the processing times and also as long as possible keep the algorithm language-independent.

The easiest scale of improvement would be just to compare the overall evaluation values for a word-based LSVM to one with MWUs included. That will be done, but I also want to go a little bit deeper and look separately at tasks in the evaluation data where MWUs are clearly involved. We will then get a more complex scale. It could for example be the case that we get an improvement when MWUs are involved, but not overall.

1.3 The evaluation experiments

When we have implemented training of LSVMs enriched with $n$-grams, it is important to determine whether there is an improvement for examples containing MWUs, but also to check whether performance is degraded compared to the word-based model for queries or tasks containing just single words. I will therefore build LSVMs with various parameter settings, with and without $n$-grams included. Then, these different LSVMs will be carefully evaluated using two kinds of evaluation data:

- **A multiple choice synonym test**
  A set of queries each containing a word (sometimes multi-word expressions) and five alternatives where exactly one of them is a synonym. The evaluation metric here is more straightforward: I will use the percentage of correct answers. The queries I use were originally used to test how well students were prepared for university studies, and they are all in Swedish.

- **A traditional document retrieval test suite**
  A set of topics (queries) and a document collection: For each topic there are manual relevance-judgements for a number of documents. In section 7.2.3 I present the evaluation metric RankEff (Ranking Effectiveness) which seems to perform better than the more used MAP (see section 5.3.3.3). I have used the same kind of data both for Swedish and English.

The various ways to include the $n$-grams in the LSVM will also affect the running time and RAM-memory usage, which will force us to keep both
1 Introduction

memory and time consumption in mind all the time. Computers get more powerful, but at the same time, the data sources handled by information retrieval systems get bigger, and IR is one of the most important applications for LSVMs.

There is a need for better evaluation metrics for the document retrieval case and also a data collection for the synonym test evaluation. Two important contributions in this thesis are:

- Compilation of the multiple choice synonym test. It contains 560 queries taken from a Swedish synonym test which is a part of “Högskoleprovet” (an entrance test for university studies in Sweden).

- The definition and testing of the new metric for ranking effectiveness, $\text{RankEff}$, which is of general interest for most document retrieval systems, for example in the TREC and CLEF environments.¹

1.4 Choice of languages

First of all I would like to stress that the training of an LSVM is a mathematical process and totally language-independent as long as the word boundaries are easy to find. Therefore, I try to keep other parts, like tokenization and preprocessing, language-independent as well. This will enable the models to be tested for other languages more easily.

I have chosen to work both on English and Swedish since the importance of MWUs may be very different in these two languages. All methods used are language-independent, i.e. I do not use resources like lexica or annotated corpora. There are plenty of published experiments in information retrieval and also on LSI for English, so I think it will be interesting to look at the much smaller language Swedish, which also has a somewhat richer morphology than English, as a complement. To run the evaluations on English data is still important for me to be able to compare my work to that of other researchers. In a wider perspective, Swedish and English are very similar, but I do not have the time, knowledge, or resources to investigate other languages that are more different than English and Swedish.

1.5 Outline of the thesis

The remainder of this thesis consists of four parts (I-IV) where part I, containing chapter 2-5, forms the background. Chapter 2 introduces the traditional vector space model which is the base for the latent semantic vector model (LSVM) which I have used in this work. The nature of an LSVM is then described in chapter 3. In chapter 4 the focus is on applications where the LSVM can be used: information retrieval, document clustering,

¹TREC (Text REtrieval Conference) is an annual information retrieval conference and competition, arranged by NIST (National Institute if Standards and Technology).
synonym tests, keyword extraction, and automatic thesaurus extraction. In chapter 5 I report on evaluation of LSVMs. The Cranfield evaluation framework has led to the document retrieval evaluations made in the TREC environment, and I also introduce the most widely used evaluation metrics.

Part II (chapter 6-8) is about evaluation methods and data collections used for evaluation. Chapter 6 presents a new evaluation set for Swedish that can be used to evaluate an LSVM. It consists of 560 queries from Swedish synonym tests including the correct answers. Chapter 7 introduces my new evaluation metric \textit{RankEff}, and investigates how well \textit{RankEff}, and other metrics like \textit{bpref} and \textit{MAP}, rank systems in earlier TREC competitions, and how they handle the problem with uncomplete evaluation data. I also study some theoretical aspects of the metrics, including a discussion of what we want to measure in an environment like TREC, or in the real world, and whether the metrics really give us this. Finally, in this second part, chapter 8 is a discussion of how to interpret evaluation results, including significance testing.

In part III (chapter 9-11) I introduce multi-word units (MWUs), both from a linguistic point of view and the more computationally oriented definitions (section 9.1). I also take a look at earlier research on how MWUs have been used in information retrieval and related fields. I argue that at this initial stage, the best way is to explore the performance for models containing both the words and all occurring \(n\)-grams up to a certain length added, and provide the necessary formal definitions of the IR-models used in chapter 9. Chapter 10 and 11 report a set of experiments aiming to investigate how well different LSVMs enriched with \(n\)-grams function compared to standard LSVMs based on words. For the two evaluation methods: synonym test (chapter 10) and document retrieval (chapter 11), I test LSVMs trained with different parameter settings for training corpus, matrix size, and dimensionality, and also the different ways to include the \(n\)-grams in the model. The document retrieval task is evaluated both for English and Swedish, but the synonym test task only for Swedish. The results are mixed. For the synonym test task we get a significant improvement when \(n\)-grams up to length 3 are added, compared to word-based models. LSVMs trained in the same way unfortunately perform worse on the document retrieval task. For Swedish, the worsening is noticeable but for English the results are similar for word-based and \(n\)-gram enriched models.

Part IV (chapter 12-13) contains a summary of the results and a chapter on proposed future work. This thesis raises many ideas for future work because there are so many different experiments to perform before being able to finally conclude how MWUs should be added to an LSVM to get a clear improvement for all applications.
Part I

Background
Part I: The background part

This part gives the needed background information.

Chapter 2
Information retrieval (IR) and the models used for IR are essential for this work, so I will give introductions to the field. More specific information about traditional vector space models can be found in section 2.1 and some other models for information retrieval in section 2.2. The latent semantic vector model (LSVM), which is a more refined vector space model, is presented in section 2.3.

Chapter 3
This chapter is an attempt to go a bit deeper in the understanding of what kind of information an LSVM contains. We can see that an LSVM is a powerful and flexible information source.

Chapter 4
Chapter 4 shows how LSVMs can be used in some different applications.

Chapter 5
The last chapter of this part is about evaluation of LSVMs. The two evaluation tasks used in this thesis are presented in section 5.2 for synonym test evaluation and section 5.3 for document retrieval based evaluation.
Chapter 2

Models for information retrieval

This chapter will present some different models for information retrieval (IR), but the focus will be on the traditional vector space model, which is historically interesting and also still used in many IR applications, and the latent semantic vector model including the singular value decomposition (SVD) which is my choice of how to calculate the LSVM to use in applications like document retrieval.

2.1 The traditional vector space model

Vector space models are used in information retrieval as a way to go from Boolean keyword search to something closer to a subjective best match search.\(^1\) The vector space model was presented in the seventies by Gerald Salton—one of the pioneers in information retrieval (Salton, 1971). In a document collection \(D\), a document \(d_j \in D\) \((1 \leq j \leq m)\) is represented as a vector of weights:\(^2\) 
\[
d_j = (w_{1,j}, ..., w_{n,j}) \text{, where each value corresponds to a term } w_i \text{ in the vocabulary of size } n.\]

The weight \(w_{i,j}\) is calculated based on the frequency and distribution of \(w_j\) in the document collection, which is then represented as an \(m \times n\) matrix (see equation 2.1).

\[
D = \begin{pmatrix}
w_{1,1} & w_{2,1} & \cdots & w_{i,1} & \cdots & w_{n,1} \\
w_{1,2} & w_{2,2} & \cdots & w_{i,2} & \cdots & w_{n,2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
w_{1,j} & w_{2,j} & \cdots & w_{i,j} & \cdots & w_{n,j} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
w_{1,m} & w_{2,m} & \cdots & w_{i,m} & \cdots & w_{n,m} 
\end{pmatrix}
\]

(2.1)

To get an idea of how a document collection matrix \(D\) may look like, a small example of a document collection is shown in table 2.2. Each cell in the table shows the frequency for the term in document \(d_i\). This is a truncated\(^2\) version of the document \(\times\) term-representation of the small document collection in table 2.1 containing nine short documents \(d_1..d_9\).

\(^1\)The task I am addressing here is document retrieval, i.e. the task of finding a ranked list of the most relevant documents to an information need expressed by a query.

\(^2\)Words with a global frequency below 2 are not included.
Table 2.1: A toy document collection to use to exemplify the co-occurrence data used to build a traditional vector model or an LSVM.

<table>
<thead>
<tr>
<th>Id</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>advise strongly regardless age see doctor checkup perhaps cardiovascular stress test start running begin new exercise program</td>
</tr>
<tr>
<td>$d_2$</td>
<td>running speed runner moves calculated multiplying cadence steps second stride length</td>
</tr>
<tr>
<td>$d_3$</td>
<td>doctor need tell planning start running training plan designed beginners copy plan show doctor</td>
</tr>
<tr>
<td>$d_4$</td>
<td>bradley deduced starlight falling earth should appear come slight angle calculated comparing speed earth orbit speed light</td>
</tr>
<tr>
<td>$d_5$</td>
<td>attempts measure speed light played important part development theory special relativity speed light central theory</td>
</tr>
<tr>
<td>$d_6$</td>
<td>strongly suggest friend start exercise athletic program discuss doctor diabetes educator first course once moving feel physically better</td>
</tr>
<tr>
<td>$d_7$</td>
<td>start programs really need screen look bit more bland will speed system noticeably</td>
</tr>
<tr>
<td>$d_8$</td>
<td>recruit doctor support exercise program minimize injuries</td>
</tr>
<tr>
<td>$d_9$</td>
<td>light exercise relieves anxiety breast cancer survivors after non-exercisers finished exercise program second 10 weeks</td>
</tr>
</tbody>
</table>

2.1.1 Terms and words

In this chapter, I write about documents, terms, and term weights just as most other IR researchers. The term *term* is however worth some reflection. Normally we have a collection of documents represented as text that we want to index to achieve better information retrieval, but the “real” terms are not marked in this representation. The text representation is a sequence of characters and it is not even trivial to find the words, and definitely not the terms that we want indexed by the search engine. What we can do without too much trouble is to use whitespace characters and non-alphanumeric characters to tokenize the text into a sequence of words and tokens that are not words, i.e. punctuation marks. Among these words there are typical function words like *and, I or on* that we do not want to index. The function words are a more or less closed set which makes it easy to filter them out, just by using a stop-list.

1. If we start with a small example, the string:

   Bill Clinton, the former president of the United States of America, likes computer science.

2. If we tokenize this example, we get a sequence of 17 tokens:
Table 2.2: The document×term-matrix extracted from the document collection in table 2.1.

<table>
<thead>
<tr>
<th>Term</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
<th>$d_7$</th>
<th>$d_8$</th>
<th>$d_9$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>light</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>strongly</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>doctor</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>running</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>speed</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>need</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>program</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>calculated</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>second</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>theory</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>exercise</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>plan</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>earth</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>17</td>
<td>11</td>
<td>14</td>
<td>17</td>
<td>15</td>
<td>18</td>
<td>13</td>
<td>7</td>
<td>15</td>
<td>127</td>
</tr>
</tbody>
</table>

3. Now, the list of words is the same sequence but the non-words have to be removed to obtain a list of 14 words:

   Bill, Clinton, ',', the, former, president, of, the, United, States, of, America, ',', likes, computer, science, '.'

4. To get something that could be used practically in a search engine we need to remove the stop-words to get a list of something we can call search terms (10 elements):

   Bill, Clinton, former, president, United, States, America, likes, computer, science

5. A more advanced definition of search term would force us to do some kind of shallow parsing to identify that Bill without Clinton is not the same. In most cases we want to be able to search for the unit Bill Clinton instead of the separate words. This more elaborated definition of search term would yield the following sequence of 5 search terms:

   “Bill Clinton”, “former president”, “United States of America”, likes, “computer science”
Many systems stop at step 4 since the fifth step takes more processing time, and all such programs fail in some cases. The user simply has to stick to the single-word search terms.

### 2.1.2 Term weighting

One popular way to calculate \( w_{i,j} \) mentioned earlier in section 2, is called the tf-idf weight (term frequency–inverse document frequency).

\[
  w_{i,j} = f_{i,j} \times \log \frac{N}{n_i} \quad \text{where} \quad f_{i,j} = \frac{\text{freq}_{i,j}}{\max_k \text{freq}_{k,j}}
\]  

(2.2)

The definition (equation 2.2) can be found in Baeza-Yates and Ribeiro-Neto (1999) where \( \text{freq}_{i,j} \) is the number of occurrences of the term \( w_i \) in document \( d_j \), \( n_i \) is the number of documents in which \( w_i \) occurs, and \( N \) is the total number of documents. A higher frequency for \( w_i \) in a document \( d_j \), or a lower \( n_i \), i.e. \( w_i \) occurs in fewer documents, will give a higher weight. There are more or less sophisticated variants of the tf-idf formula, but the basic ideas are the same:

- A term is more important for a document if it occurs many times in that document
- A term is less important if it occurs in many documents

Except for these general ideas the term weighting formula should handle special cases like very low frequencies, very long or very short documents, etc.

A query \( q \) is modeled in the same way as a document: \( q = (w_{1,q}, \ldots, w_{n,q}) \), as an n-dimensional vector, and query terms not in the document vocabulary are not used since they are not in the vector space. Query terms may be weighted as well with for example equation 2.3 (Salton and Buckley, 1988). Since a query is seen as a special case of a document, the same term weighting formula could be used but there are differences between documents and queries, both in length, type, and of course their function, so a slightly different formula is often used.

\[
  w_{i,q} = f_{i,q} \times \log \frac{N}{n_i} \quad \text{where} \quad f_{i,q} = \frac{1}{2} + \frac{\text{freq}_{i,q}}{2\max_k \text{freq}_{k,q}}
\]  

(2.3)

\( \text{freq}_{i,q} \) is the frequency of \( w_i \) in \( q \) and \( \max_k \text{freq}_{k,q} \) is the maximum frequency for any term in \( q \). For short queries, where the frequency for all terms is 1, the leftmost part of equation 2.3 \( (f_{i,q}) \) will also be 1, leading to the simpler formula \( w_{i,q} = \log \frac{N}{n_i} \).
2.1 The traditional vector space model

2.1.3 The Okapi BM25 term weighting scheme

The most used weighting scheme since it was introduced at TREC-3, is the Okapi BM25 function and variations of it. BM25 was developed as a part of the City University of London’s Okapi IR system, starting with the first version: BM1 (Best Match 1). A combination of BM11 and BM15 became BM25 at TREC-3 (Robertson et al., 1995, 1996). For a query \( q \) and a specific document \( d \) of length \( dl \) (\( avdl \) is the average) I give a definition of the BM25 score in equation 2.4. \( w^{(1)} \) (defined in equation 2.5) is the Robertson-Sparck Jones weight for the term \( t \) in query \( q \) (Robertson and Sparck Jones, 1976). \( N \) is the number of documents in the collection, from which \( n \) contains the current term \( t \). \( R \) is the number of known relevant documents and \( r \) of these contain \( t \). \( tf \) is the frequency of \( t \) in \( d \), and \( qtf \) is the frequency of \( t \) within the topic from which \( q \) was derived. \( k_1, ..., k_3 \) and \( b \) are tuning parameters depending on the database, set to different values by different systems. To get an idea of how the formula works, typical settings for systems competing in TREC-4 were 8 for \( b \), 1.0-2.0 for \( k_1 \), 0.6-0.75 for \( k_3 \), and 0 for \( k_2 \). The BM25 formula is complicated but we can see in the formula that \( k_2 > 0 \) will lead to higher scores for short documents, which in some cases, for example in web page retrieval, is a desired feature.

\[
\sum_{t \in q} w^{(1)} \frac{tf(k_1 + 1) qtf(k_3 + 1)}{K + tf} + k_2 q |avdl - dl| \frac{avdl}{avdl + dl} \tag{2.4}
\]

\[
K = k_1(1 - b + b \frac{dl}{avdl})
\]

\[
w^{(1)} = \log \left( \frac{(r + 0.5)(N - n - R + r + 0.5)}{(n - r + 0.5)(R - r + 0.5)} \right) \tag{2.5}
\]

2.1.4 Similarity calculation

The reason for all the efforts on term weighting is of course in order to be able to find out relevant documents according to queries or example documents. Most search engines give a list of the \( n \) most relevant documents, and in some cases a similarity score for each returned document. Similarity values (equation 2.6) may be calculated between documents and queries using the cosine between their vectors. This value will be 1 if the vectors \( d_j \) and \( q \) are parallel, i.e. have the same direction. If they are orthogonal (the angle between them is 90°), the cosine will be 0. A typical document retrieval task will be to calculate the similarity between the query \( q \) and all documents \( d_j \), and then select the wanted number of documents with the highest similarity scores.

\[
sim(d_j, q) = \frac{d_j \cdot q}{|d_j| \times |q|} = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \tag{2.6}
\]
2 Models for information retrieval

If we just want to rank the documents in D according to relevance to the query $q$, and D is normalized such that $\forall d_j \in D : |d_j| = 1$, we could just calculate the vector $s = q^T D$ to get a vector of similarity scores between $q$ and each document. This vector is then sorted to get a list of documents ordered by relevance to $q$.

2.2 Other models for information retrieval

There are many other models for document retrieval and relevance ranking. Baeza-Yates and Ribeiro-Neto (1999) mention, among others, the generalized vector space model, the probabilistic model, an extended Boolean model, and the neural network model. Very briefly the key ideas are presented in the following sections.

2.2.1 The generalized vector space model

The difference from the traditional vector space model (section 2.1) is that in the generalized version, there is no assumption of independence between index terms. Each term is not seen as a vector in the vector space, but instead a linear combination of the base vectors extracted from the document collection. The main advantage of having non-orthogonal base vectors is that the training of the model may be implemented more efficiently. Read more about usage and training of such a model in, for example, Wong et al. (1985) or the more recent Liu, Zhang, Yan, Yang, Yan, Chen, Bai and Ma (2004). Both the traditional vector space model and the latent semantic vector space model are special cases of the generalized vector space model.

Mathematically, a vector model based on orthogonal base vectors is easier to understand than a generalized vector space model, since we normally think of dimensions as orthogonal. On the other hand, a generalized vector space model with non-orthogonal base vectors may be trained in a way that makes the vectors actually stand for something meaningful since we are not bound to orthogonal base vectors. In an LSVM based on SVD, two terms or documents corresponding to two different base vectors will always have a similarity value of zero since base vectors are orthogonal.

2.2.2 Probabilistic information retrieval

The key idea is that retrieval, ranking, and similarity calculations are based on probability theory—probabilities will decide which documents a retrieval system returns. The most well known probabilistic model is the binary independence retrieval model introduced by Robertson and Sparck Jones (1976). Binary stands for the fact that the document vectors are built up of binary values. For the document vector $x = x_1, ..., x_n$ where $x_i = 1$ if

---

3To get the real similarities from equation 2.6, we should also divide the relevance scores by $|q|$.
the $i$th term occurs in the document modeled by $x$, otherwise $x_i = 0$. Independence because terms are assumed to occur in documents independently. The independence-assumption makes it much easier to calculate estimates of the probabilities. For a given query $q$ the goal is to rank the document in a collection $D$ based on the probability of each document $d_j$ being relevant to $q$. Let us define $R(q)$ as the set of documents relevant to $q$, we then define the similarity in equation 2.7. The formula can be interpreted as saying that the similarity is the ration between the probability that a given document $d_j$ is an element in the set of documents relevant to $q$ and divided by the probability that $d_j$ is not an element in the set of documents relevant to $q$.

$$sim(d_j, q) = \frac{P(d_j \in R(q))}{1 - P(d_j \in \bar{R}(q))}$$

(2.7)

To calculate $sim(d_j, q)$, the normal way is to assume independence between index terms. That assumption together with Bayes’ rule and the fact that $P(R)$ and $P(\bar{R})$ are constant for all documents, end up with a similarity score containing probabilities of the types $P(k_i|R)$ and $P(k_i|\bar{R})$, i.e. the probability for $k_i$ to occur in a document randomly selected from $R$ and $\bar{R}$ respectively. The details for deriving the similarity formula can be found in Baeza-Yates and Ribeiro-Neto (1999).

Since $R$ is unknown to some extent, the estimations of $P(k_i|R)$ and $P(k_i|\bar{R})$ are difficult to make. Another negative aspect with the model is that the frequencies for the index terms inside documents are not taken into account since the term weights are binary. On the other hand, it is intuitively appealing that the document’s ranking is based on the probability of being relevant. There are other variants of probabilistic IR-models based on for example Bayesian networks or inference networks, but I will not go into that here.

### 2.2.3 The extended Boolean model

In many earlier information retrieval and bibliographic systems, the Boolean model was used. The model is based on set theory and Boolean algebra, so a query is built up by keywords and operations. Each keyword stands for the set of documents containing the word, and the operations make it possible to form unions, intersections, and complements. The strength of the Boolean model is that it is easy to understand why a document $d_1$ is retrieved and another document $d_2$ is not—just look at the logical expression that forms the query. A Boolean expression will always be True or False, and in this case it was True for $d_1$ and False for $d_2$. The main objection against the Boolean model is that in the model there is no such thing as a relevance value except for the binary value. Therefore Salton et al. (1983) presented an extended version of the Boolean model which includes the features of partial matching and different weights for different terms. Let us think of two simple queries $q_{or} = k_x \lor k_y$ and $q_{and} = k_x \land k_y$ calculated with tf-idf.
for the documents $d_j$ and $d_{j+1}$, figure 2.1 shows the two terms as a two dimensional space where the value in each dimension ($k_x$ on the x-axis and $k_y$ on the y-axis) goes from 0 to 1.

Figure 2.1: Two terms in the extended Boolean model

A reasonable similarity metric $sim(q_{or}, d_j)$ is the distance from the point $(0, 0)$ to $d_j$ (see equation 2.8) since $(0, 0)$ is clearly the worst point, and for $sim(q_{and}, d_j)$ we have to use the 1 minus the distance from $(1, 1)$ (see equation 2.9)) instead since $(1, 1)$ is clearly the best point for $q_{and}$ and we want the score to go from 0 (worst) to 1 (best).\(^4\)

\[ sim(q_{or}, d_j) = \frac{\left( w_{a,j}^p + w_{b,j}^p \right)^{1/p}}{2} \]  

\[ sim(q_{and}, d_j) = 1 - \left( \frac{(1 - w_{a,j})^p + (1 - w_{b,j})^p}{2} \right)^{1/p} \]  

\(^4\)The reasoning for $and$ compared to $or$ is different since there is no “best point” for $q_{or}$ in the original Boolean model since $(0, 1)$ and $(1, 0)$ are just as good as $(1, 1)$, or “worst point” for $q_{and}$ ($(0, 1)$ and $(1, 0)$ are just as bad as $(0, 0)$).
2.3 Latent semantic indexing

\[ sim(q, d_j) = \left( \frac{1 - \left( \frac{(1-w_{a,j})^p + (1-w_{b,j})^p}{2} \right)^{1/p} + w_{c,j}}{p} \right)^{1/p} \]  

(2.10)

Note that for \( q_{or} \), we will not get an optimal score for \( d_j \) when \( w_{a,j} = 1 \) and \( w_{b,j} = 1 \), which is the case for the original Boolean model. Let us set \( p = 2 \) and insert the values:

\[ sim(q_{or}, d_j) = sim(q_{or}, \{1, 0\}) = \left( \frac{1^2 + 0^2}{2} \right)^{1/2} = \frac{1}{\sqrt{2}} \approx 0.71 \]

The full power of the extended Boolean model has not been used very much since it was introduced in 1983, but it has a theoretical interest since many other models may be expressed in it.

### 2.2.4 The neural network model

The idea of how to use neural networks as a machine learning method was inspired by the way a human brain works. Billions of neurons are connected in a network, and based on the signals they receive, they emit output signals to other neurons. The simplified versions implemented in a computer have much less neurons than a human brain. The synaptic connections between the neurons in our brains can vary in strength over time, which is emulated in the computational model by putting variable weights on each connection. Neural networks have been shown to be very good for pattern recognition, so how can we make use of this in an information retrieval context? One way, proposed by Wilkinson and Hingston (1991), is to have one layer of neurons for the query terms—one neuron for each term, another layer with one neuron for each document, and between these two, a layer with one neuron for each document term as in figure 2.2. The query term neurons initiate the training process by sending signals to the document term nodes, and these nodes may now send new signals to the document nodes. As a response, the document nodes may send signals back to the document term nodes, and so on. The signals get weaker and weaker until the final iteration. What is happening here is that after several bounces between document term neurons and document neurons, there may have been signals from one query term neuron to documents actually not containing the corresponding term. The effect is in this way similar to the chain mechanism obtained by LSI: We may find relevant documents, that contain none of the query terms in the query.

### 2.3 Latent semantic indexing

Instead of going deeper into various information retrieval models, I will continue with an improved vector space model—the latent semantic vector
model. LSI (Latent Semantic Indexing)\(^5\) is an attempt to overcome an important weakness in the traditional vector space model (Dumais et al., 1988; Deerwester et al., 1990). We want to search for concepts rather than terms or (even worse) just words. Since the similarity in a traditional vector space model is calculated using the dot-product between the query vector \(q\) and the document vector \(d_j\), the only terms that count are the ones both in \(q\) and \(d_j\). Synonyms to the query terms will not count at all. The other main weakness with the traditional vector space model, polysemy (or ambiguity), is not dealt with directly by LSI. A typical ambiguous word like *swallow* will get one single vector which mirrors the contexts of all occurrences of the word form, regardless of the meaning in each context.\(^6\)

\(^5\)LSI is sometimes called LSA (Latent Semantic Analysis), but as far as I have seen the terms are used as synonyms. Maybe some researchers want to stress that they are using it for something else than the traditional indexing in IR-systems. I will use LSI regardless if we are talking about indexing or some other aspect of the method.

\(^6\)See section 3.3 for Thomas Landauer’s explanation of how LSI can handle ambiguity (Landauer et al., 1997).
2.3 Latent semantic indexing

LSI uses a vector space, just as the traditional model, but the dimensions are not the same. Instead of one dimension for each term, the LSI model dimensions are combinations of terms, and the latent semantic vector space ($S_{LSI}$) normally contains much fewer dimensions than a traditional vector space model ($S_{Trad}$). To compare document and query vectors in $S_{LSI}$ we need to transform them, using a projection from $S_{Trad}$ to $S_{LSI}$. The projection can be done in many different ways, for example with so called singular value decomposition (SVD), described in section 2.3.2. The aim with the projection is to squeeze the high dimensional vectors from $S_{Trad}$ into the more narrow space $S_{LSI}$ which is supposed to automatically catch similarity between distinct terms. Landauer (2002) shows some examples of subjectively more or less related words, and how they will be placed by LSI in a vector space. Some of the word relations Landauer mentions are listed below. When I write “close”, it is close in the sense that the cosine between the vectors corresponding to the words, is high.

- Synonym pairs, i.e. salt–NaCl and doctor–physician are close to each other.
- Word pairs with related meanings, i.e. red–green–orange and chemistry–physics are close.
- Derivative pairs like compare–comparison, detect–detectable are close.
- Different forms of the same stem (inflections), i.e. walk–walks–walking or thing–things are close, but note that the orthographic similarity is not used to form the LSI model.
- Antonyms like black–white, yes–no and hot–cold are highly related.
- Finally, we can see that blackbird–bird are close to each other, but blackbird–black are not, which is intuitively what we would expect from a good similarity metric.

The LSVM is a result of SVD or some other method to calculate the vector space, so the fact that we find the relations above are just empirical observations without any theoretical explanation. More reasoning on what kind of relations we could expect to find in an LSVM can be found in Sahlgren (2006).

2.3.1 Structure of the training data

The training data for an LSVM is almost always a co-occurrence matrix, but there are two distinct ways to build this matrix:

- **Document x Term**: Make a 2-dimensional matrix with all terms in one dimension and all documents in the other (see table 2.2). In each cell the number of occurrences of the corresponding term in the document is stored.
• **Term×Term:** Make a 2-dimensional matrix with all terms in one dimension \(d_1\) and a subset (could be the full set if the time or space complexity is not a problem) of all terms in the other \(d_2\) (see table 2.3). In each cell the count of the \(d_1\)-term in the context of the \(d_2\)-term is stored, where the context is typically defined as a specific window size, for example ±2 words as used in this section.

The number stored in each cell in the matrix could be the frequency as described above, but especially for the document×term case, the results get much better if we use a weighted version of the frequency instead.\(^7\)

To get a clear understanding of how to make these co-occurrence matrices from a corpus, I will start from the small example in section 2.1. We start with a small hypothetical document collection (see table 2.1) containing very short documents,\(^8\) just to explain the principles. Let us start with a small example collection of nine short documents (table 2.1).

With the document collection in table 2.1 as input, we can see the document×term-matrix in table 2.2. Table 2.3 shows the term×term-matrix from the same document collection. The numbers are co-occurrence counts between word types within the context size and the orders between co-occurring words are kept.

The tables show only the words with a global frequency of at least 2. We can see in table 2.2 that three words, *theory*, *plan* and *earth*, occur in only one document each. The other 11 words occur in at least two documents, so these words create links between documents because the documents have words in common.

Note that the term×term-matrix may be collected using individual word order or not. I have chosen to keep the word order in this example. Otherwise the matrix will be symmetric. An easy way to obtain a term×term-matrix \(M_{sym}\) without using the word order is to calculate: \(M_{sym} = M + M^T.\)^\(^9\) Consequently, from the encoded word order in table 2.3, we can see that the word *light* occurs followed once each by the words *theory* and *exercise*, but not vice versa. The only case of co-occurrence in both directions is the pair *(speed, earth)*. There is no word that co-occurs with itself, which can be seen from the fact that there are only zeros in the top-left to bottom-right diagonal.

It is most common not to use the individual order between co-occurring words, which means that there is no difference between the vectors representing \(\{w_1, w_2\}\) and \(\{w_2, w_1\}\). The example matrices in table 2.3 and table 2.2 are not as sparse as co-occurrence matrices normally are, since the

---

\(^7\)For example the tf-idf weight defined in equation 2.2.

\(^8\)So called function words have been removed from these examples. This is a normal thing to do since they do not affect the vector space very much but make the calculation more time consuming. These words (pronouns, prepositions, etc.) are evenly distributed over documents and will therefore make almost no difference to the resulting LSVM. These words are sometimes called stop-words and are collected in a stop-list, cf. section 2.1.1.

\(^9\)\(M\) is the original term×term-matrix including word order information.
2.3 Latent semantic indexing

Table 2.3: The term×term-matrix extracted from the document collection in table 2.1 using a context size of ±2 words.

<table>
<thead>
<tr>
<th>Left \ Right</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>light (1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>strongly (2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doctor (3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>running (4)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>speed (5)</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>need (6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>program (7)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>calculated (8)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>start (9)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>second (10)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>theory (11)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>exercise (12)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>plan (13)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>earth (14)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Documents here are chosen to give at least some non-zero elements while the matrices still have to fit the pages. Note that for a vector model using word×word matrices, a document vector is calculated as the sum of the word-vectors corresponding to the words it contains. Experiments by Lavelli et al. (2004) show that word×word-co-occurrences almost always outperforms the document×word-representation when the results are used for term categorization and term clustering.

### 2.3.2 Singular value decomposition

SVD (Berry, 1992) is one method to calculate the information needed for projection of a vector in a traditional vector space $S_{\text{Trad}}$ into the new more narrow $S_{\text{LSI}}$. In this particular application, the method takes a document × term matrix\(^{10}\) $A = \{a_{ij}\}$ (an $m \times n$ matrix) and outputs a decomposed version, equation 2.11, where $U$ is an $m \times m$ orthogonal matrix, $V$ is an $n \times n$ orthogonal matrix, and $\Sigma$ is an $m \times n$ diagonal\(^{11}\) matrix. The matrices have some more properties: The non-zero elements in $\Sigma$ are ordered such that $s_{11} \geq s_{22} \geq \ldots \geq s_{nn} > 0$ (assume that $n \leq m$) and these elements are the square roots of the $n$ eigenvalues of $AA^T$. It also holds that $U^TU = \ldots$

\(^{10}\)It is also possible to work with word by word matrices as the example in table 2.3. The term weighting is less crucial for word×word-matrices and extreme document sizes are not a problem, but on the other hand we will in most cases be forced to cut the matrix in one dimension since the number of terms normally is much greater than the number of documents.

\(^{11}\)I.e. $\Sigma$ has non-zero element only in the top-left to the bottom-right diagonal.
$V^T V = I_n$ where $I_n$ is the identity matrix of size $n$.$^{12}$

$$A = U\Sigma V^T$$

(2.11)

There are many algorithms that calculate the SVD, and the best suited for LSI is the so called Lanczos method (Berry, 1992), since it is more efficient for sparse data matrices like the typical co-occurrence matrices used in LSI. I will not go into more details here, since it is not important for my work to know exactly how the SVD is calculated, and I will focus instead on the properties of the resulting projection. What I want is to use the SVD to make a projection from $S_{Trad}$ to $S_{LSI}$. This is done by calculating $A_k$ as a new approximation of $A$, where the weights in each cell are smoothed based on co-occurrence between terms so that equation 2.12 holds.

$$A_k = \sum_{i=1}^{k} u_i \times \sigma_i \times v_i^T, k < r$$

(2.12)

$A_k$ is the best rank-$k$ approximation (in a least squares sense) to the matrix $A$, which means that vector similarity in terms of cosine in the original vector space $S_{Trad}$ is kept in the new vector space $S_{LSI}$. The projection from $S_{Trad}$ to $S_{LSI}$ is a linear transformation, and is easily calculated by matrix multiplication, which is also not computationally intensive. Performing the actual SVD on the other hand consumes large amounts of computer power in terms of RAM-memory needs and CPU time consumption. The time complexity for the Lanczos SVD is actually $O(N^2k^3)$ where $k$ is the dimensionality after reduction and $N$ is the summed number of rows and columns in the training matrix (Berry et al., 1995). Adding more data to an existing SVD is not straightforward. To get a perfect result we have to re-run the SVD with all the data, but there are methods to fold in more data into an existing SVD and calculate an approximation of the new vector space containing both the initial and the new data. This is important if the data collection is dynamic. As long as the increase is moderate, the approximated SVD is almost as good as a full SVD, but when the added data is bigger we have to make a re-run to get a good result (Gorrell, 2006).

For a discussion on more efficient algorithms for the SVD, I recommend Fierro and Bunch (1995) where an algorithm called SPK-RSVD-LSI is presented and tested. The results seem to be just as good as for the original SVD algorithms for the LATIMES and MEDLINE document collections, and more efficient. Witter (1997) is also interesting since they claim that their method “Downdating the Reduced Model” is less time consuming than recomputing the SVD when documents or terms are removed, and O’Brien (1994) present methods for efficient adding of information to an existing LSVM. The general Hebbian algorithm for incremental SVD (Gorrell, 2006) has as a main advantage a minimal memory requirement, but I

$^{12}$The identity matrix $I_n$ is the quadratic diagonal matrix with 1 all along the main diagonal.
am not aware of any information retrieval experiments using it. Another useful property is of course the incrementality, since it then very naturally can be used for growing data collections.

2.3.3 Other methods to calculate an LSVM

As a context we should note that there are alternative methods to calculate the projected lower dimensional vector space. Many other methods, like for example the Hyperspace Analogue to Language presented in Lund et al. (1995) or principal component analysis, have the same complexity problems as the SVD, but a method called random indexing has a potential to work much faster than the full SVD, since it avoids the need for a complex dimensional reduction (Sahlgren, 2005; Kanerva et al., 2000). Instead the idea is to add randomly generated sparse vectors of length up to a few thousands, corresponding to each context (document or word), to obtain the lower dimension context vectors directly. To my knowledge there are no optimized implementations of random indexing, but for the experiments in this thesis we could just as well use random indexing instead of SVD. The main reasons to use SVD in this work instead of random indexing is:

1. The optimal projection used in SVD is more tested in an information retrieval context, so it feels more safe at this point.

2. In these experiments we are using static document collections, so the fact that random indexing handles dynamic collections more efficiently is not critical.

As mentioned earlier, there is also research on how to do dynamic SVD for growing data collections, so maybe the efficiency aspect is not that important (Gorrell, 2006). There is no reason why my methods to include n-grams or MWUs in an LSVM should not work in a similar way if I was using random indexing or some other algorithm instead.
Chapter 3

The nature of a latent semantic vector model

The number of dimensions in the LSVM is an important choice. If it is too small, there is not enough room in the semantic space. Totally unrelated terms will get too close. If we instead choose too many dimensions, there will be no generalization effect which means that we get results similar to the traditional vector space model. Many research papers take a pragmatic view of how to choose the number of dimensions, and find out an optimum experimentally instead of trying to calculate a theoretically optimal number. Typical numbers mentioned are in the interval 100-300 dimensions (Deerwester et al., 1990; Dumais, 1995), but this depends on the document collection. A larger database seems to need a more high dimensional LSVM, which is also intuitively what one would expect since a larger collection spans over more semantic fields. What seems to be a fact is that around the optimal number, the performance is still close to optimal, so it is not critical to find the exact optimum. I will not try to theoretically find an optimal number of dimensions for an LSVM, but just accept that this is an unsolved problem so far.

3.1 Concepts versus dimensions

The dimensions in an LSVM can be thought of as concepts, but is this really close to the reality? Deerwester et al. (1990) point out that the meaning of LSI dimensions are complex and cannot be directly interpreted. This is not surprising, since even if there was a complete set of orthogonal concepts to fill up the conceptual space with, why should the dimensions found by SVD be identical with these concepts? The space may be rotated in many ways and still keep all vector distances intact. However, it is not realistic to think that a conceptual space can be constructed by orthogonal concepts recognized by humans. The concepts we know are not independent, and we can easily think of concepts for which no terms exist. But even if the dimensions do not correspond to concepts known by humans, we can still try to think of, let us say, a 300 dimensional space as 300 independent
3.2 How much meaning can be fit into a bag-of-words model?

Linguists may object that the bag-of-words representation used during the SVD calculation makes it impossible to catch the essential meaning of a text. The so called function words (pronouns, prepositions, adverbs, etc.) occur in all kinds of context, and will therefore get almost no weight at all during the calculation of a passage’s meaning. Some of them are even put in a stop-list to get no impact at all. But we all know that these words make a difference for the meaning, for example the word “not” may change the meaning to the opposite. On the other hand, I concluded in section 2.3 that antonym pairs are highly related to each other in an LSVM—actually just as highly related as synonym pairs (Landauer et al., 1998). There is an interesting article, Landauer et al. (1997), where the authors try to find out if LSI really can find a passage’s meaning by comparing human judgments of meaning similarity to judgements made by LSI. Based on the fact that human judgments correlate to a high degree with judgments made by LSI, they conclude that “a great deal of information about the meaning of passages may be carried by words independently of their order”. They also point out that LSI is much more than just counting and weighting words and co-occurrences. Another interesting feature of an LSVM used in Landauer et al. (1997) is that the length of the vector representing a section of text is a measure of “the amount of domain relevant information” as a complement to the vector direction (“the semantic direction”). It is a bit

---

1It is impossible for humans to really imagine such a vector space. However, the reasoning here will be the same for just the three dimensions that most people actually can imagine, except that a space of just three cannot cover a complete semantic space.

2A bag-of-words is a representation of a text that does not keep the word order, i.e. a set that keeps the frequency of each element.

3A passage here is just a segment of text, i.e. a section, sentence, phrase, etc.
unclear from their experiments if an LSVM really can give measures of both quality of quantity, but the idea of being able to separate these measures is really interesting.

### 3.3 Multiple meanings of terms in an LSVM

Multiple meanings of terms is a serious problem in an LSVM. Instead of a vector representing a concept, ambiguous words tend to be represented by a vector which is the weighted sum of the vectors for each meaning of the term. The more common meaning counts more which makes it very difficult to find the vector corresponding to the less common meanings. Thomas Landauer (Landauer et al., 1997) claims that the needed information actually can be found in the model if we look more carefully:

“To repeat, LSA deals with inconsistent contextual usages by representing a word as an equation containing many independent variables, usually several hundred. Therefore, a word in LSA could, in principle, have as many entirely unrelated meanings as there are dimensions in its representational space.”

Landauer’s reasoning on how an ambiguous word can make a passage ambiguous and how this fits well into the LSI representation is elegant, but this is not necessarily how it works in practice. We can agree that each dimension should stand for a meaning independent of all other dimensions, but does Landauer’s conclusion really follow from that? An unambiguous term should then only have one non-zero component in the LSVM, which is very rarely the case. On the other hand, depending on what I mean by different meanings of a word, one could argue that there are no such things as unambiguous terms. It is difficult to get an intuitive picture of how concepts are stored in an LSVM because of the assumption that the different dimensions or concepts are independent, which is not the case for any (complete) set of concepts in natural language. More research is needed here to come to clarity about ambiguous words in LSVMs, and probably by good researchers in all the fields of linguistics, psychology, computer science, and mathematics together.

### 3.4 MWUs in an LSVM

There is no theoretical problem to have an LSVM where single word terms co-exist with MWUs. An argument for this is that we could see a single word term as a special case of an MWU—a multi-word unit built up by only one word. The problems arise when we try to convert a query or a document to a set of vectors in the model that we can add together to reach the query or document vector. When we train the LSVM, we need to decide what to put in the bag-of-words, i.e. if it is supposed to contain MWUs and terms instead of just words, and we have the same problem during lookup.
3.5 MWUs in information retrieval

One aspect is however problematic. Some MWUs contain important terms that we sometimes want to be able to look up in the model. For example, we want to find *Bill Clinton* and *Hillary Clinton*, but also just *Clinton*. This is a problem both for training and lookup, but the three terms can co-exist in the model without problem. We would expect all three to be more related than the average between terms, but it is difficult to say if *Bill Clinton* should be closer to *Hillary Clinton* than *George Bush*. That will depend on which context we are looking at. If we have a document collection with a large proportion of articles about politics, the answer will probably not be the same as if we have more articles about family relations.

Wu and Gunopulos (2002) explored the usefulness of phrases in an LSI-based system for document classification. They used SVD to reduce the number of dimensions and as a test suite they used a relatively small document collection from Reuters containing 21578 documents divided into 135 topics. 9603 documents were used for training. They were not able to get a significant improvement when phrases are added to an already well performing single-word model, but they found one very interesting conclusion: “Useful phrases occur much less frequently than useful words.” Wiemer-Hastings (2000) and Wiemer-Hastings and Zipitria (2001) are investigating a different task, namely a system for automatic student tutoring. By comparing vectors corresponding to student answers to questions, their system were able to rate the students’ answers almost as well as human judges. One of their approaches to improve even more was to use so called *Structured LSA*, which is a preprocessing technique that identifies subject, object, and verb, and also resolve pronominal anaphora, etc. To be able to see the potential of this method, they did the preprocessing by hand before calculating the SVD. The correlations between the Structured LSA models and human judges were higher than between LSA and humans. In an IR task it may be a drawback not being able to search for single words when they are parts of phrases.

3.5 MWUs in information retrieval

We can all agree on the fact that word order is important for the meaning of a text, but on the other hand it is difficult to imagine how much information we can actually get from the raw bag-of-words representations. Modern information retrieval systems with advanced term weighting schemes like the BM25 perform surprisingly well. The positive effect of a good weighting scheme is even stronger on very large document collections. How much could we gain in terms of retrieval effectiveness if we were able to use both words and MWUs in the retrieval process? Mitra et al. (1997) tried to investigate if phrases could improve an IR-system. They had the possibility to run on

---

4 There are actually experiments performed 10 years earlier by Fagan (1987), doing roughly the same type of experiments, but they had to use very small document collections.
more realistic document collections (more than 200 000 documents) from TREC and they also tried to use both syntactic and statistical phrases. They managed to get a small but not significant improvement in average precision by combining syntactic and statistical phrases. Their thoughts about why the improvement is not significant is that modern information retrieval systems are so good at document ranking based on single words that adding the phrases will in many cases only emphasize aspects in the query that are already found by using single words. Turpin and Moffat (1999) continued the experiments from Mitra et al. (1997) by optimizing various parameters of how to choose the phrases to include in the index. They start with a worse initial model so their improvement is bigger. On the other hand the final precision value is lower, so they agree on the conclusions. However, they conclude that phrases helped at low recall levels but not at the top ranking positions. Turpin and Moffat (1999) agree with the suggestion in Smeaton and Kelly (1998) that a separate vector space could be created for phrases and then weighted together with the word-based vector space.

We can conclude that earlier attempts of adding phrases to IR-systems did not result in the improvements one would expect or hope for. But the document retrieval application is very complex, so there are many aspects that could affect the results negatively that have to be investigated empirically:

- The algorithm that finds the phrases to include—how do we know that the phrases that we think are “good phrases” really are the right to include to get an improved IR result?

- The term weighting schemes are developed for word-based vector-spaces. Maybe they should be defined differently when the phrases are included or at least tuned differently.

- The existing TREC test collections have the weakness that the pooling process may favour the kind of systems that have participated in the competitions since not all documents are relevance-judged.

- The conversion from topic to query is far from trivial. It may be easier and definitely more explored for words-only models.

- Many small choices are important, like how to handle stop-words, frequency thresholds, etc.

- The choices or parameters are not independent so each of them cannot be optimized globally.

There is still no final evidence that MWUs can or cannot improve traditional information retrieval, and even if there were, we could get a completely different result for an LSVM instead of the traditional vector space model without dimensional reduction.
Which terms that are actually indexed in an IR-system often depends on the tokenization. The easiest tokenization is to use whitespace as a delimiter of terms and to index all words in the documents as terms. In some languages, like English, compounds often contain words separated by spaces, while languages like Swedish or German put compounds together without spaces between the parts. It is also not fully standardized, especially for English, where to insert a whitespace or not. The set of terms based on whitespace as delimiter will become very different between Swedish and English. Based on the structural difference between Swedish and English, I expect the inclusion of MWUs in an LSVM used in an IR-context to be more important for English than for Swedish.
Chapter 4

Latent semantic vector models and applications

At this point we have seen that LSVMs have a potential for being useful as a resource in document retrieval systems. The goal with this chapter is to describe some applications, other than document retrieval, that can benefit from an LSVM. This list of applications is not an attempt to build a complete list, but rather an overview of some different applications where LSI is an interesting data source. We will see that the LSVMs come in very handy in many applications and one could easily come up with new applications for these models. Other applications not described here are for example: information filtering, text classification, and word sense disambiguation. After this chapter I will leave all applications except for the synonym test and document retrieval that we are going to use for evaluation, but it is important to have the other applications in mind since the LSVM in itself is supposed to work in all these applications also when the MWUs are added.

4.1 Synonym tests

A synonym test solved by a computer is not really a useful application but rather a test of how well the knowledge needed can be learnt by a computer. The application is defined as follows.

• Input:
  1. a set of queries that can be single words or phrases
  2. for each query a set of synonym candidates

• The task is, for each query, to select the candidate that is closest in meaning.

• Each set of query and candidates should be constructed in such a way that exactly one of the candidates can be seen as a synonym to the query.
One way to solve the task is to use an LSVM to find out which of the alternatives that is closest to the query in a vector sense. If the model covers all the terms and reflects their meaning, I expect this approach to achieve a reasonably good result (Landauer and Dumais, 1997; Karlgren and Sahlgren, 2001). The task is straightforward for an LSVM—just train the model on the right kind of data, calculate the vectors for the query word and its alternatives, and the alternative vector closest to the query vector is selected. Landauer and Dumais (1997) compare how well an LSVM and human beings can learn word meanings. It turns out that the scores on the TOEFL test (“The Test Of English as a Foreign Language”) are similar for humans and computers with a similar amount of training data. Other methods based on web search have been shown to score much higher than SVD (Turney, 2001), but they have used the full web as information source rather than an amount of text similar to what a human has read until the age when he or she takes the TOEFL test, which is what people have used with the LSVM approach.

4.2 Information retrieval

The use of LSI in information retrieval (IR) is already mentioned, since the traditional vector space model, which LSI is based on, is so much a product of attempts at doing good IR (Salton and McGill, 1983; Baeza-Yates and Ribeiro-Neto, 1999). This section gives some more information on the typical benefits for IR given by LSI.

4.2.1 Document retrieval

Document retrieval is the task of finding the (most) relevant documents in a collection, given a query. Defining the task a little more in detail gives the following variations.

- Among a set of documents, find the subset of documents relevant to a given query
- Rank a set of documents according to relevance to a given query

In some applications we need just one of these tasks and in other ones, we want to perform both of them: find the relevant subset of documents and rank them after relevance to a query. The ranking task is the one where LSI comes in most naturally, since the ranking can be based on all terms in each document instead of just the terms present in the query. LSI will not help very much when it comes to deciding if a document is relevant or not, but if the system has the capability to rank the full document collection after relevance to the query, the threshold between relevant or not is not crucial.

We will see in section 10.2 what kind of data that enables an LSVM to solve synonym tests.
for the user as long as the ranking is good enough. LSI is probably used in many document retrieval systems, but most of them are commercial, so I do not know exactly how the vector models are used (Telcordia, 2003; Chen et al., 2001).

4.2.2 Query expansion

Many different approaches to query expansion exist, but the general idea is to add query terms to the query before sending it to the search engine to gain an improvement in retrieval effectiveness (Ahlgren, 2004b). The improvement can be substantial, especially for systems where users type in very short queries. Useful resources to obtain automatic query expansion are term weighting schemes and co-occurrence statistics (Qiu and Frei, 1993; Mitra et al., 1998).

With access to an LSVM there are at least these three possibilities:

1. For each term in the query: find related terms and add these to the query.
2. Find terms related to the query as a whole and add these to the query.
3. Do a first preliminary search and use the $n$ most relevant documents to expand the query.

For the first approach we could use a similarity thesaurus which could be constructed automatically. The second approach could use an LSVM directly. Terms close to the query-vector are good candidates to be added. In the third method, we can compute vectors for each $n$ document returned by the first search and then add a set of terms close to these vectors. Qiu and Frei (1993) argue that their concept-based query expansion is better than LSI because they avoid the problem to decide on an optimal dimensionality. This is not that crucial if the document collection is mostly static, and otherwise there are alternatives to the full SVD as I mentioned in section 2.3.2 and 2.3.3.

All these alternatives for query expansion can be implemented fully automatically with a possibility for the user to choose which of the computed terms to actually add before sending the query to the search engine.

4.3 Document clustering

There are some different variations of the document clustering task. What they have in common is that the document clustering system should:

1. Take as input a set of documents and possibly a set of categories
2. Assign a category to each document
The first choice to make is if the documents should be categorized into known or unknown categories. The first case is rather a task of document classification than clustering. The case of unknown categories can be implemented fully data-driven where one argument can be the preferred number of categories. We also need to decide if the clustering should be flat or hierarchical. These properties give four basic combinations:

- Flat partitioning into a number of categories decided from the data, i.e. not known in advance.
- Flat categorization into a set of known categories.
- So called *agglomerative clustering* (a clustering algorithm) into a hierarchy not known in advance.
- Categorization into a hierarchy of known categories.

A more detailed overview of the clustering task and various algorithms can be found in Rosell (2005), chapter 3.

### 4.3.1 Using an LSVM for document clustering

Document clustering is an application where an LSVM can be used in a very straightforward way:

1. Train an LSVM using a representative document collection. It may contain all the documents that are going to be clustered, but this is not necessary.
2. Use the LSVM to compute the $n$-dimensional vector for each document.
3. Run your favourite clustering algorithm on the set of vectors, using for example Euclidean distance.\(^2\)
4. The result of the clustering is, depending of the choice of algorithm, one level or hierarchical clusters of documents close to each other in the LSVM space.

If we have a classification hierarchy or known flat categories we could use vector distance between documents and category descriptions in an LSVM to find out the closest category.

\(^2\)If we want to place the documents into a tree structure, we can use agglomerative clustering. If the set of documents is very big, the bad time complexity makes it practically impossible to use agglomerative clustering. An alternative is the K-means algorithm that gives us just a partitioning of the document set, but with a much better time complexity.
4.3.2 Earlier experiments

Using an LSVM (trained by SVD) for document clustering seems to be a well performing method. Deerwester et al. (1990) point out that a clustering based on an LSVM improves the efficiency of document retrieval but maybe not the quality compared to a traditional vector space model. Xu et al. (2003) also use a vector space model, but with non-orthogonal base vectors. They claim, based on experiments, that their method both makes it easier to derive the cluster and also gives a better final result.

4.4 Keyword extraction

Keywords are words\(^3\) that are selected with the intent to represent the content of a document, and the task can be defined as: Given a document database \(D\) and a document \(d\), and possibly the number of wanted keywords or a quality threshold, return a list of keywords that are representative for \(d\).

If we want keywords for each document in \(D\), we could just run the algorithm several times. Using knowledge from the IR-field we can use tf-idf\(^4\) based algorithms or other statistical formulas like Z-score (Liu, Ciliax, Borges, Dasigi, Ram, Navathe and Dingledine, 2004).

Keyword extraction can be performed directly using an LSVM (Fortuna et al., 2005). For a specific document \(d\) in a document collection \(D\), the LSVM is trained on the full document collection. The list of terms closest to the vector \(\vec{d}\) (calculated from \(d\)) forms a list of keywords. The list is sorted by distance to \(\vec{d}\), see table 4.2 for a realistic example.\(^5\) One major problem with this approach to extract keywords is that only the terms in the LSVM can be found, so if the model only contains single words, we will not get any compounds. Another problem that is actually compensated for in table 4.2 is that words with very low frequency is in many cases not what humans would choose as keywords. Therefore, words with an absolute frequency below 10 are removed. In table 4.3 I have done the same thing as in table 4.2 but now the used LSVM also contains word pairs. The documents \(d_1\) and \(d_2\) can be found in table 4.1. \(d_1\) and \(d_2\) are lemmatized. Note that the lemmatizer is a very simple language-independent implementation without any kind of lexicon. Just simple rules for groups of characters. Developing a good lemmatizer is not part of this work. On the other hand, the used lemmatizer is language-independent.

Compared to for example a keyword extraction algorithm based on mutual information, the LSVM algorithm has the strength that it can find relevant keywords that are not present in the document but somewhere else in the document collection.

\(^3\)Possibly compounds or phrases depending on what kind of keywords we want.

\(^4\)Read about tf-idf term weighting in section 2.1.2.

\(^5\)The used example documents can be found in table 4.1.
Table 4.1: Some example documents used in section 4.4

<table>
<thead>
<tr>
<th>Id</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>maradona lives: at least argentina’s flawed 33-year-old glory thrived against an outclassed greece, scoring a goal and playing all but the last seven minutes in his team’s 40 win at foxboro stadium, massachusetts. he was not the argentina star of the day. that honour belonged to gabriel (the archangel) batistuta, who achieved the world cup’s first hattrick a secondminute poke through the greek goalkeeper’s legs, a 43rdminute right foot blast from 18 yards and an injury time penalty after a handball offence. ...</td>
</tr>
<tr>
<td>$d_2$</td>
<td>moscow, 9 mar boris yeltsin has refused to receive the former u.s. president, richard nixon, who is paying an unofficial visit to russia. he announced this during a short conversation with journalists at the kremlin wall, after the laying of wreaths at the grave of yuriy gagarin. the russian president said that: nixon has met rutskoy and zyuganov, but the interesting thing is that he came here to meet me. after that, yeltsin noted: i will not meet nixon, the government will not receive him, and filatov [yeltsin’s chief of staff] will not receive him ...</td>
</tr>
</tbody>
</table>

4.5 Automatic thesaurus extraction

The task is to automatically create a thesaurus, i.e. a list of main words and for each of them a list of related words. If possible, the words under each main word can be sorted by type of word or type of relation to the main word. The extraction is made given a training corpus so the thesaurus will be domain specific.

We have seen in section 2.3 that LSVMs can find words related to each other in many different ways, for example synonyms and antonyms. Kilgarriff and Yallop (2000) and Peters and Kilgarriff (2000) point out that a “looser” thesaurus of the type that could be generated by LSI is useful in language engineering, but if we want a lexicon, we should instead use a “tighter” thesaurus. In Schütze (1998) SVD-based LSI with 100 dimensions is used to find word vectors. Earlier experiments by Schütze have shown that this dimensional reduction does not decrease the sense discrimination power. In this work he also performs sense disambiguation which makes it possible to build a thesaurus with the same word form with different meanings at many different locations in the thesaurus.

At first glance it may look like a weakness, but in practical use it is
Table 4.2: Top-10 words closest to the example document $d_1$

<table>
<thead>
<tr>
<th>Rank</th>
<th>Term</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>missed</td>
<td>0.39</td>
</tr>
<tr>
<td>2</td>
<td>behind</td>
<td>0.38</td>
</tr>
<tr>
<td>3</td>
<td>maradona</td>
<td>0.37</td>
</tr>
<tr>
<td>4</td>
<td>beat</td>
<td>0.37</td>
</tr>
<tr>
<td>5</td>
<td>playing</td>
<td>0.37</td>
</tr>
<tr>
<td>6</td>
<td>ball</td>
<td>0.36</td>
</tr>
<tr>
<td>7</td>
<td>left</td>
<td>0.36</td>
</tr>
<tr>
<td>8</td>
<td>played</td>
<td>0.36</td>
</tr>
<tr>
<td>9</td>
<td>game</td>
<td>0.36</td>
</tr>
<tr>
<td>10</td>
<td>soccer</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Table 4.3: Top-10 words and bigrams closest to $d_2$ in a mixed model of words and $n$-grams

<table>
<thead>
<tr>
<th>Rank</th>
<th>Term</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>saying that</td>
<td>0.56</td>
</tr>
<tr>
<td>2</td>
<td>explained</td>
<td>0.54</td>
</tr>
<tr>
<td>3</td>
<td>convinced</td>
<td>0.54</td>
</tr>
<tr>
<td>4</td>
<td>referring</td>
<td>0.53</td>
</tr>
<tr>
<td>5</td>
<td>conversation with</td>
<td>0.52</td>
</tr>
<tr>
<td>6</td>
<td>an interview</td>
<td>0.51</td>
</tr>
<tr>
<td>7</td>
<td>his words</td>
<td>0.51</td>
</tr>
<tr>
<td>8</td>
<td>his opinion</td>
<td>0.50</td>
</tr>
<tr>
<td>9</td>
<td>president_yeltsin</td>
<td>0.49</td>
</tr>
<tr>
<td>10</td>
<td>boris</td>
<td>0.46</td>
</tr>
</tbody>
</table>

often a strength: A thesaurus automatically extracted from an LSVM is highly domain specific. This is good in some cases, because many domain specific terms are also used in other domains but with a different meaning. Typically, a manually collected thesaurus like Roget’s thesaurus (Roget, 2002) is a general language thesaurus which makes it impossible to cover all separate domains. This will make it much less useful in an IR context than a set of domain specific automatically trained thesauri for each domain that the IR system is supposed to cover.

Table 4.4 shows a few main words and the top-10 closest words in an LSVM generated from TREC-data. This LSVM is the same as the one used in section 11.3. I have selected some words of different types, and also both good and bad thesaurus entries. If the LSVM is used in an IR context, we never see these details of the model, but we can see here that some kind
“The Hand of God goal was scored by Diego Maradona in the quarter-final match of the 1986 FIFA World Cup between England and Argentina, played on 22 June in Mexico City’s Estadio Azteca. Maradona illegally used his left hand to score the goal, but the referee thought he had used his head and allowed the goal.”, www.Wikipedia.org

Figure 4.1: The hand of God goal

of manual filtering is needed to get what we normally would call a thesaurus. However, this manual work is much less than the work of building a domain specific thesaurus from scratch.

Table 4.4: Some main words from an example thesaurus automatically generated from TREC-data

<table>
<thead>
<tr>
<th>Main word</th>
<th>Associated words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maradona</td>
<td>Shilton, Romario, FIFA, soccer, goalkeeper, referees, midfielder, Bebeto, Baggio, Koeman</td>
</tr>
<tr>
<td>Nixon</td>
<td>Watergate, Reagan, Kissinger, Ronald, Jimmy, Scowcroft, Bush, Lyndon, president, Quayle</td>
</tr>
<tr>
<td>pineapple</td>
<td>cream, cheese, chicken, shredded, juice, slices, chopped, broth, onion, sauce</td>
</tr>
<tr>
<td>sports</td>
<td>athletes, sport, olympic, olympics, tennis, sporting, championships, tournaments, soccer, football</td>
</tr>
<tr>
<td>christmas</td>
<td>thanksgiving, shoppers, easter, prechristmas, halloween, retailers, festive, weekends, festivities, stores</td>
</tr>
</tbody>
</table>

We can see from the entries in table 4.4 that this “thesaurus” is really dependent on the training data. One striking observation is that even if these words are highly frequent, some specific events get into the top of the word lists. See for example Maradona, who is one of the top three (?) football players of all times, but still we have the words: Shilton, goalkeeper, and referees among the top-10 words, just because of one single event where he scored against the English goalkeeper Peter Shilton (see figure 4.1). We can see the same effect for Nixon. The term president is in the list but the term closest to Nixon is Watergate.

--Shilton is not mentioned in the example document about Maradona in table 4.1. The closeness between Shilton and Maradona comes from other documents and co-occurrences between other words.
Chapter 5

Evaluation of latent semantic vector models

This chapter describes the two evaluation methods I have selected for this work and it also gives some information about which aspects of an LSVM they measure. Using an LSVM as a component in an IR-system is probably the most important practical application for LSVMs. I have chosen document retrieval to obtain a well developed evaluation methodology for my models, and as a complement also a synonym test based evaluation.

5.1 Intrinsic and extrinsic evaluation

There are two philosophically different ways of doing evaluations. We can measure either intrinsic or extrinsic value. According to Zimmerman (2004) the intrinsic value is the value that a thing has “in itself” or “in its own right” and extrinsic value is the value that is not intrinsic. In my case, this can be interpreted to mean that evaluation in an application is extrinsic and when we try to look at the model as it is, we will make an intrinsic evaluation. Since an LSVM contains so much fine-grained information in so many dimensions, it is very difficult to make a fully intrinsic evaluation. Even if it just contains a few words, it is very difficult to find a gold standard for word similarities. I have chosen two applications for evaluation: synonym test and document retrieval. These two will catch different aspects of the LSVMs. The document retrieval evaluation is clearly extrinsic. We will never see any details of what the LSVM actually gives since document retrieval evaluates the LSVM embedded in an application.

Synonym test evaluation is different. Here we will at least see which synonyms the system will choose. The score will measure how often the LSVM can find the right synonym, which makes the synonym test evaluation in some sense more intrinsic than document retrieval evaluation (Jones and Galliers, 1993). We can actually see similarities between single terms in an LSVM. A fully intrinsic evaluation of an LSVM is difficult to design since we then would need a gold standard telling how close different vectors should be to each other in a perfect semantic vector space. What we actually can evaluate fully intrinsic are things like the efficiency for lookup and training,
and stability over different training sets. An intrinsic evaluation would not use any kind of application but using the LSVM as it is. Since the main use of LSVMs is in some kind of application, extrinsic evaluation is of more interest for the research field.

One criticism against LSVMs is that the dimensions in the vector space are difficult to interpret. This is true and none of the selected evaluation tasks make any use of this information. A fully intrinsic evaluation of a semantic vector space could take this property into consideration as well, but I have accepted that looking at separate dimensions in an LSVM is not meaningful or interesting for the kind of applications I have looked at in section 4. Considering the fact that solving synonym tests is a rather artificial task and not really an application, the synonym test evaluation is the most intrinsic evaluation task I can find that has some kind of meaning and still be well defined. Automatic thesaurus extraction would be more intrinsic, but the result is much more difficult to judge.

## 5.2 Evaluation with synonym tests

Synonym test evaluation is much less complicated than document retrieval evaluation, but we still have to describe the task and the evaluation metrics used.

### 5.2.1 Definition of the evaluation task

The synonym test is a set of queries. Each query consists of a query word or phrase and a list of synonym candidates. Among the candidates, exactly one is a synonym to the query word (or phrase). The task for the system is to choose the right alternative for each query. Except for the right alternatives, the rest of them are seen as equally bad from a scoring perspective.

The evaluation metric is straightforward—just calculate the proportion of correct answers. These give 1 point each and wrong answers are counted as 0 points each. There are other possible definitions of an evaluation metric. For example we could use the average rank of the correct answer among the alternatives. This metric could be useful for comparing low performing systems, but I am not going to use it here, but instead use the binary metric above, which is also the metric used when the test is used on humans.

### 5.2.2 Earlier experiments

To my knowledge all synonym test evaluations of LSVMs have been performed using the TOEFL test ("The Test Of English as a Foreign Language"). This kind of test is a good source of evaluation data since it is not constructed for testing a formal model like an LSVM, but rather human beings. Some researchers claim that what an LSVM can learn based on the training data using the SVD is similar to the knowledge a human achieves from the same amount of "training data". Landauer and Dumais (1997)
made many experiments comparing what the model learned compared to humans with similar document sets. On the TOEFL test, Berry et al. (1995) and Landauer and Dumais (1997) report a 64.4% score using LSI trained by SVD, and Karlgren and Sahlgren (2001) get a slightly better score with random indexing instead of SVD, but the differences highly depend on what training data that have been used. Turney (2001) uses “Pointwise Mutual Information” with a huge training set, namely the Web indexed by Alta Vista, to reach a much better result than the LSI experiments by Berry’s team and Karlgren & Sahlgren, respectively. Rapp (2003) reports a result of 92.5% using SVD and the British National Corpus as training data. Compared to Landauer and Dumais (1997), a word-by-word matrix and a small ±2 window were used, the training data was also lemmatized and stop-words were filtered away. Still, this huge improvement compared to similar experiments, is difficult to explain. It is maybe not likely that LSI is the best automatic method to get the top scores on synonym tests, but since the goal is not to win synonym test competitions, they could still be interesting as evaluation sets for LSVMs.

5.3 Evaluation with document retrieval

I will now define a framework for the document retrieval task. The basis is the Cranfield evaluation framework, but in practice it is necessary to rethink some of the properties of this framework.

5.3.1 The Cranfield evaluation framework

The goal behind the Cranfield 2 experiments was to define a laboratory environment for document retrieval systems (Cleverdon, 1991). The framework contained:

- A document database $D$, containing a large set of documents.
- A set $I$ of information needs expressed in some way, for example in plain text.
- For each information need in $I$, all the documents in $D$ have been relevance-judged.

There was only one dimension of relevance, namely topical similarity, and it was also defined as binary—a document is either relevant or not.\footnote{In the first Cranfield experiments a five-graded scale of relevance was used.} Relevance was also defined as an objective property, so user preferences were not included in the framework. In the definition above, the number of relevance-judged documents is not mentioned since all documents in $D$ were supposed to be judged by relevance to get a complete list of relevant and irrelevant documents for each information need. Since we in this framework
know the number of relevant documents, the basic accuracy metrics Recall and Precision will work fine.\(^2\)

\[
Recall = \frac{|D_{\text{relevant}} \cap D_{\text{retrieved}}|}{|D_{\text{relevant}}|} \quad (5.1)
\]

\[
Precision = \frac{|D_{\text{relevant}} \cap D_{\text{retrieved}}|}{|D_{\text{retrieved}}|} \quad (5.2)
\]

Recall and Precision can be combined to get the F-score (equation 5.3). The value \(\alpha\) is the weight of how important each metric is.

\[
F_\alpha = \frac{(1 + \alpha) \times \text{Precision} \times \text{Recall}}{\alpha \times \text{Precision} + \text{Recall}} \quad (5.3)
\]

For \(\alpha = 0\) we get \(F_0 = \text{Precision}\), \(\alpha = 1\) gives the same weight for Recall and Precision respectively, and \(\alpha > 1\) gives higher weight for Recall. Just as for Recall and Precision, F-score range from 0 to 1. Since both Recall and Precision are problematic in practical IR-evaluation, F-score is also not very useful.

### 5.3.2 The TREC environment

TREC (Text REtrieval Conference) is an annual information retrieval conference and competition, arranged by NIST (National Institute of Standards and Technology). Since the TREC conference has been held many times with many participating groups and systems, the concept has been tried out quite well.

#### 5.3.2.1 Properties for the TREC environment

The framework for the document retrieval task is similar to the Cranfield evaluation framework, but for practical reasons, some of the properties have been changed:

- The set of information needs is described as a set of topics (\(T\)), where a topic \(t\) consists of three parts: Title, description, and narrative.
- The set of relevance-judgements for each topic is not complete. Instead the set of relevance-judged documents is formed by so called pooling.

The TREC environment is more adapted to the reality that it takes too much time and money to judge all documents by relevance in \(D\), since \(D\) may contain millions of documents. Typically, \(T\) contains 50 or 100 topics, which makes it impossible to follow the Cranfield framework. Instead, pooling is used to select which documents that should be judged by relevance.

\(^2\)Note that \(D_{\text{relevant}}\) denotes the set of relevant documents from the full set of documents, and not just the set of retrieved relevant documents. \(D_{\text{relevant}}\) is often practically impossible to count, so some kind of estimation has to be performed. However, in the ideal Cranfield evaluation framework this problem does not exist.
5.3.2.2 The pooling process

The pooling process is needed since it is too expensive to relevance judge the full document collection for each topic. Instead, a selection is made based on the documents retrieved by the participating systems. CLEF and TREC use a similar procedure:

1. Each system $s$ in the competition retrieves a top-1000 list for each topic $t \in T$.

2. For each topic $t$, the union of all top-100 documents from all systems forms the pool $P_t$:

$$P_t = \bigcup_{s \in S} \bigcup_{i=1}^{100} \text{document}(s, t, i)$$

$\text{document}(s, t, i)$ is the $i$:th ranked document for $t$ by the system $s$.

Note that not all retrieved documents are relevance-judged but just the top-100 from each system. Again, this is to make the judgement work less time consuming. This definition of pooling guarantees at least 100 relevance-judged documents per topic, but it may be the case that none of these are relevant. A topic where all systems agree on the top-100 will get a pool-size of exactly 100 documents and if all systems disagree such that they have no document in common on the top-100 positions, we end up with the maximum pool-size: 100 times the number of participating system.

5.3.3 Evaluation metrics for document retrieval

In this section I will present existing evaluation metrics. The initial metrics, Recall and Precision (equation 5.1 and equation 5.2) are of theoretical interest since all metrics in this thesis are based, more or less, on them.

5.3.3.1 Notation

Let us just recall the notation especially important in this section. We have a database of documents $D$ and a set of topics $T$. For each topic $t \in T$ there is a pool $P_t = \{d_1, ..., d_p\}$ of relevance judged documents such that $P_t \subseteq D$. Among these, $R_t = \{d_1, ..., d_r\}$ and $N_t = \{d_{r+1}, ..., d_p\}$ are the sets of known relevant and irrelevant documents respectively, so $\{R_t, N_t\}$ is a partition of $P_t$. For an arbitrary topic I will use: $P$, $R$ and $N$. Their sizes will be denoted: $p = |P|$, $r = |R|$ and $n = |N|$.

---

3CLEF and TREC organize international competitions for various IR tasks. TREC is the older and more well known competition and CLEF focuses more on multi lingual tasks.

4The number of relevance-judged documents from each retrieved list is called $dcv=$Document Cut-off Value.. Note that $dcv$ in CLEF and TREC is smaller ($dcv = 100$) than the number of retrieved documents for each topic, which is 1000. In some cases an even smaller $dcv$ was used for the CLEF data.
5.3.3.2 The traditional evaluation metrics

The most well known evaluation metrics are *Recall* and *Precision* (equation 5.1 and equation 5.2), and many other metrics are developed from these. These metrics also have very intuitive interpretations:

- **Recall**: How large proportion of the existing relevant documents did we find?
- **Precision**: How large proportion of the retrieved documents were relevant?

One major problem with *Recall* is that we need the number of relevant document in the database, which we normally do not know. If we use only *Recall* without the *Precision*-value, we sometimes get a very bad picture of the system’s performance, since retrieving all documents in $D$ will always give $\text{Recall} = 1$. What people often do is to calculate *Precision* at different *Recall*-levels. Another weakness with *Recall* and *Precision* is that they are not applicable at all for the typical IR task of just ranking a database of documents for relevance. In this case *Recall* will always be 100% and *Precision* the same as the proportion of relevant documents in the database, i.e. the same values for all systems without taking the ranking into account at all.

5.3.3.3 Mean uninterpolated average precision

A more tailored metric for information retrieval than the most well known *Recall* and *Precision* metrics is the mean uninterpolated average precision (MAP). MAP (equation 5.4) is an average over the topics of the average precision ($\text{AP}$) which is calculated as an average over the relevant documents for a topic $t$ and a system $M$, defined in equation 5.4. *Precision*($d_i$) is the precision for the retrieved list down to document $d_i$.

$$
\text{AP}(t, M) = \frac{\sum_{d_i \in D_{\text{relevant}} \cap D_{\text{retrieved}}} \text{Precision}(d_i)}{|D_{\text{relevant}} \cap D_{\text{retrieved}}|} 
$$

$$
\text{MAP}(M) = \frac{1}{|T|} \sum_{t \in T} \text{AP}(t, M) 
$$

The sets $D_{\text{relevant}}$ and $D_{\text{retrieved}}$ of relevant and retrieved documents are the values for the current topic $t$. Precision is dependent on both $t$ and $M$.

To be able to use MAP (equation 5.5) in the TREC environment (defined in section 5.3.2), we have to decide what to do with the retrieved documents that are not relevance-judged. Normally, in a TREC evaluation, all non-judged documents are assumed to be irrelevant. This assumption will be discussed more in detail in section 7.3. AP does not use all relevance-judgements. Known irrelevant documents are used only if they are ranked better than the worse ranked relevant document according to $M$.

---

5I use $M$ for system since a system in this case is equivalent to a model.
5.3.3.4 R-precision

R-precision counts the share of relevant documents among the top $r$ documents (note that $r$ is the number of relevant documents), we define:

$$R\text{-}precision(t, M) = \frac{|R \cap P_r|}{r}$$

Where $P_r$ is the set of $r$ top-ranked documents by system $M$ for topic $t$.

In some cases R-precision is a good metric, but consider the following sequences of relevant (+) and irrelevant (−) documents, from four different systems $M_1, ..., M_4$ when $r = n = 10$ (table 5.1). $M_1, ..., M_4$ will all get the result $R\text{-}precision = 0.5$, but for example $M_3$ is clearly worse than $M_2$ as a result from a search engine, so for the cases in table 5.1, R-precision is not very informative.

Table 5.1: Ranking lists from four hypothetical systems that demonstrate one problem with R-precision

<table>
<thead>
<tr>
<th>System</th>
<th>Ranking list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>++++−−−−−−−+++</td>
</tr>
<tr>
<td>$M_2$</td>
<td>++++−−−−−−−+++</td>
</tr>
<tr>
<td>$M_3$</td>
<td>−−−−++−−−−−+++</td>
</tr>
<tr>
<td>$M_4$</td>
<td>−−−−++−−−−−+++</td>
</tr>
</tbody>
</table>

5.3.3.5 Buckley & Voorhees: bpref

bpref (Buckley and Voorhees, 2004) in equation 5.6, is a metric defined to work for test suites with incomplete information, presented at the SIGIR conference in 2004. For the model $M$ and the topic $t$ containing a set $R$ of $r$ known relevant documents and a set $N$ of known irrelevant documents, bpref is defined as a sum over the $r$ documents in $R$ ($d_i$), let $N_r$ be the set of the $r$ most highly ranked known irrelevant documents, and $N_r(d_i, t, M)$ the number of documents $d$ such that $\text{Rank}(d, t, m) < \text{Rank}(d_i, t, M)$, where $d_i \in R$, $d \in N_r$.

$$bpref(t, M) = \frac{1}{r} \sum_{i=1}^{r} 1 - \frac{N_r(d_i, t, M)}{r}$$  \hspace{1cm} (5.6)

Note that the definition does not use the assumption that documents that are not judged by relevance are counted as irrelevant. One problem with the bpref definition that the authors note in their paper, is that only the same number of known irrelevant documents as the number of known relevant ones ($r$) are used for the calculation. This is a problem especially when $r$ is low. bpref-10 (equation 5.7) is an improved variant of bpref, using $r + 10$ irrelevant documents, but it still does not use all known irrelevant documents:
5.3 Evaluation with document retrieval

\[ bpref-10(t, M) = \frac{1}{r} \sum_{i=1}^{r} 1 - \frac{N_{r+i0}(d_i, t, M)}{r + 10} \]  \hspace{1cm} (5.7)

Note also that the \textit{bpref} metrics is only defined for data sets with at least \( r \) (or \( r + 10 \) for \textit{bpref-10}) known irrelevant documents. If we accept that “the first \( r \) judged irrelevant documents as retrieved by the system” (Buckley and Voorhees, 2004) may not exist, then we could define \textit{bpref} and \textit{bpref-10} even if \( n < r \), but maybe the results are not what Buckley & Voorhees intended.
Part II

Evaluation Methods and Data Collections
Part II: Evaluation Methods and Data Collections

In this second part, I present two new contributions to the evaluation of an LSVM, in the form of a new test collection for synonym test evaluation in Swedish, and a new evaluation metric for document retrieval. For each of the two evaluation methods I will present:

- The collection of data to use for the evaluation
- Suitable metrics
- A discussion on significance testing

Chapter 6
The data collection for evaluation of LSVMs, based on an annual Swedish synonym test used as a qualification test for university studies, is presented. This collection is later in the thesis used for evaluation of LSVMs.

Chapter 7
This chapter defines new evaluation metrics for document retrieval, aware of the problem with uncomplete relevance information. A series of experiments show that the metrics, in particular RankEff, are good complements to the commonly used MAP.

Chapter 8
Chapter 8 focuses on how to interpret the results from the two evaluation tasks described in chapter 6-7. A scale of improvement is specified, followed by discussions on how to do significance testing in the two cases. Both resources are important for the third part of the thesis where I perform evaluations of differently configured LSVMs. The experiments are needed to be able to find out how well an LSVM can function when it is trained to contain both single words and MWUs (or n-grams).
Chapter 6

Synonym test based evaluation

In this chapter I will look at the synonym test evaluation task and also present a new synonym test evaluation data collection. The collection is for Swedish and contains 560 queries, which is very good compared to the English collection (see section 4.1 for more information) with only 80 queries commonly used in LSI experiments.

In part III of this thesis, I will use the defined framework and the presented dataset to see if \( n \)-grams can improve the results of a synonym test based evaluation, but the evaluation set is very general and can be used to test or compare any set of semantic models that provide similarity metrics between words (or phrases).

6.1 Definition of the synonym test task

I will start with a detailed definition of the task.

- The input is:
  1. a set of queries that can be single words or phrases,
  2. for each query a set of synonym candidates.

- The task is, for each query, to select the candidate that is closest in meaning.

- Each set of query and candidates should be constructed in such a way that exactly one of the candidates can be seen as a synonym.

- The correct alternative for each query gives 1 point, the other alternatives give 0 points.

6.2 SweHP560 synonym test data collection

The evaluation set SweHP560\(^1\) is based on a Swedish synonym test which is a part of “Högskoleprovet” (an entrance test for university studies) (Grön-}

\(^1\)The name SweHP560 comes from the number of queries, the language Swedish, and the original name of the test: “HögskoleProvet”.

49
There is a new test twice a year and the collection contains the exercises from 14 tests from the years 1998-2006, in total 560 queries with five alternative answers each. Hopefully, the size of the test suite is big enough to give significant differences between various LSVMs. Here are a few examples to get the picture—the correct answer is marked in italics.

- **paradoxal (paradoxical)**: förmodad (assumed), fördelaktig (advantageous), likartad (similar), övertygande (convincing), motsägelsefull (contradictory)

- **avisera (announce)**: inrikta (aim), utvärdera (evaluate), rådgöra (consult), underrätta (inform), avgränsa (delimit)

- **ambulera (ambulate)**: flytta omkring (move around), sätta igång (get started), byta bort (exchange to something else), skynda på (hurry up), hålla fast (hold tight)

- **avbräck (setback)**: ekonomisk skada (economical loss), utjämning (level out), allvarligt misstag (serious mistake), förkortning (abbreviation), synligt hinder (visible obstacle)

- **ta skruv (do the trick)**: få nog (get enough), ta sats (take-off), gå snett (go awry), ta i, ha verkan ("have effect")

- **inte skräda orden (not mince matters)**: artikulera väl (articulate well), säga på ett fint sätt (say in a nice manner), säga sin uppriktiga mening (give one’s honest opinion), vara fåordig (being silent), vara pratsam (being talkative)

More details about the ORD test can be found in Scott (2004). Here is some information relevant for this thesis:

- Each ORD test consist of 19 nouns, 11 verbs, 8 adjectives, and 2 adverbs. The proportions are based on a Swedish frequency dictionary (Allén, 1972).

- Each test contains at least some queries where the task is to find a hyponym instead of a synonym.

- The alternatives should be built up by easier words than the queries.

- The authors of the ORD tests try to include words from each of the following fields in each test:
  - technology and science
The ORD test has been criticized for several reasons. For example, it is said to be unrealistic since the words are taken out of context. Some researchers claim that this kind of test is not gender neutral but it is unclear if men or women are favoured. I cannot see any reason why this criticism should be problematic for the use of the ORD test in this thesis.

One should note that ORD in “Högskoleprovet” is rather difficult even for future university students. The result for these tests was in average between 55% and 60% among people who are trying to qualify for university studies 1996-1998 (Stage et al., 1998). Many of the words are rare or old-fashioned, and some are idioms like for example *hugget som stucket* (means *whatever* or word by word: *cut as stung*). Especially the multi-word idioms should confuse an ordinary LSVM.

### 6.3 Statistics for SweHP560

I will give some basic facts about the data collection here. Since I am interested in MWUs, I give statistics for the length of queries and alternatives in tables 6.1 and 6.2.

#### Table 6.1: Share of queries and alternatives with different lengths in SweHP560

<table>
<thead>
<tr>
<th>Length</th>
<th>Queries</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Share</td>
<td>Count</td>
</tr>
<tr>
<td>1</td>
<td>89.1%</td>
<td>499</td>
</tr>
<tr>
<td>2</td>
<td>5.7%</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>3.6%</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>1.2%</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>0.2%</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.2%</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

Most of the queries are actually single words (89.1%), but as many as 42.7% (239 queries) of them have at least one multi-word alternative. The queries contain in total 662 word tokens divided into 610 word types. Counting both queries and alternatives, we end up with 5126 tokens and 3379 types.
Table 6.2: Overview of query and alternative lengths in SweHP560

<table>
<thead>
<tr>
<th></th>
<th>Queries</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>560</td>
<td>2880</td>
</tr>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Average</td>
<td>1.18</td>
<td>1.59</td>
</tr>
<tr>
<td>StdDev</td>
<td>0.59</td>
<td>1.02</td>
</tr>
</tbody>
</table>

6.4 Evaluation with a synonym test

This evaluation is rather straightforward. I choose to just calculate the percentage of correct answers. The result may be compared to a baseline that chooses randomly among the five possible answers, with an expected result of 20% (112 correct answers out of the 560). Note that for this evaluation to be relevant at all, I assume that the gold standard is correct and that a well functioning LSVM has the capability to decide which one of the alternatives that is the synonym. Strong similarity obtained by LSVMs is absolutely not the same as synonymy, but hopefully synonymy implies strong similarity even if the reverse is often not true. One case when the LSVM has great difficulties to find the right answer is when one or more terms in the query or alternatives are not present in the training data. But this property in the evaluation is good since a really good LSVM is supposed to have a large vocabulary that contains all important terms.

The size of the test, 560 queries, may sound rather small, but many earlier experiments (Turney, 2001; Landauer and Dumais, 1997) based on the TOEFL test (“The Test Of English as a Foreign Language”) use a set of only 80 queries. Whether the size is large enough or not depends on how big the score differences have to be to be statistically significant. More discussion on this can be found in chapter 8.

A potential problem when using a synonym test for evaluation of an LSVM is that the LSVM will identify not just synonyms as similar, but also words related in other ways, like for example antonyms, different forms of the same stem or derivations. I do not worry to much about this since the tests are designed to test humans, and it is unlikely that humans think that an inflected form of a query word, or that a word with the wrong part-of-speech is the answer.
Chapter 7

Document retrieval based evaluation

This chapter describes the document retrieval-based evaluation task. A potential problem with MAP, the evaluation metric used in the TREC environment, is that it was designed for the Cranfield environment where all documents are relevance-judged for each topic, which is not the case in TREC and CLEF. Inspired by bpref (Buckley and Voorhees, 2004), I define a new evaluation metric \textit{RankEff}, which is aware of the case when not all returned documents are relevance-judged. I use data from earlier TREC collections, including rank lists from a large set of systems, and compare the results to the often used MAP and the new bpref-metrics (Buckley and Voorhees, 2004). The metrics are theoretically investigated and experimentally compared with aspect to system-ranking stability for incomplete evaluation data, significance levels, etc. Some of the experiments in this chapter were performed together with Per Ahlgren. The results were also presented in Ahlgren and Grönqvist (2006a) and Ahlgren and Grönqvist (2006b), for example, those in section 7.6.

7.1 Introduction

As mentioned in section 5.3.1, a typical experiment in information retrieval (IR) makes use of a test collection. Such a collection consists of three components:

- A database $D$ of documents
- A set $T$ of topics (information needs, written in natural language)
- For each topic in $T$, a non-empty set of relevant documents from $D$

In the Cranfield evaluation model (Voorhees, 2002), which prescribes the use of test collections, a completeness assumption is made: for each topic $t \in T$, each document $d \in D$ is judged for relevance in relation to $t$ (Cleverdon, 1991). In the normal case, though, $D$ contains a large number of documents, and it is practically impossible to judge each document for each topic. Instead, the documents that belong to a proper subset of $D$ are relevance-judged. This will lead to three types of documents in $D$:
- Relevance-judged documents labeled as relevant
- Relevance-judged documents labeled as irrelevant
- Not relevance-judged documents that may be either relevant or irrelevant

When a new system is tested against a TREC collection, it may be the case that documents that have not been relevance-judged with respect to $t$, i.e., documents that do not belong to $P_t$, are retrieved. Such documents are assumed to be irrelevant when a traditional metric like precision is applied. This assumption is less satisfactory, since it may favour systems that participated in the pooling process.

Issues in retrieval evaluation have received fairly much attention in the literature (Hull, 1993; Keen, 1992; Sanderson and Zobel, 2005; Tague-Sutcliffe, 1992), and alternatives to traditional metrics have been proposed, for example in Järvelin and Kekäläinen (2002). In this chapter, I treat the two metrics of retrieval effectiveness: $bpref$ (Buckley and Voorhees, 2004) and $RankEff$, which do not take into account documents that have not been relevance-judged. These metrics are compared, both theoretically and experimentally. In addition, I experimentally compare the two metrics to a well-known evaluation metric, mean uninterpolated average precision (MAP).

### 7.2 Inventing new metrics

The evaluation metrics developed in this thesis are useful for all kinds of document retrieval, as long as the system has the ability to rank any set of documents by relevance to any given topic. Most IR-models have this ability. The only one among the models I have discussed in section 2.2 that does not give the needed ranking ability is the ordinary (not extended) Boolean model.

For a small subset of the data collection (presented in section 7.6.1) there is a fine-grained scale of relevance, similar to the scale used for the Swedish data, presented in section 11.2.3.1. I have chosen to use just irrelevant and relevant (marginally, fairly, and highly relevant in the IR test suite counted together), because this binary scale is what is used in the rest of the data, including the Swedish CLEF-data used in part III of this thesis.

#### 7.2.1 The share of non-judged documents

Table 7.11 and table 7.12 show that the total number of unjudged documents is high enough to give huge differences in evaluation values depending on the relevance for the unjudged documents. For a more detailed view, figure 7.1, where I exemplify with TREC-12 data, the share of documents that

---

1See section 5.3.2 for information about the TREC environment.
7.2 Inventing new metrics

![Graph showing the share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-12.](image)

Figure 7.1: Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-12

have not been judged is as large as 80% at ranking position 1000, as an average over all runs and all 100 topics. Just after rank 300 the number of unjudged documents passes the number of judged ones. For TREC-12, the top 125 documents were used during the pooling process, i.e. $dcv$ is 125. Despite that, the share of unjudged documents at ranking position 125 is about 13%. Even at ranking position 1, there are documents that have not been judged. This should not happen theoretically, but some files sent in to TREC too late or in bad file formats could have caused it.

One concern with unjudged documents is that runs that did not contribute to the pools, may retrieve unjudged relevant documents, and thereby

---

2 For completeness, I have included figure 7.2 and figure 7.3 showing the same kind of graphs for the TREC-8 and TREC-10 data. The only noticeable difference is that the strong rise at $dcv$ for the share of unjudged documents is found at 100 documents for TREC-8 and TREC-10.

3 $dcv$ for TREC-12 is 125 and for TREC-8 and TREC-10 it is 100.
Figure 7.2: Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-8

be treated unfairly during evaluation. Although it has been argued on empirical grounds that the TREC collections are not biased against such runs (Voorhees, 2002; Zobel, 1998), however, I believe that large shares of unjudged documents, especially at low ranking positions (but above the dcv), is a sufficient motivation for the construction of evaluation metrics that do not take unjudged documents into account.

7.2.2 Why we need new metrics
Except for the \textit{bpref} (section 5.3.3) and \textit{RankEff} metrics (section 7.2.3), we have looked at some other possibilities. Before we look at the metrics, let us try to find out what we really aim for. One major problem with the evaluation of document retrieval systems is that it is not entirely clear what we want to measure, but a few properties are almost clear:

- All users prefer relevant documents over irrelevant ones.
7.2 Inventing new metrics

![Graph](image)

Figure 7.3: Share of relevant, judged, and unjudged documents at different ranking positions, as an average over all systems and topics in TREC-10

- A specific ranking of a set of documents for a topic becomes better if a relevant document $d_r$ and an irrelevant document $d_n$ swaps places iff $\text{Rank}(d_r) > \text{Rank}(d_n)$

- A nice property for an evaluation metric is that the values should be bounded to an interval, for example $[0, 1]$

Some other properties are not that clear:

1. Is it really possible to design a one-dimensional metric that compares different ranked lists of relevant and irrelevant documents?

2. Should the score change if two relevant (or irrelevant) documents swap places?

3. Should the score change if one relevant document moves up in the list, and another one moves down the same number of steps?
4. If the metric gives a number in the interval $[0, 1]$, it will be possible to compare results between topics. If we compare the relevance-judgement lists $r_1$ and $r_2$ of two topics $t_1$ and $t_2$ from a system, and it turns out that the first part of $r_2$ is the same as $r_1$ as a whole, but $r_2$ continues with a number of irrelevant documents, should then $r_2$ get a better value than $r_1$?

5. Are some relevant documents more relevant than other ones, i.e. should we use a graded scale of relevance?

Let us also assume that number 1 is true. The kinds of evaluation in TREC and CLEF rely on that assumption and comparing systems objectively becomes more difficult otherwise. I will not go deeper into this question, but just assume that it is meaningful to define a general metric for ranking effectiveness.

The answer to question 2 should be “no”, as long as we are using binary relevance-judgements and there is no more information about the swapped documents. Question 3 is much more difficult to answer, since the answers depend on the information need, but let us say “no” in the general case. It seems sound to answer “yes” to question 4 since $r_2$ is more difficult to achieve by chance. Question number 5 should be answered with “probably”, but since it depends both on the type of information need and document properties that are difficult to find (as long as we only have access to binary relevance data), we will assume the answer to be “no”. If we have different levels of relevance, new problems will arise, like how big the difference in relevance should be in numbers.

Since I am using existing test suites, the easiest approach—to look at top lists of different lengths for retrieval results for various queries and count the number of relevant documents—is not really fair. The problem is that except for the known relevant and known irrelevant documents for each query, we now have a bunch of not relevance-judged documents. Those documents constitute the missing information in what (Buckley and Voorhees, 2004) refer to as “Incomplete information”. We do not want to count non-judged documents as irrelevant, instead we want to base the evaluation metric on the manually judged documents only. One reason not to rely on the pooling in the experiments is that the systems used for the pooling, in at least one of the test suites, were only based on keyword search with different term weighting schemes, which may give very different top lists compared to an LSI based system. But this is a general problem since a new system that finds relevant documents using different methods than the existing systems will get worse scores, given that the old systems used during pooling did not

---

4 At first glance this may seem totally uninteresting, but in some cases it could be useful to be able to find out which are hard or easy to find relevant documents for.

5 This can happen if the topic corresponding to $r_2$ has a larger number of irrelevant documents than the one that gave $r_1$ but both have the same number of relevant documents. Let us here think of the task as ranking the full pools and not just a top-list of a fixed length.
find all the relevant documents—especially those found by the new system. It is reasonable to think that new innovative systems will find many documents that are not found with the old methods. This favouring of systems that find the same set of relevant documents as the old systems, will slow down the development of better systems, so there is a need to find better ways to use the IR test suites than the traditional evaluation metrics that count non-judged as irrelevant.

Before I go on with new evaluation metrics, we should note that the metrics used in the TREC-environment and the metrics designed in this thesis do not use the content of documents—just the relevance property.

### 7.2.3 The RankEff metric

Buckley & Voorhees’ metric \( bpref \) does handle incomplete data, but it has weaknesses. I have so far developed one new metric: \( \text{RankEff} \), which is not yet tested very much. Equation 7.1 defines the \( \text{RankEff} \) metric for a topic \( t \) and an IR-model \( M \). Given a document \( d_i \in R \), \( N(d_i, t, M) \) is the number of documents \( d \) such that \( d \in N \), and \( \text{Rank}(d, t, M) > \text{Rank}(d_i, t, M) \). \( \text{Rank}(d_i, t, M) \) is the ranking function that given a topic, an IR-model, and a document, returns the position for the document’s position in a ranking list.

\[
\text{RankEff}(t, M) = \frac{1}{rn} \sum_{i=1}^{r} N(d_i, t, M) \tag{7.1}
\]

\( \text{RankEff} \) is similar to \( bpref \) but it uses more of the relevance-judgements in many cases. Table 7.1 illustrates the calculation of \( \text{RankEff} \) for a small example with three relevant and seven irrelevant documents. For this case equation 7.1 gives a sum over the three relevant documents. Each value for \( N(d_i, t, M) \) is included in the table. The result for that example is:

\[
\frac{1}{3*7} \sum_{i=1}^{3} N(d_i, t, M) = \frac{1}{21} (6 + 4 + 4) = \frac{2}{3}
\]

To get an idea of what the values say, we can note that if all known relevant documents are ranked better than all known irrelevant ones, we get 1. The opposite case leads to 0. The metric gives the mean number of known irrelevant documents that have a higher ranking than a known relevant document, in relation to the number of known irrelevant documents, which could easily be verified for the extreme cases 0 and 1, by looking at equation 7.1 and the example in table 7.1

### 7.2.4 The RankSum metric

A very straightforward way to define a metric of how close we are to an optimal ranking list of the relevance-judged documents, is to sum the ranks.
Table 7.1: Illustrative example of a \textit{RankEff} calculation

<table>
<thead>
<tr>
<th>Rank</th>
<th>Doc</th>
<th>Relevance</th>
<th>(N(d_i, t, M))</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(d_{\text{non}1})</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>(d_{\text{rel}1})</td>
<td>+</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>(d_{\text{non}2})</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>(d_{\text{non}3})</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>(d_{\text{rel}2})</td>
<td>+</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>(d_{\text{rel}3})</td>
<td>+</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>(d_{\text{non}4})</td>
<td>-</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>(d_{\text{non}5})</td>
<td>-</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>(d_{\text{non}6})</td>
<td>-</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>(d_{\text{non}7})</td>
<td>-</td>
<td>-</td>
<td>14</td>
</tr>
</tbody>
</table>

for all known relevant documents, and relate that sum to the possible maximum and minimum values. The minimum would be when all known relevant documents have rank 1 to \(r\), and the maximum if they are all at the bottom (from \(p-r+1\) to \(p\)). Note that the documents that are not relevance-judged are removed. \(RS_{\text{min}}\) and \(RS_{\text{max}}\) gives the minimum and maximum value for the sums of ranks for the known relevant documents, given a topic.

\[
RankSum_1(t, M) = 1 - \sum_{i=1}^{r} \frac{Rank(d_i, t, M) - RS_{\text{min}}(p, r)}{RS_{\text{max}}(p, r) - RS_{\text{min}}(p, r)}
\]  

(7.2)

Since \(RS_{\text{min}}(p, r)\) and \(RS_{\text{max}}(p, r)\) are easily calculated, we can rewrite equation 7.2 as:

\[
RankSum_1(t, M) = 1 - \frac{\sum_{i=1}^{r} \text{Rank}(d_i, t, M) - \frac{r(r+1)}{2}}{\frac{r(p+p-r+1)}{2} - \frac{r(r+1)}{2}}
\]

And finally we have:

\[
RankSum_1(t, M) = 1 - 2\sum_{i=1}^{r} \text{Rank}(d_i, t, M) - r^2 - r
\]

(7.3)

\(RankSum\) may also be defined in terms of irrelevant document rankings:

\[
RankSum_2(t, M) = \frac{\sum_{i=r+1}^{p} \text{Rank}(d_i, t, M) - RS_{\text{min}}(p, n)}{RS_{\text{max}}(p, n) - RS_{\text{min}}(p, n)}
\]

And simplified to:

\[
RankSum_2(t, M) = 2\sum_{i=r+1}^{p} \text{Rank}(d_i, t, M) - n^2 - n
\]

(7.4)
We may rewrite \( \text{RankSum}_2 \) using the fact that:

\[
\sum_{i=r+1}^{p} \text{Rank}(d_i, t, M) = \frac{p(p+1)}{2} - \sum_{i=1}^{r} \text{Rank}(d_i, t, M)
\]

\[
\text{RankSum}_2(t, M) = \frac{2 \sum_{i=r+1}^{p} \text{Rank}(d_i, t, M) - n^2 - n}{2np - 2n^2} = \frac{p(p+1) - 2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M) - n^2 - n}{2np - 2n^2} = \frac{p(p+1) - 2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M) - (p-r)^2 - (p-r)}{2p(p-r) - 2(p-r)^2} = \frac{p^2 + p - p^2 - r^2 + 2rp - p + r - 2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M)}{2p^2 - 2rp - 2p^2 - 2r^2 + 4rp} = \frac{2rp - r^2 + r - 2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M)}{2rp - 2r^2} = \frac{2rp - r^2 + r - 2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M)}{2rp - 2r^2} = 1 - \frac{2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M) - r^2 - r}{2rp - 2r^2} = \text{RankSum}_1(t, M)
\]

So \( \text{RankSum}_1 \) and \( \text{RankSum}_2 \) are actually identical, but we have the choice to define \( \text{RankSum} \) based on either the relevant (equation 7.3) or irrelevant (equation 7.4) documents. How does \( \text{RankSum} \) then compare to \( \text{RankEff} \)? Let us rewrite the \( \text{RankEff} \) definition from equation 7.1. We need to get rid of \( N(d_i, t, M) \) and if we assume that the relevant documents \( d_1, ..., d_r \) are numbered such that \( d_1 \) is the highest ranked relevant document according to \( M \), then \( d_2 \) and down to \( d_r \), which is the lowest ranked relevant document. Then we can actually write \( N(d_i, t, M) = n + i - \text{Rank}(d_i, t, M) \). Let us use this to rewrite \( \text{RankEff} \) like this:

\[
\text{RankEff}(t, M) = \frac{1}{rn} \sum_{i=1}^{r} N(d_i, t, M) = \frac{1}{rn} \sum_{i=1}^{r} (n + i - \text{Rank}(d_i, t, M)) = \frac{1}{rn} \sum_{i=1}^{r} n + \frac{1}{rn} \sum_{i=1}^{r} i - \frac{1}{rn} \sum_{i=1}^{r} \text{Rank}(d_i, t, M) = 1 + \frac{1}{rn} \left( \frac{r(r+1)}{2} - \sum_{i=1}^{r} \text{Rank}(d_i, t, M) \right) = 61
\]
Document retrieval based evaluation

\[
1 - \frac{\sum_{i=1}^{r} \text{Rank}(d_i, t, M)}{rn} = 1 - \frac{2 \sum_{i=1}^{r} \text{Rank}(d_i, t, M) - r^2 - r}{2rn} = \text{RankSum}_1(t, M)
\]

I have now shown that

\[
\text{RankSum}_1(t, M) = \text{RankSum}_2(t, M) = \text{RankEff}(t, M)
\]

for all systems \( M \) and topics \( t \), which means that they are identical. I will use the name \( \text{RankEff} \) for them from now.

### 7.2.5 \( \text{bpref} \) compared to \( \text{RankEff} \)

Let us look at \( \text{RankEff} \) for the special case when \( r = n = \frac{p}{2} \):

\[
\text{RankEff}(t, M) = 1 - \frac{\sum_{i=1}^{r} N(d_i, t, M)}{r(p - r)} = 1 - \frac{\sum_{i=1}^{r} N(d_i, t, M)}{r^2} = \frac{1}{r} (r - \sum_{i=1}^{r} \frac{N(d_i, t, M)}{r}) = \frac{1}{r} \sum_{i=1}^{r} \left( 1 - \frac{N(d_i, t, M)}{r} \right) = \frac{1}{r} \sum_{i=1}^{r} \left( 1 - \frac{N_r(d_i, t, M)}{r} \right) = \text{bpref}(t, M)
\]

So in this special case when \( \text{bpref} \) actually uses all available information, the metric is identical to \( \text{RankEff} \).

In the remainder of this section I compare \( \text{bpref} \)-10, rather than \( \text{bpref} \), to \( \text{RankEff} \). One may ask if \( \text{bpref} \)-10 and \( \text{RankEff} \) are equivalent in the sense that they always agree with respect to the order of two compared models. More formally, one may ask if the following statement holds:

\[
\text{bpref}-10(M_1, t) < (>,=) \text{bpref}-10(M_2, t)
\]

if and only if

\[
\text{RankEff}(M_1, t) < (>,=) \text{RankEff}(M_2, t)
\]

However, the statement above is not true, which is shown by the following counterexample. Assume that \( n = 30 \) and \( r = 2 \), for a given topic \( t \). Then \( n - r = 28 = |I| \) = the number of known irrelevant documents for \( t \). Assume that the methods \( M_1 \) and \( M_2 \) rank documents according to the data in Table 7.2, where "+" indicates a known relevant document, "−" a known irrelevant document, and "..." stands for the positions 16–28 in the lists of retrieved documents. Further, positions 16–28 are assumed to contain, for both \( M_1 \) and \( M_2 \), known irrelevant documents.

With the data in Table 7.2, where \( r \) and \( n \) is the number of relevant and irrelevant documents, we obtain the following for \( \text{bpref} \)-10 and the two methods:
7.2 Inventing new metrics

Table 7.2: Rankings associated with two hypothetical systems where the resulting ranking lists are identical except for one relevant document

<table>
<thead>
<tr>
<th>Id</th>
<th>Ranking list</th>
<th>r</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>+ − − − − − − − − − − − + − − − −</td>
<td>2</td>
<td>&gt;14</td>
</tr>
<tr>
<td>$M_2$</td>
<td>+ − − − − − − − − − − − − − +</td>
<td>2</td>
<td>&gt;14</td>
</tr>
</tbody>
</table>

$$b_{pref-10}(M_1,t) = \frac{1}{2} \left( 1 - \frac{0}{12} + 1 - \frac{12}{12} \right)$$

$$= 0.500$$

$$= b_{pref-10}(M_2,t)$$

For $RankEff$ we obtain:

$$RankEff(M_1,t) = \frac{28 + 16}{2 \times 28} \approx 0.786$$

and

$$RankEff(M_2,t) = \frac{28 + 0}{2 \times 28} = 0.500$$

Thus, $RankEff(M_1,t) > RankEff(M_2,t)$. Given the data of the example, $M_1$ obviously performs better than $M_2$ with respect to rank effectiveness, and it is desirable that a metric for rank effectiveness detects this.

$RankEff$ has some appealing properties in relation to $b_{pref-10}$:

- It uses more information, since each known irrelevant document is taken into consideration.
- It can handle data sets with any number of relevant and irrelevant documents, except if $R = \emptyset$ or $N = \emptyset$, leading to a division by zero.
- It handles topics with a small number of relevant documents better than $b_{pref-10}$ in the sense that unreasonably large differences in measurement values between models are prevented.

7.2.6 A weighted RankSum metric

In some cases we want to base the score more on the top documents in the ranking list. The $RankEff$ metric as well as $b_{pref}$ will for example give the
same score for the sequences E7 and E8 of relevant (+) and irrelevant (−) documents in table 7.3.\(^6\)

Table 7.3: Rankings associated with two hypothetical systems with the same number of relevant and irrelevant documents

<table>
<thead>
<tr>
<th>Id</th>
<th>Ranking list</th>
<th>r</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>E7</td>
<td>+ + + + − − − − − − − − + + + + + + + + − − − −</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>E8</td>
<td>− − − − + + + + + + + + + + − − − −</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

The first ranking is very much preferred if you need just four relevant documents. To catch this preference, we need another kind of metric. This weighted RankSum (WRS)\(^7\) gives a higher weight the higher the document is ranked. \(RS_{\text{max}}^2(p, r)\) and \(RS_{\text{min}}^2(p, r)\) denote the maximum and minimum values for the summed squares of the relevant documents’ rankings, and in the formula they guarantee (just as for \(RS\)) that the value stays in the interval [0, 1]

\[
WRS(t, M) = \frac{\sum_{i=1}^{r} (p+1 - \text{Rank}(d_i, t, M))^2 - RS_{\text{min}}^2(p, r)}{RS_{\text{max}}^2(p, r) - RS_{\text{min}}^2(p, r)} \quad (7.5)
\]

Equation 7.5 may be rewritten using \(p\) and \(r\):

\[
WRS(t, M) = \frac{\sum_{i=1}^{r} (p+1 - \text{Rank}(d_i, t, M))^2 - \sum_{i=1}^{r} i^2}{\sum_{i=p-r+1}^{p} i^2 - \sum_{i=1}^{r} i^2} \quad (7.6)
\]

The kind of sums in equation 7.6 can be simplified using:

\[
\sum_{i=1}^{n} i^2 = \frac{2n^3 + 3n^2 + n}{6}
\]

we may rewrite equation 7.6 as:

\[
WRS(t, M) = \frac{\sum_{i=1}^{r} (p+1 - \text{Rank}(d_i, t, M))^2 - \frac{2r^3 + 3r^2 + r}{6} - \frac{2(p-r)^3 + 3(p-r)^2 + p-r}{6} - \frac{2r^3 + 3r^2 + r}{6}}{2p^3 + 3p^2 + p - 2(p-r)^3 + 3(p-r)^2 + p-r - 2r^3 + 3r^2 + r}
\]

Expand to:

\[
WRS(t, M) =
\]

\(^6\)I have chosen the names E7 and E8 for these ranking lists to match the names in the tables in section 7.4

\(^7\)I call it weighted RankSum rather than weighted RankEff since the definition is based on the idea of a sum over ranking positions just as RankSum. Note that the unweighted RankEff and RankSum metrics are equivalent (see section 7.2.4).
7.2 Inventing new metrics

\[
6 \sum_{i=1}^{r}(p + 1 - \text{Rank}(d_i, t, M))^2 - 2r^3 + 3r^2 + r \\
2p^3 + 3p^2 + p - 2(p - r)^3 + 3(p - r)^2 + p - r - 2r^3 - 3r^2 - r
\]

And finally what I will use as a definition of weighted RankSum:

\[
\text{WRS}(t, M) = \frac{6 \sum_{i=1}^{r}(p + 1 - \text{Rank}(d_i, t, M))^2 - 2r^3 + 3r^2 + r}{6(p^2r - pr^2 + pr - r^2)}
\] (7.7)

The WRS-metric will prefer the ranking list E7 in favour of E8 from table 7.3. As we can see in table 7.5 (section 7.4), WRS for E7 is 0.618 compared to 0.382 for E8.

7.2.7 Distinguishing between documents containing different facts

Evaluation metrics based only on lists of relevance values for documents at each position in the retrieved list is what we have looked at so far. Another aspect of a document retrieval system is whether it retrieves documents that covers all kinds of documents relevant to the topic. What if we have a topic \( t \) that requests a set of facts \( F = f_1, ..., f_m \) and a set of relevant documents \( d_1, ..., d_n \) containing these facts? Some document may contain only one fact \( f_i \) and some may contain more than one fact. A user that wants to find the facts \( F \) would be much more happy with a system that returns a list of documents containing one single relevant document with all the facts in \( F \) than a list of hundreds of relevant documents containing only the answer to one fact in \( F \). It is easy to find examples of topics where we have this scenario, for example topic 137 from CLEF in figure 7.4.\(^8\)

- **Topic number**: C137
- **Title**: International beauty contests
- **Description**: Search for the names of the winners in international beauty contests during 1994 or 1995
- **Narrative**: Relevant documents should mention the name of men or women that have won an international beauty or charm contests during 1994 or 1995. Contest for body builders are not relevant.

Figure 7.4: Translated version of topic C137 from CLEF

Suppose that the document collection contains 10 names of winners in international beauty contests (10 distinct facts) and a total number of 200 relevant documents. If one of the relevant documents, \( d_j \), contains all the 10 winners, a very useful answer list from a retrieval system would be the

---

\(^8\)Topic 137 from CLEF is translated from Swedish.
retrieved list starting with \( r_1 \) and continued by any set of documents. If they are relevant we may get the facts known from \( r_1 \) confirmed but we do not get any new facts, so even the list: \( r_1, r_2, ..., r_m \), where \( r_2, ..., r_m \) gives all the 10 existing facts. Another system \( M_2 \) may return 100 relevant lists where all of them contain the same person. System \( M_2 \) will then get a top score by evaluation metrics like \( MAP \), \( bpref \) and \( RankEff \), while it is not very useful for a user with the particular information need described by topic 137. System \( M_1 \) on the other hand gives exactly what the user needs, but gets a low \( MAP \), \( bpref \) and \( RankEff \) score.

We have seen that for this particular topic and a hypothetical document collection and two systems, metrics like \( MAP \), \( bpref \) and \( RankEff \) are not very successful. However, this hypothetical situation is so extreme that an evaluation metric that actually judges the system in the same way as a user normally does, would require more or less full parsing of the topic, which is not very convenient for a document retrieval evaluation metric. Such an evaluation metric must be able to parse different kinds of domains, and therefore be much more difficult to build than the evaluated IR-systems. Cases like this are hopefully not that common so they will not affect the results in TREC or CLEF. If we in the future want to design a competition aware of the problem with distinct facts relevant to a topic, then probably the manual judgement has to include the task of identifying the facts in each relevant document, which is not really document retrieval but information extraction.

One aspect that could possibly be included in an evaluation metric used to find the best system in a TREC-like environment, is to try to calculate the topical difference between the retrieved relevant documents for each system. A higher difference should then give a higher score. A negative effect of such a metric is that it might not be neutral in its judgement. Systems using similar kinds of algorithms as the metric could be favoured, which is not good at all. However, that kind of evaluation metrics and an analysis of them will not fit into this thesis.

### 7.3 Practical adjustments of evaluation metrics

In the TREC environment, retrieval effectiveness is evaluated at 1000 retrieved documents. It may be the case, and often is, that some known relevant documents are not among the top 1000 retrieved, for a given system and a given topic. Because of this, the definitions of the metrics need to be adjusted to fit practical situations. However, I only give definitions of \( bpref \)-10 and \( RankEff \), since I compared the latter to the former (and not to the coarser variant \( bpref \)) in the experiments (section 7.6).

Let \( R' \) be the set of retrieved known relevant documents for a model \( M \) and a topic \( t \in T \), and let \( r' \leq r \) be the number of documents in
7.4 Comparison of metrics using made-up examples

Let $I_{10+r}(d'_i)$ be the number of documents $d$ such that $d \in I_{10+r}$ and $\text{Rank}(d, t, M) < \text{Rank}(d'_i, t, M)$, where $d'_i \in R'$. Finally, let $I(d'_i)$ be the number of documents $d$ such that $d \in I$ and $\text{Rank}(d, t, M) > \text{Rank}(d'_i, t, M)$, where $d'_i \in R'$.

I then give the following practical definitions of $b\text{pref}-10$ and $\text{RankEff}$:

$$b\text{pref}-10(M, t) = \frac{1}{r} \sum_{i=1}^{r'} 1 - \frac{I_{10+r}(d'_i)}{10 + r} \tag{7.8}$$

$$\text{RankEff}(M, t) = \frac{\sum_{i=1}^{r'} I(d'_i)}{r(n-r)} \tag{7.9}$$

A system that retrieves only a small number, say $k$, of known relevant documents at the first $k$ ranks, receives a smaller value on the two metrics than a system that retrieves a large number, say $l > k$, on the $l$ first ranks. This is a desirable property, which the metrics have in common with $\text{MAP}$. $\text{MAP}$ is a metric that has been shown to be relatively stable with respect to capacity to distinguish between systems (Buckley and Voorhees, 2000).

Note that the two metrics, just like $\text{MAP}$, give a non-optimal value when not all known relevant documents have been retrieved even, if all retrieved relevant documents are ranked better than all retrieved irrelevant documents. Further, equation 7.8 and equation 7.9 define the same functions as equation 5.6 and equation 7.1 respectively, given that $r' = r$.

7.4 Comparison of metrics using made-up examples

To get a view of how the different metrics handle some special cases, I have defined a set of made-up test examples. These examples serve as a complement to the few earlier examples that I have looked at during the definition of $\text{RankEff}$ and comparison to $b\text{pref}$.

7.4.1 The tested metrics

I have included five metrics in this comparison:

1. $b\text{pref}$ (see section 5.3.3.5, equation 5.6)
2. $b\text{pref}-10$ (see section 5.3.3.5 equation 5.7)
3. $\text{RankEff}$ (section 7.2.3, equation 7.1) which is identical to $\text{RankSum}_1$ (section 7.2.4, equation 7.3) and $\text{RankSum}_2$ (section 7.2.4, equation 7.4).

---

9I use $R'$ and $r'$ in contrast to $R$ and $r$ to stress that $R'$ is the set of retrieved known relevant documents while $R$ is the set of all known relevant documents for the current topic.

10For $d, d' \in D$, I define $\text{Rank}(d, t, M) > \text{Rank}(d', t, M)$ as true if $d$ is not retrieved and $d'$ is retrieved.
Table 7.4: Some made-up examples that should test the behaviour for the different evaluation metrics

4. Weighted ranksum (WRS), see section 7.2.6, equation 7.7

5. Mean uninterpolated average precision (MAP), see section 5.3.3.3, equation 5.5

Note that both $b_{pref}$ and $b_{pref}-10$ are included as well as RankEff and the weighted ranksum (WRS).

The following made-up or real examples will show how well the metrics handle different kinds of data. The made-up examples test how they handle short ranking lists where we have more or less extreme distributions of relevant documents. We will also see the difference when one or very few relevant documents move around in the lists.

### 7.4.2 Results for made-up examples

First, to get an idea of how the metrics work in practice, I will take a look at the small toy example (table 7.4) and calculate the metrics. Note that in the tables in this chapter ‘+’ means relevant and ‘−’ is irrelevant. Obviously, these tables only contain known relevant and irrelevant documents. The results will be found in table 7.5.

It is difficult to say what is right or wrong in these results. In many cases all metrics give roughly the same results, which is not unexpected since they are all designed to catch ranking effectiveness based on a ranked list of relevance-judgements, but we can also note a few differences. Here are some comments to the results in table 7.5.

<table>
<thead>
<tr>
<th>Id</th>
<th>Ranking list</th>
<th>$r$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>+ + − −</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E2</td>
<td>− − + +</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E3</td>
<td>+ + − − + + − −</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>E4</td>
<td>+ − + − − − − − −</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E5</td>
<td>+ + − − − − − − − + +</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>E6</td>
<td>− − + + − − − − + − +</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>E7</td>
<td>++ + + − − − − − + + + +</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>E8</td>
<td>− − − − + − + + + + + − − − − −</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>E9</td>
<td>− + + +</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>E10</td>
<td>− − − + + − − − − − − − − − − − − − − −</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>E11</td>
<td>+ − + − + − − − − − − − − − − − − − − − −</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>E12</td>
<td>+ − − − − − − − − − − − − − − − − − − − +</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>E13</td>
<td>− + + − − − − − − − − − − + + + + + − +</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>E14</td>
<td>− + + − − − − − − − − + +</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
7.4 Comparison of metrics using made-up examples

Table 7.5: Results for the made-up examples
* The cases marked with ‘*’ actually violates the definition of \textit{bpref} or \textit{bpref}-10 since the number of irrelevant documents is too small. The metric is calculated with the present number of irrelevant documents.

<table>
<thead>
<tr>
<th>Id</th>
<th>\textit{bpref}</th>
<th>\textit{bpref}-10</th>
<th>\textit{RankEff}</th>
<th>\textit{WRS}</th>
<th>\textit{MAP}</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>1.000</td>
<td>1.000*</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>E2</td>
<td>0.000</td>
<td>0.833*</td>
<td>0.000</td>
<td>0.000</td>
<td>0.417</td>
</tr>
<tr>
<td>E3</td>
<td>0.444</td>
<td>0.872*</td>
<td>0.667</td>
<td>0.622</td>
<td>0.667</td>
</tr>
<tr>
<td>E4</td>
<td>0.750</td>
<td>0.958*</td>
<td>0.750</td>
<td>0.750</td>
<td>0.833</td>
</tr>
<tr>
<td>E5</td>
<td>0.750</td>
<td>0.958*</td>
<td>0.917</td>
<td>0.880</td>
<td>0.833</td>
</tr>
<tr>
<td>E6</td>
<td>0.250</td>
<td>0.679*</td>
<td>0.438</td>
<td>0.404</td>
<td>0.375</td>
</tr>
<tr>
<td>E7</td>
<td>0.500</td>
<td>0.778*</td>
<td>0.500</td>
<td>0.618</td>
<td>0.722</td>
</tr>
<tr>
<td>E8</td>
<td>0.500</td>
<td>0.778*</td>
<td>0.500</td>
<td>0.382</td>
<td>0.490</td>
</tr>
<tr>
<td>E9</td>
<td>0.667*</td>
<td>0.923*</td>
<td>0.000</td>
<td>0.000</td>
<td>0.639</td>
</tr>
<tr>
<td>E10</td>
<td>0.000</td>
<td>0.769</td>
<td>0.769</td>
<td>0.633</td>
<td>0.383</td>
</tr>
<tr>
<td>E11</td>
<td>0.500</td>
<td>0.917</td>
<td>0.929</td>
<td>0.882</td>
<td>0.750</td>
</tr>
<tr>
<td>E12</td>
<td>0.500</td>
<td>0.500</td>
<td>0.500</td>
<td>0.529</td>
<td>0.562</td>
</tr>
<tr>
<td>E13</td>
<td>0.312</td>
<td>0.679*</td>
<td>0.438</td>
<td>0.438</td>
<td>0.408</td>
</tr>
<tr>
<td>E14</td>
<td>0.375</td>
<td>0.679*</td>
<td>0.438</td>
<td>0.471</td>
<td>0.443</td>
</tr>
</tbody>
</table>

- \textit{bpref}-10 does not handle these small artificial examples very well. E2 is the worst possible ranking for two relevant and two irrelevant documents, but it still gets a high score.

- For E1, E2, and E4, all metrics except \textit{bpref}-10 are the same, and the values are what I expect: the best possible list result in a value of 1 and the worst gives 0. A list between the extremes and better than random gives a value between 0.5 and 1

- E5 should have a better score than E4 since that ranking is more difficult to achieve. \textit{WRS} and \textit{RankEff} agree on this but the other metrics do not

- For E11 and E12 \textit{bpref} gives the same value, but all the other metrics agree that E11 is better that E12, which is correct.

- For E10 \textit{bpref} gives score 0.000 which must be seen as a failure, since there are many possible worse ranking lists for these three relevant and 13 irrelevant documents than the one presented.

- E6, E13 and E14 are very similar. Only one + in the beginning and one in the end is moving. They get the same \textit{RankEff} but \textit{WRS} favours the examples where the higher + is moved towards the beginning.

\textit{RankEff} and \textit{WRS} handle all the examples well. They differ slightly since \textit{WRS} puts a higher weight on positions in the beginning of the ranking list.
7.5 Comparison of metrics using real examples

The examples in table 7.4 are all made-up, so some theoretical weaknesses in the metrics may not be a problem in real life examples. This section contains results from the metrics run on real world examples. The ranking lists are produced with an LSVM based system presented in Grönqvist (2005), but in this context I am just interested in how well the different metrics handle the ranking lists, so there is no need for more details about the system parameters. The queries are created just by taking the text from the topic and remove some key phrases like “Documents are relevant only if”.

7.5.1 Examples with few relevant documents

First I have a set of topics with few relevant documents in the table 7.6 and table 7.8 and the results in table 7.7. I would have expected higher scores for topic 120 for the $bpref$ and $bpref$-10, but since the number of relevant documents is very low (only one), they will use very little information. For the examples with low percentage of relevant documents, we get bad results for the $bpref$ metrics. $RankEff$ on the other hand gives an expected high score since the only known relevant documents is ranked among the top 10%.

7.5.2 Examples with a higher number of relevant documents

The third table of real examples show examples with a higher share of relevant documents (table 7.9), and some of them actually have more relevant than irrelevant documents.

The $bpref$ metrics work better now when they get more data to work on, but for topic T21 the result is not very good. Comparing with for example T7, I would like T21 to get a much lower score since the relevant documents are almost evenly distributed over the ranked list. This is because $r > n$, and the $bpref$-metrics are not optimized for this. $RankEff$ gives a score of 0.45 for T21 which is what I expect for this example since the ranking list is slightly worse than a random distribution of the relevant documents.

7.5.3 Stability for each metric

Since it is still difficult to say very much about the quality for each metric, I have performed a small experiment, using a word-based LSVM.\footnote{The examples are taken from the IR test suite presented in section 11.2, but that is not very important at this stage since I am using just individual ranking lists.}

\footnote{I am not interested in comparing LSVMs at this stage, so the model used here is just a standard word-based LSVM. In part III there will be more details about parameter settings for the LSVMs.}
7.5 Comparison of metrics using real examples

<table>
<thead>
<tr>
<th>Id</th>
<th>Ranking list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T120</td>
<td>✔</td>
</tr>
<tr>
<td>T57</td>
<td>✔</td>
</tr>
<tr>
<td>T61</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table 7.6: Some example topics with very few relevant documents

1. Start with the pool for a topic with a fair share of relevant documents, in this case T72, and the ranking from a specific system.

2. Calculate the different metrics for the ranking list.

3. Remove one randomly selected document. It may be relevant or irrelevant.

4. Go back to step 2 if there are documents left.

The procedure will result in a graph with one line per metric. If we for example remove a relevant document at the top position, the metrics normally get a lower value, but if the document was irrelevant, the score increases instead. Since the documents in the pool are more or less arbitrary, I would expect a good relevance metric not to vary too much. The metric is supposed to measure the quality of the ranking and not the pool composition. In figure 7.5 for T72, we can see that RankEff followed by WRS has the most stable curves. Figure 7.6, where all metrics are linearly normalized to start at 1 (by dividing with their start values), shows this even more clearly. \( bpref-10 \) is a little more stable than \( bpref \) but we can see that both \( bpref \)-metrics increase from their start values, which is an unwanted property for a relevance metric.

\[\text{I have chosen T72 because it has a fair number of relevant documents. See table 7.8 for the ranking list for T72 and table 7.7 for statistics.}\]
Table 7.7: Results for the real example topics

<table>
<thead>
<tr>
<th>Id</th>
<th>r</th>
<th>n</th>
<th>bpref</th>
<th>bpref-10</th>
<th>RankEff</th>
<th>WRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T120</td>
<td>1</td>
<td>162</td>
<td>0.000</td>
<td>0.000</td>
<td>0.932</td>
<td>0.870</td>
</tr>
<tr>
<td>T57</td>
<td>2</td>
<td>255</td>
<td>0.500</td>
<td>0.875</td>
<td>0.994</td>
<td>0.988</td>
</tr>
<tr>
<td>T61</td>
<td>5</td>
<td>177</td>
<td>0.480</td>
<td>0.707</td>
<td>0.975</td>
<td>0.953</td>
</tr>
<tr>
<td>T18</td>
<td>9</td>
<td>263</td>
<td>0.099</td>
<td>0.140</td>
<td>0.831</td>
<td>0.710</td>
</tr>
<tr>
<td>T58</td>
<td>12</td>
<td>296</td>
<td>0.361</td>
<td>0.386</td>
<td>0.877</td>
<td>0.795</td>
</tr>
<tr>
<td>T20</td>
<td>18</td>
<td>199</td>
<td>0.441</td>
<td>0.548</td>
<td>0.918</td>
<td>0.860</td>
</tr>
<tr>
<td>T72</td>
<td>16</td>
<td>125</td>
<td>0.301</td>
<td>0.394</td>
<td>0.808</td>
<td>0.709</td>
</tr>
<tr>
<td>T7</td>
<td>40</td>
<td>146</td>
<td>0.797</td>
<td>0.827</td>
<td>0.936</td>
<td>0.909</td>
</tr>
<tr>
<td>T45</td>
<td>67</td>
<td>70</td>
<td>0.631</td>
<td>0.679*</td>
<td>0.647</td>
<td>0.656</td>
</tr>
<tr>
<td>T21</td>
<td>119</td>
<td>50</td>
<td>0.769*</td>
<td>0.787*</td>
<td>0.450</td>
<td>0.509</td>
</tr>
<tr>
<td>T48</td>
<td>132</td>
<td>16</td>
<td>0.993*</td>
<td>0.993*</td>
<td>0.941</td>
<td>0.979</td>
</tr>
</tbody>
</table>

Figure 7.7 and figure 7.8 show T7 and T58 where T7 has a higher share (40 out of 186) of relevant documents. We can see that T7 is easier to handle for the metrics, while T58 is just as difficult as T72, but RankEff and WRS still have stable curves.

### 7.5.4 Combining the result from several topics

If we want to compare models using evaluation results, we definitely want to use more than one topic at a time. The easiest way is to calculate the mean value over the topics, which is meaningful since the metrics developed in this thesis is normalized to the $[0,1]$ interval. We can also calculate the median and standard deviation. Unfortunately the standard deviation will not show the stability of the model, but rather the difference in difficulty for the topics, so it is not optimal to use this standard deviation for a significance score between two different models. I will return to this problem in section 7.6.

### 7.6 Experimental comparison using log-files from TREC

In this section the log-files from the participating systems in three years of TREC are used to find out how well the investigated metrics handle the task of ranking a set of systems competing in TREC.\(^{14}\) bpref and bpref-10 are similar so let us keep the more elaborated bpref-10 and skip the original bpref for now. WRS will also not be included in these experiments.

Error rates associated with the metrics are studied in section 7.6.2, the correlation between system-rankings generated from different metrics when

---

\(^{14}\)See section 7.6.1 for a presentation of the three TREC collections used.
Table 7.8: Some example topics with relatively few relevant documents
Table 7.9: Some example topics with a high percentage of relevant documents

the full pools of judged documents are used (section 7.6.3), and the effects of gradually reducing the pools of judged documents on consistency of absolute average evaluation values and on the stability of system rankings in section 7.6.4. The principal concern is to compare RankEff and bpref-10. However, the two metrics are compared to MAP, heavily used in TREC.

### 7.6.1 Test collections

The experiments used the same test collections that were used in Buckley and Voorhees (2004), TREC-8 (ad hoc task), TREC-10 (Web track) and TREC-12 (robust track). Table 7.10 gives an overview of these three collections. In the last two columns, the mean number of relevant documents per pool and the mean pool size are reported.

Table 7.11 gives, for each test collection, the number of system runs used in the experiments. In this work, distinct runs are considered as two distinct systems. A group that participates in a given TREC may be associated with several runs, and thereby with several systems. Also given in Table 7.11 are the mean number of retrieved documents (in TREC, maximum 1000 per topic), relevant and unjudged documents across the runs. Since the metrics

---

15 The TREC-10 and TREC-12 are called TREC-2001 and TREC-2003, respectively in some publications.
focused on in this work are dependent on the number of retrieved documents, each run that retrieved less than 95% of the maximal number of retrieved documents was excluded. For example, for TREC-8 (with 50 topics) the maximal number of retrieved documents for a run is $50 \times 1000 = 50,000$. Therefore, each TREC-8 run that retrieved less than 47,500 was excluded. Further, each run that did not retrieve any documents for one or more topics was excluded. These exclusion rules are the same as those employed in Buckley and Voorhees (2004). 7 runs each from TREC-8 and TREC-12 were excluded, while 21 runs were excluded from TREC-10.

Table 7.11 shows that on the average (over all runs and topics) there is a relatively large number of retrieved documents that have not been relevance-judged, irrespective of which of the three TREC-10s considered. If we only take
Figure 7.6: Normalized scores for a reduced ranking list of topic T72—a topic with a medium share of relevant documents

Figure 7.7: Normalized scores for a reduced ranking list of topic T7—a topic with a high share of relevant documents
### 7.6 Experimental comparison using log-files from TREC

![Graph showing normalized scores for a reduced ranking list of topic T58—a topic with a few known relevant documents](image)

**Figure 7.8**: Normalized scores for a reduced ranking list of topic T58—a topic with a few known relevant documents

In the top 100 documents into account, there is still a noticeable number of unjudged documents (Table 7.12).

#### Table 7.11: Averages for the number of retrieved, relevant, and unjudged documents over all systems and TREC

<table>
<thead>
<tr>
<th>TREC</th>
<th>Runs</th>
<th>Top 1000 Averages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Retrieved</td>
</tr>
<tr>
<td>TREC-8</td>
<td>122</td>
<td>997.5</td>
</tr>
<tr>
<td>TREC-10</td>
<td>76</td>
<td>989.6</td>
</tr>
<tr>
<td>TREC-12</td>
<td>71</td>
<td>999.5</td>
</tr>
</tbody>
</table>

#### 7.6.2 Error rates

One issue in retrieval evaluation is the following situation: Two topic sets of the same size disagree as to which of two systems is the better. Let $EM$ be an evaluation metric. An error rate in this respect can be obtained empirically by comparing the mean $EM$ values of two systems on two disjoint topic sets of the same size $z$. If one of the systems performs better than the other for one of the two sets but worse for the other set, a swap has
Table 7.12: Averages for the top-100 retrieved documents over all systems
and topics

<table>
<thead>
<tr>
<th>TREC</th>
<th>Runs</th>
<th>Top 100 Averages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Retrieved</td>
</tr>
<tr>
<td>TREC-8</td>
<td>122</td>
<td>99.97</td>
</tr>
<tr>
<td>TREC-10</td>
<td>76</td>
<td>99.53</td>
</tr>
<tr>
<td>TREC-12</td>
<td>71</td>
<td>100.0</td>
</tr>
</tbody>
</table>

occurred.

When the error rates are computed, the difference in EM values between
two systems (with respect to one of the two involved topic sets) is classified
into one of (in my case) 101 bins, where each bin represents a range of
differences in EM values: [0, 0.002), [0.002, 0.004), …, [0.198, 0.2), [0.2, 1].

Now, the error rate for a bin b, in relation to a topic set size z, is defined
as the share of the differences d in b such that d is associated with a swap.

When the error rates are computed, one can obtain information, for a given
topic set size, on the least difference in EM values for having 95% confidence
in the conclusion. If we for example get a swap for less than 5% of the system
pairs and disjoint topic sets when the initial EM difference is greater than
\( \delta \), we conclude that if the EM difference between two systems is at least
\( \delta \), then we can be 95% certain that the system with the higher EM value
actually is better. For more detailed information on the computing of error
rates, see Voorhees and Buckley (2002). To get stable values, the full set
of systems for each TREC has been used, which gives a large number of
possible system pairs. Further, 1000 random partitionings of each topic set
has been used.

Table 7.13: Experimentally extracted significant differences for each metric
in TREC-8 calculated for \( z = 25 \), i.e. 50 topics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>TREC-8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
</tr>
<tr>
<td>MAP</td>
<td>0.413</td>
</tr>
<tr>
<td>( bpref )-10</td>
<td>0.455</td>
</tr>
<tr>
<td>RankEff</td>
<td>0.745</td>
</tr>
</tbody>
</table>

Error rates for \( bpref \)-10, RankEff and MAP were calculated, using topic
set size 50 for TREC-12 and 25 for TREC-8 and TREC-10 (the maximal
sizes, given 100 and 50 topics respectively). The share of swaps for all pairs
of runs from TREC-8, TREC-10 and TREC-12 was counted, using the 1000
different randomly selected pairs of disjoint topic sets.
Table 7.14: Experimentally extracted significant differences for each metric in TREC-10 calculated for $z = 25$, i.e. 50 topics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Best</th>
<th>$\delta$</th>
<th>$%$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.283</td>
<td>0.050</td>
<td>17.7</td>
<td>0.0022</td>
</tr>
<tr>
<td>bpref-10</td>
<td>0.332</td>
<td>0.058</td>
<td>17.5</td>
<td>0.0023</td>
</tr>
<tr>
<td>RankEff</td>
<td>0.777</td>
<td>0.074</td>
<td>9.5</td>
<td>0.0063</td>
</tr>
</tbody>
</table>

Table 7.15: Experimentally extracted significant differences for each metric in TREC-12 calculated for $z = 50$, i.e. 100 topics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Best</th>
<th>$\delta$</th>
<th>$%$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>0.311</td>
<td>0.028</td>
<td>9.0</td>
<td>0.0028</td>
</tr>
<tr>
<td>bpref-10</td>
<td>0.326</td>
<td>0.032</td>
<td>9.8</td>
<td>0.0024</td>
</tr>
<tr>
<td>RankEff</td>
<td>0.763</td>
<td>0.030</td>
<td>3.9</td>
<td>0.0094</td>
</tr>
</tbody>
</table>

In Table 7.13, table 7.14 and table 7.15, the least absolute differences (the $\delta$ values) for having 95% confidence in the conclusion are given for each metric and each TREC. A column labeled ‘$\%$’ reports the share in percent of the best observed $MAP$ (average $bpref-10$, average $RankEff$) value obtained for a run and for a given TREC that $\delta$ represents. A $\sigma$ column gives the standard deviation over the observed $MAP$ (average $bpref-10$, average $RankEff$) values for a given TREC. For TREC-8 and TREC-10, the largest $\delta$ values are associated with $RankEff$. However, this metric also has the largest standard deviations, which indicates that one often obtains a larger absolute difference between two systems compared to $MAP$ and $bpref-10$. Relative to the best observed values, the differences associated with $RankEff$ are less than the differences for $MAP$ and $bpref-10$. In this relative sense, $RankEff$ outperforms the two other metrics.

7.6.3 Correlations between system-rankings with full pools of relevance data

The expression full pools of relevance data is a bit misleading since there are plenty of documents in the database that are not relevance-judged. Section 7.2.1 is a discussion of this topic. The pools are full in the sense that they consist of all relevance-judgements found in the TREC collections.

Let $S$ be the set of the $n$ runs executed in a given evaluation context, and let $EM$ be an evaluation metric. A system-ranking with respect to $S$ and $EM$ is a list $(r_1, \ldots, r_n)$ of the runs in $S$ such that $EM_{Avg}(r_i) \geq$
Document retrieval based evaluation

$EM_{Avg}(r_j)$ if $i < j$, where $EM_{Avg}(r_k)$ is the $EM$ average for run $r_k$ over the involved topics. (For $MAP$, $EM$ is uninterpolated average precision (AP), and $EM_{Avg}(r_k) = MAP(r_k)$.)

One way to empirically study if two evaluation metrics measure the same thing is to compute the correlation between two system-rankings, where one ranking is obtained from one of the metrics, the other from the other metrics. Table 7.16 gives, for each of the three TREC's, correlation data for $bpref$-$10$, $RankEff$ and $MAP$. Correlations are measured by Kendall’s $\tau$ (Kendall and Gibbons, 1990). The Kendall correlation values between $bpref$-$10$ and $RankEff$ are fairly weak over all TREC's. $bpref$-$10$ has higher correlations with $MAP$ than with $RankEff$. One of the correlation values between $bpref$-$10$ and $MAP$ is (slightly) less than, while the remaining two values are greater than 0.9, which has been used as a cut-off for equivalent rankings (Buckley and Voorhees, 2004). Figures 7.9–7.11, one for each TREC, give a picture of the level of agreement with regard to system-rankings for the three metrics. The $x$-axis in each figure represents systems, sorted by decreasing $MAP$ values. Both $bpref$-$10$ and $RankEff$ exhibit a decreasing trend. However, the low Kendall correlation between $RankEff$ and $MAP$ (Table 7.16) is clearly mirrored in the three graphs. $RankEff$ disagrees with $MAP$ (and with $bpref$-$10$) as to which system has the better performance for several pairs of systems. For a noticeable case, consider Figure 7.10 (TREC-10). It is difficult to see the exact numbers in the figure, but one system ranked at position 42 by $MAP$ is ranked as number 7 by $RankEff$. Similarly, we can see a system in Figure 7.11 (TREC-12) ranked number 17 by $MAP$ and 43 by $RankEff$.

We can conclude that for the three metrics, $MAP$ and $bpref$-$10$ have the closest correlation to each other and that $RankEff$ is closer to $MAP$ than to $bpref$-$10$. This may be a bit surprising since $RankEff$ and $bpref$-$10$ are more similar to each other based on their definition, which will be discussed more later in this chapter.

Table 7.16: Kendall correlations between system-rankings for each metric

<table>
<thead>
<tr>
<th>Metrics</th>
<th>TREC-8</th>
<th>TREC-10</th>
<th>TREC-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MAP$ vs. $bpref$-$10$</td>
<td>0.932</td>
<td>0.892</td>
<td>0.939</td>
</tr>
<tr>
<td>$MAP$ vs. $RankEff$</td>
<td>0.856</td>
<td>0.760</td>
<td>0.770</td>
</tr>
<tr>
<td>$bpref$-$10$ vs. $RankEff$</td>
<td>0.823</td>
<td>0.698</td>
<td>0.741</td>
</tr>
</tbody>
</table>

Using Kendall’s $\tau$ was proposed by Buckley and Voorhees (2004). I have also calculated the Spearman rank order correlation coefficient and it gives results in line with Kendall’s $\tau$. 

80
7.6 Experimental comparison using log-files from TREC

7.6.4 Effects of gradually reducing relevance data

For a given topic \( t \in T \) and a given TREC, start with the full pool, \( P_t \), of judged documents. Then gradually reduce \( P_t \) in the following way. Let \( P^{rel}_t \) be the set of known relevant documents for \( t \), and let \( P^{irr}_t \) be the set of known irrelevant documents for \( t \). Then the union of \( P^{rel}_t \) and \( P^{irr}_t \) is equal to \( P_t \). For each percentage value \( \alpha \in V_\alpha = \{1, 2, \ldots, 10, 15, 20, \ldots, 95\} \), the corresponding number of relevant (and irrelevant respectively) documents were randomly selected from \( P^{rel}_t \) (\( P^{irr}_t \)). For example, if 95% of the relevant (irrelevant) documents were to be selected and if the number of relevant documents was 40 and the number of irrelevant ones was 200, \( 0.95 \times 40 = 38 \) relevant and \( 0.95 \times 200 = 190 \) irrelevant documents were selected. If the number, say \( x \), of documents to be selected was a non-integer, 0.5 was added to \( x \), and then the greatest integer less than \( x + 0.5 \) was selected. This approach yields, for each \( \alpha \in V_\alpha \), \( \alpha \) as the expectation value over the
7.6.4.1 Change in absolute average values

The graph of Figure 7.14 shows for the three metrics and for TREC-12, the effects of gradually reducing the pools of judged documents on consistency of absolute average values. For each metric, a plotted value is the average over all topics and all runs, in relation to a certain level of incomplete relevance data. The graphs for TREC-8 and TREC-10 (figure 7.12 and figure 7.13) are similar. The reduction has a clear impact on MAP and, to a lesser extent, on \( b_{pref}-10 \). The former metric decreases as the level of incompleteness increases, with few exceptions. \( b_{pref}-10 \) increases as the
Figure 7.11: Average values for each metric and system in TREC-12, ordered by MAP values

level of incompleteness increases. However, \textit{bpref-10} changes at a slower rate compared to \textit{MAP}, with regard to $\alpha = 100\%$ down to $\alpha = 35\%$. \textit{RankEff} is the metric that exhibits the most consistent behaviour: the average value stays approximately the same until 4\% ($\alpha = 4\%$) of the relevance data are left. TREC-8 seems to be the most difficult collection for \textit{RankEff} since the value is close to the starting value only until $\alpha = 20\%$ where it rises a bit.

One problem with inconsistency in the present sense is that it might affect systems that did not contribute to the pools. If such a system tends to retrieve unique relevant documents, and the used evaluation metric tends to decrease (or increase) when the relevant documents are reduced, then the system is treated unfairly (or favoured).

\subsection{Correlations between system-rankings}

It is highly desirable that an evaluation metric is stable in the sense that it ranks systems (at least approximately) in the same relative order under
different levels of incompleteness with respect to relevance data. Assume that the pooling method has been used for a given test collection, and that several relevant documents are not in the pools. Assume further that the retrieval effectiveness of a set of systems have been measured with respect to an evaluation metric $EM$, and that a system-ranking is at hand. If $EM$ is unstable, we might ask what would happen if more relevance data were added. It is possible that a hypothetical new ranking would disagree with the old one, perhaps to a large extent. Disagreement of this kind would be a serious problem for the Cranfield evaluation model.

The graphs in figures 7.15–7.17, which correspond to TREC-8, TREC-10 and TREC-12, plot the Kendall correlations between system-rankings obtained from MAP ($bpref$-10, $RankEff$) using the full relevance pools, and system-rankings obtained from the same metric using reduced pools. The $x$-axis represents the level of relevance data incompleteness, while the $y$-axis shows the $\tau$ value. Optimal stability performance for a metric would
correspond to a straight line from the upper left corner to the upper right corner. For the test data used in this work, \textit{RankEff} is more stable than \textit{MAP} and \textit{bpref}-10. For each of the three graphs, the plot for \textit{RankEff} is flatter than the plots for the other two metrics. As the relevance data are gradually reduced, \textit{RankEff} goes on to rank systems in approximately the same relative order as when the relevance pools are not reduced. For TREC-8 and TREC-12, we have to move to incompleteness levels of 90\% (\(\alpha = 10\)) and 89\% (\(\alpha = 11\)) before the value of \(\tau\) drops below 0.9. TREC-10 deviates from the other two TREC's in this respect, since its corresponding incompleteness level is 75\% (\(\alpha = 25\)).
7.7 Conclusion

The two relatively new metrics of document retrieval effectiveness \textit{RankEff} and \textit{bpref}-10 have been studied. The metrics have been constructed as a response to the problem of incomplete relevance data, a problem that is likely to be inherent in large test collections (like the TREC\textsc{es}). Both metrics are such that they do not take into account documents that have not been relevance judged. The metrics were theoretically compared, and experimentally compared to each other and to the well-known evaluation metric MAP.

7.7.1 Results

The experimental results indicate that \textit{RankEff} may be a better alternative to MAP than \textit{bpref}-10 when the relevance data are incomplete. With re-
7.7 Conclusion

Figure 7.15: Kendall correlation values for system-rankings between full relevance pools and reduced pools for TREC-8

spect to consistency of absolute average evaluation values as the relevance data are reduced, RankEff has by far the best performance. bpref-10 performs better than MAP, which is problematic from this perspective. As indicated in section 7.6.4.1, inconsistency in this respect might affect systems that did not contribute to the pools.

System-ranking stability, the ability of a metric to rank systems in the (approximately) same relative order under different levels of relevance data incompleteness, is a very important variable. For reasons given in section 7.6.4.2, an evaluation metric that performs badly on this variable would constitute a serious problem for the Cranfield evaluation model. The three graphs of section 7.6.4.2 (figure 7.15-7.17) show that RankEff exhibits a better performance than bpref-10. MAP has the worst performance, for all three considered TRECks.

The relative error rates of RankEff are lower than the corresponding rates for bpref-10, which in turn has rates similar to the rates of MAP. As pointed
Figure 7.16: Kendall correlation values for system-rankings between full relevance pools and reduced pools for TREC-10

out in Buckley and Voorhees (2000), the error rate is only one property of an evaluation metric. A recall-oriented metric with a low error rate would of course be improper to use when the retrieval of a few relevant documents at top positions is of interest. However, a low (relative) error rate is clearly a desirable property of an evaluation metric.

MAP and $b_{pref-10}$ are in close agreement as to the ranking of systems when the full pools of judged documents are used for evaluation (Table 7.16), whereas the agreement between $RankEff$ and $b_{pref-10}$, and between $RankEff$ and MAP, is fairly small. This indicates that $RankEff$ measures something else than $b_{pref-10}$ and MAP. I consider $RankEff$ to be a metric of rank effectiveness. One of the differences between $RankEff$ and $b_{pref-10}$ concerns the case where a known relevant document swaps position with a known irrelevant one. A metric of rank effectiveness should increase as a consequence of such a change. It is not hard to prove that $RankEff$ increases under the change. For $b_{pref-10}$, this is not always the case. One
might argue that a metric that correlates poorly with an established metric, like MAP, is likely to be a poor metric (Buckley and Voorhees, 2004). However, if the established metric is problematic, it is questionable if a high degree of correlation is desirable. What has been shown in section 7.2.5 is that RankEff and $bpref$-10 are similar as long as the number of known relevant documents is not extremely small. In that case RankEff does a better job than $bpref$-10.

During the experiments, only a limited set of test data consisting of three test collections, has been used. Therefore, the results of the experimental part of this work should be interpreted with caution. Evidence for the relative merits of RankEff has been found, but the metric should be tested against $bpref$-10 and MAP (and perhaps against other evaluation metrics) using other test collections with incomplete relevance data.
7.7.2 Discussion

I have tried to describe what RankEff measures by an explanation of the equation it is defined by. The similar bpref-metrics have the same weakness since their functionalities are even less explained in the articles they are used in. One reason used by Buckley and Voorhees (2004) to argue that bpref is a good metric that measures something meaningful, is that it correlates strongly with MAP, but that is not enough since it is not clear what happens to the MAP-values when relevance-judgements are missing. Even if we accept MAP as a good metric, it does not help the RankEff-metric since they are not strongly correlated. More empirical work have to be done to get an intuitive picture of when RankEff gives good or bad scores.

If metrics like RankEff and bpref-10 are used for evaluation in environments like TREC, one might consider instructing the participating groups to rank the full pool of judged documents for a topic. In that way, there would be no need for the practical adjustments described in section 7.3, and RankEff and bpref-10 could be used as they are intended to be used, and most important, the full pools will be used to find a correct ranking of the competing systems.

For system-ranking it is relatively robust against the case when the amount of relevance-judgements is small or if the number of relevant documents is too low or too high. RankEff is therefore an important contribution of general utility for the field of information retrieval. At this point I can conclude that RankEff is the best metric to use in part III of this thesis.
Chapter 8

Interpretation of evaluation results

Two different evaluation tasks are used. The results of these evaluations should be interpreted carefully. First, I will decide how to combine the results from the cases when the task is clearly MWU-dependent and when it is not. I will also investigate what kind of significance testing that can be used.

8.1 Scale of improvement

I have chosen to look separately at results for cases where MWUs are clearly involved. In the synonym test this distinction is trivial to make since some of the queries and answers are MWUs, so I simply assume that these queries are MWU-dependent. For the document retrieval task it is not at all trivial to know in advance when MWUs will have an impact. I have tried to do this by looking at topics with very different scores between compared models, deciding which of them that contain central multi-word terms in the topic text. A more ambitious approach would be to define separate test collections containing MWU-dependent topics and documents.

A difference between two LSVMs should now be compared separately for sub-tasks that are MWU-dependent and those who are not. The results may of course be different for the two different evaluations, so a complete comparison of two models will contain values for each task-type.

8.2 Significance testing of document retrieval results

Van Rijsbergen (1979) noted that the data distributions for IR evaluations do not follow the requirements for sign, Wilcoxon or t-test, so none of these should be used without being very careful. However, these are used for significance testing of IR experiments, since that is what is available. To find out how well we can rely on the significance tests, Zobel (1998) used TREC-5 data (50 topics and runs from 61 systems) to see if differences, that were considered significant according to the tests, actually were stable between different sets of topics. Their conclusion was that the Wilcoxon test
is better than t-test or ANOVA tests. Experiments in Buckley and Voorhees (2000), similar to the ones performed in section 7.6.2 in this thesis, point out that 50 topics will give an error rate at 2-3% (absolute values) for MAP, but metrics based on fewer retrieved relevant documents from each system, like \( \text{PREC}_{10} \), get much less stable. \( \text{Recall}_{1000} \) gets a low error rate, but has on the other hand a very low standard deviation, which makes it less useful for system ranking.

My experiments show that \( \text{RankEff} \) and \( \text{WRS} \) are more stable than \( \text{bpref} \), which results in that a smaller absolute difference is needed for significance. This result in combination with the fact that \( \text{RankEff} \) has a higher standard deviation over systems means that \( \text{RankEff} \) is more useful in order to find significant differences between systems, but there is still work to be done here. The scores for each topic is not normally distributed, which probably make the statistical tests more demanding than needed. Instead I am going to consider the differences in each case and compare to the experimentally calculated difference levels needed for significance (section 7.6.2).

### 8.3 Significance testing of synonym test results

How much better result do we need for one LSVM \( M_1 \) to conclude that it is actually better than another one \( M_2 \), and not just gets a better result as a lucky coincidence?

We have two systems \( M_1 \) and \( M_2 \), and each of them solves a set of \( n \) synonym test queries. The results are illustrated in table 8.1. The null hypothesis \( H_0 \) is that both systems are equally good and that the difference is caused by random fluctuations. I will now look at two different tests of whether \( H_0 \) is true or not.

<table>
<thead>
<tr>
<th>Query</th>
<th>Answer</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>no</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>E</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 8.1: Answers from two hypothetical LSVMs \( M_1 \) and \( M_2 \) for \( n \) synonym queries
8.3 Significance testing of synonym test results

8.3.1 The $\chi^2$-test

One common test in the synonym test case is to calculate the $\chi^2$-value. Let us start with information from table 8.1 and count the number of correct and wrong answers for each system. In table 8.2 I have also calculated totals and expected values.

<table>
<thead>
<tr>
<th>LSVM</th>
<th>Right</th>
<th>Wrong</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>270</td>
<td>290</td>
<td>560</td>
</tr>
<tr>
<td>$M_2$</td>
<td>242</td>
<td>318</td>
<td>560</td>
</tr>
<tr>
<td>Total</td>
<td>512</td>
<td>608</td>
<td>1120</td>
</tr>
<tr>
<td>Expected</td>
<td>256</td>
<td>304</td>
<td>560</td>
</tr>
</tbody>
</table>

Table 8.2: Score and expected value for two hypothetical LSVMs $M_1$ and $M_2$, based on the 1120 observations

In this example we have 560 queries and the two LSVMs may be right or wrong for each of them. The null hypothesis is that the two LSVMs have the same performance, i.e. the difference is not significant. Since we have two LSVMs we should use one degree of freedom. Table 8.3 shows some critical values for the $\chi^2$-test. For example (see table 8.2), the scores for two hypothetical LSVMs: 242 (43.2%) and 270 (48.2%) gives the $\chi^2$-value 2.82 (equation 8.1 where $O$ stands for observed and $E$ for expected) which is just significant at the 0.1 level. We can see that rather big differences in percentages are needed for significance with the current size of the evaluation set. To get an improvement significant at the 0.01 level from 280 (50% correct answers) we will need a score of 323 (57.7% correct), which gives a $\chi^2$-value of 6.64 (just above the 0.01 level limit 6.63).

$$\chi^2_{\text{observed}} = \sum \frac{(E - O)^2}{E} = \frac{(270 - 256)^2 + (242 - 256)^2}{256} + \frac{(290 - 304)^2 + (318 - 304)^2}{304} \approx 2.82 \quad (8.1)$$

<table>
<thead>
<tr>
<th>Critical $\chi^2$-values</th>
<th>Significance level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degrees of freedom</td>
<td>10%</td>
</tr>
<tr>
<td>1</td>
<td>2.71</td>
</tr>
</tbody>
</table>

Table 8.3: $\chi^2$-values for some useful significance levels

8.3.2 McNemar’s test

One problem with the $\chi^2$-test in some cases is that the number of changes is relatively small, which leads to the conclusion of a non-significant difference, even if one of the systems is better than the other in all cases where
they differ. The McNemar test, presented in Guillick and Cox (1989), uses only the data where the systems differ. We need matched observations to be able to use the McNemar test, which we have. Let us define two probabilities: \( p_1 = P(\text{correct}(M_1)|E_1) \) and \( p_2 = P(\text{correct}(M_2)|E_1) \), where \( E_1 \) is the event that exactly one of the two systems gives the right answer and \( P(\text{correct}(M_1)) \) that the system \( M_1 \) is right. To calculate this we need the information in table 8.1, but only the cases where exactly one of the systems is right. The null hypothesis \( H_0 \) is when \( p_1 = p_2 = 0.5 \). The \( p_1 \) (or \( p_2 \)) value should be compared to an binomial distribution to find out if the difference is significant. To find out if either of the systems \( M_1 \) or \( M_2 \) is significantly better, we calculate a two-tailed \( p \)-value. If \( p \) is below the wanted significance level, we have a significant difference between the systems \( M_1 \) and \( M_2 \).

As an example we use the same data as for the \( \chi^2 \)-example in section 8.3.1 but in table 8.2 the information needed to calculate \( p_1 \) and \( p_2 \) is missing but can be found in table 8.4. For the examples in table 8.4 we get the two tailed \( p \)-value 0.0002, which indicates strong significance.

<table>
<thead>
<tr>
<th></th>
<th>( M_2 ) correct</th>
<th>( M_2 ) wrong</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 ) correct</td>
<td>230</td>
<td>40</td>
<td>270</td>
</tr>
<tr>
<td>( M_1 ) wrong</td>
<td>12</td>
<td>278</td>
<td>290</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>242</td>
<td>318</td>
<td>560</td>
</tr>
</tbody>
</table>

Table 8.4: Number of correct answers for two hypothetical LSVMs \( M_1 \) and \( M_2 \), based on the 560 queries

Note that the McNemar test is not affected by the number of cases where both the systems have the same result, i.e. both are correct or both are wrong.

### 8.3.3 Summary

For the example presented in the tables 8.2 and 8.4, the \( \chi^2 \)-test gives a significant difference between \( M_1 \) and \( M_2 \) at the 0.1 level while the McNemar test gives significance at the 0.0002 level! The prerequisites for both tests are fulfilled, but McNemar is much more useful for finding significant differences between systems. The reason is probably that it uses more information. \( \chi^2 \) does not use the pairwise answers for each query, which McNemar does, but just the number of correct answers for each system. When the McNemar test is used I include the \( \chi^2 \)-test as well since it has been used in similar evaluations.
Part III

N-grams in
Latent Semantic
Vector Models
Part III: N-grams in Latent Semantic Vector Models

Part III contains definitions of MWUs and the models used in the experiments, followed by chapters with experiments on synonym test evaluation and document retrieval evaluation.

Chapter 9
The first chapter of this part gives definitions of MWUs and $n$-grams. It continues with arguments on why I choose to use simple $n$-grams instead of syntactic or statistical phrases. I then define how the $n$-grams have been added to the LSVMs to form different models used in the evaluation experiments. Two experimental setups are used for evaluation. We find out how the performance is affected by the added $n$-grams, compared to a word-based model.

Chapter 10
This chapter presents the experiments for the synonym test evaluation performed with the SweHP560 data collection. The initial experiments explore how the used parameters should be set to get optimal performance for word-based models. The results from the following experiments can then conclude that for the synonym test task, we get an improvement by adding $n$-grams to the model.

Chapter 11
The document retrieval experiments use two separate evaluation sets; one Swedish and one English, are presented. The results show a noticeable change for the worse for the English data when $n$-grams are added. For Swedish, the change is smaller, and clearly not significant.
Chapter 9

Latent semantic vector models enriched with n-grams

This chapter starts with some information about various types of MWUs and how we can find them automatically. I continue by motivating which models I use and define all the LSVMs used in the experiments. I say something about the software and hardware used since there are important limitations imposed by them. The limitations have forced me to fine-tune the parameters to find a balance between different parameters that affect running time and memory needs.

9.1 Multi-word units

An assumption throughout this thesis is that multi-word units (MWUs) can improve information retrieval systems, see for example Zhai (1997) and Aljlayl et al. (2001). Word-based LSVMs are known to improve performance in IR applications and since multi-word terms often are chosen manually as index terms, I am interested in adding them to LSVMs. My hope is that MWUs can co-exist with single words in an LSVM without degrading the performance for single word queries. So let me start with an overview of earlier research followed by an attempt to find a definition of the term MWU that could be implemented to fit into an LSVM.

The major goal in this thesis is to improve LSVMs, in particular in an information retrieval context. Some typical examples of MWUs with a potential of doing that are:

- president bill clinton
- peanut butter
- à la carte
- spaghetti bolognese
- sort of
- latent semantic indexing
Some other sequences of words that should not be seen as MWUs are:

- latent semantic
- improve performance
- an attempt to find a definition

*latent semantic* is not an MWU because it is an incomplete unit. The next two are just normal word combinations. Various definitions of MWUs will follow, and none of these three examples of non-MWUs will match the definitions.

### 9.1.1 Linguistic multi-word units

Multi-word units (MWUs), or multi-word expressions, as they are sometimes called in linguistic research, are an important topic both for linguists and computational linguists. It is clear that the term is problematic since there are so many more or less formal ways to describe the MWUs. A definition of MWU given by Baldwin (2006) is the following:

**Definition 1** An MWU is

- decomposable into multiple simplex words,
- lexically, syntactically, semantically, pragmatically and/or statistically idiosyncratic.

Even if definition 1 is very clearly understandable to a linguist, it is difficult to use directly in order to find MWUs automatically. Sag et al. (2002) mention that MWUs may be found using symbolic or statistical methods, and they point out that the purely statistical methods are not enough. Their definition of MWU is, as they say very roughly, “idiosyncratic interpretations that cross word boundaries (or spaces)”.

For an NLP system aiming for wide coverage of general written language, the handling of MWUs is extremely important. Especially in specialized domains, a majority of all terms are MWUs.

As an initial attempt to find out a feasible definition, Baldwin (2006) points out that there are many kinds of MWUs, and that these can be described with a set of features: *Lexical, Syntactic, Semantic, Pragmatic* and *Statistical*. These features are not independent and also not always easy to determine. *Statistical* indicates whether the phrase is a phrase on statistical grounds, and *Lexical, Syntactic, Semantic* and *Pragmatic*, indicate on which linguistical levels the phrase is composed. The *statistical* feature can be implemented using probability theory (see definition 3 in section 9.1.2.2) while the other features are operationalized with nothing less than a grammar of the language of the corpus.
One term commonly used in linguistics is *collocation*, which means slightly different things for different linguists. A definition found in The Free Online Dictionary\(^1\) is presented in definition 2.

**Definition 2** Collocation:  
An arrangement or juxtaposition of words or other elements, especially those that commonly co-occur, as rancid butter, bosom buddy, or dead serious.

Definition 2 is not very useful in an implementation context since it is too vague. In section 9.1.2 I will try to give more precise definitions that are possible to implement.

**9.1.2 Operationalization of MWUs**

We now need to formalize the MWU-extraction to be able to use it in any kind of computer program. Basically, there are two ways to automatically find MWUs according to Mitra et al. (1997). We can use:

- **“Syntactic phrases”:** any set of words that satisfy certain syntactic relations or constitute specified syntactic structures make up a phrase. Thus, if we specify that an adjective followed by a noun constitutes a phrase, “economic impact” would be a phrase.”

- **“Statistical phrases”:** any pair (or triple, quadruple, etc) of non-function words that occur contiguously often enough in a corpus constitute a phrase. Thus, the words “United” and “States” may occur contiguously a large number of times in a corpus, and would constitute the phrase “United States”.

I want to add a complementing way of finding MWUs, which in some sense is a generalization of the two methods above, and also less theory dependent.

- **N-grams:** All occurring sequences of words up to the length \(n\).

Let us look at these three options one by one in the following sections.

**9.1.2.1 Syntactic phrases**

The language-dependent algorithms can use a lexicon and a grammar to get better results than the language-independent ones. On the other hand, a lexicon based method will have problems to find phrases of words that are not in the lexicon (or structures not in the grammar). A language-independent method is also easier to use on multilingual databases, since there is no need to have a working language detection and separate phrase extracting programs for each language. The major weakness for statistical methods without lexica or grammars is that they overgenerate, so depending on how much the application suffers from generation of unwanted

\(^1\)http://www.thefreedictionary.com/collocation
phrases, we may want to choose syntactic phrase parsing rather than statistical phrases.

An approach used by Merkel (1999) is to try to find syntactic phrases with “knowledge-lite” algorithms. The implemented systems start with a statistical approach based on raw frequency and entropy, to find candidate phrases, followed by adopting language filtering. One of the implemented systems has a high precision while the other one has higher recall and lower precision. Note that Merkel’s algorithms combine a limited lexicon with statistical features.

An alternative worth mentioning here is the MaltParser (Nivre et al., 2006), which is a data-driven parser-generator. Given a treebank, i.e. a parsed corpus, it produces a deterministic parser. The grammar is extracted from the parsed corpus, but we still need a parsed corpus from the right domain to get a working parser. With that resource available, we can produce a highly efficient inductive dependency parser (Nivre, 2005).

### 9.1.2.2 Statistical phrases

A corpus linguistic definition of statistical collocation, that is suitable for an automatic implementation is the one shown in definitions 3. The statistical collocations found for different lengths will form a set of statistical phrases.

**Definition 3** Statistical collocation:

*A statistical collocation is defined as a sequence of words which co-occur more often than would be expected by chance.*

This definition does not exactly catch the linguistic collocation so I will write “statistical collocation” instead of “collocation” for collocations found by definition 3. Statistical collocations according to 3 could be found in a corpus by calculating the mutual information (equation 9.1):

\[
I = \log \frac{P(w_1, w_2, ..., w_n)}{P(w_1) \cdot P(w_2) \cdot ... \cdot P(w_n)}
\]

(9.1)

The corpus size is normally much bigger than \( n \) in equation 9.1, which makes it possible to approximate the equation to get a formula with frequencies instead of probabilities:

\[
I = \log \frac{freq(w_1, w_2, ..., w_n)}{freq(w_1) \cdot freq(w_2) \cdot ... \cdot freq(w_n)}
\]

(9.2)

Equation 9.2 makes it easy to find all phrases that are collocations according to definition 3.

There are many more approaches to find statistical phrases (Manning and Schütze, 2001). Just to mention two examples:

- **Pearson’s Chi-square**
  
  Based on \( \chi^2 \) calculated from frequency values for the words \( w_1 \) and \( w_2 \) together, only one of them without the other, and when none of them are present.
• **Dunning’s log likelihood**
  Based on the binomial distribution and log likelihood ratio.

### 9.1.2.3 N-grams
Finding all \( n \)-grams in a text is an easy task both theoretically and computationally. We decide on a maximum length \( n \) and just collect all occurring words, word-pairs, word-triples, etc. from the text. The extraction of \( n \)-grams takes linear time in the size of the corpus and just a constant amount of RAM-memory. When new data is added, there is no need to process the old texts, just to process the new ones. Compared to statistical or syntactic phrases, the \( n \)-grams are much more simple. They cannot be used when we need high precision phrase, but if we can accept that not just collocations or phrases are included in the output, then \( n \)-grams may be an option. There will be more details and some examples in chapter 9.2.2.

### 9.1.3 Choosing which MWUs to add to the model
At this point I have argued that adding MWUs to an LSVM can be useful for many applications. We have also seen that there are many ways to find MWU-candidates in a text. The linguistic definition of MWU is not directly operationalized, but there are ways to automatically find MWUs in plain text, for example syntactic phrases found by a grammar or by a pattern-based methods or statistical phrases. I have instead chosen to use all occurring \( n \)-grams up to length 3 as MWU-candidates, which leads to a much higher number of units to add to the existing words in the texts. I will now discuss the reasons for making this choice between the three methods discussed in section 9.1.2. This will not be a complete set of arguments for and against each of the three options, but rather the main arguments that led me to the choice of \( n \)-grams in favour of syntactic phrases or statistical phrases during the experiments in this thesis.

1. **Completeness**
   A first argument in favour of using \( n \)-grams instead of more elaborated syntactic or statistical phrases is the simple fact that by using \( n \)-grams, no phrases are missing! Even if we have a well functioning algorithm of finding phrases using statistics or parsing, there will be a share of the existing phrases that are never found. Of course, compared to the number of useful phrases, the \( n \)-gram approach will find many times more \( n \)-grams that are not useful as phrases. On the other hand, these non-phrases will never be seen if we use the LSVM in a document retrieval task—people do not search for “finding phrases using”. A large share of these non-phrases have the global frequency of one and will therefore not affect any vector in the vector space except the vector for the non-phrase itself. Consequently, I think that the \( n \)-gram approach finds all MWUs and the non-phrases it finds will not do that much harm, except when the model is used in an application like automatic keyword extraction or thesaurus extraction.
2. **Efficiency**

The second argument to use plain n-grams is that it can be done in linear time to the size of the training data and with a constant (very low) need of RAM-memory. This is not the case for statistical phrases because we need a frequency list of n-grams up to a certain length to be able to calculate them and such a frequency list is expensive to calculate in terms of RAM memory need. Adding new data to the corpus forces us to recalculate the statistical phrases for the full corpus, including the the old data. Especially for a language like Swedish, where compounds are created in a creative way without putting spaces between the parts. Shallow parsing can be much more economical to perform but it still uses more time and space than n-grams.

3. **Language-independence**

Using n-grams is language-independent as long as there exist an easy way to make the tokenization. For a language like Chinese, this may be a problem. Since I have used a robust language-independent tokenizer, the task of finding all MWU-candidates using n-grams can be done with the same program for all languages that use some kind of alphabet and put spaces between words. Switching to a new language, or just a new domain, will force us to find a new parser if we use the parsing approach, and if we use a statistical approach we will have to do some tuning of the thresholds and parameters needed to extract statistical phrases.

4. **Simplicity**

Putting MWUs in an LSVMs is not very much explored. Therefore, I think that a good starting point is to keep things simple during the initial experiments. As we will see in section 9.4, there are many parameters to vary in such an experiment and to add one more parameter, the way of finding MWUs, will make it more complex to understand how to tune the parameters in order to find a functioning MWU-enriched LSVM. If we start with the n-grams, the next step will be to try more sophisticated ways to find the MWU-candidates, like statistical phrases or shallow parsing.

5. **Stop criteria**

If statistical collocations are used, we keep each sequence of words which co-occur more often than would be expected by chance. One problem with the approach is that this is actually the case for most occurring n-grams, even those with a frequency of one. The reason for this is the combinatorial explosion of the number of potential collocations from a specific vocabulary and the low frequency of most of the words in this vocabulary. Lin (1998) solves this problem by simply subtracting a constant \( c \) from the frequency count in equation 9.2, i.e. some kind of reversed additive smoothing. We also have to decide on a threshold \( t \) for the mutual information \( I \) and these constants \( t \)
and $c$ have to be tuned for every corpus and they will be different for different collocation lengths.

6. **Training artifacts**

One risk with the idea of extracting MWUs in a text and add them to the texts to make them searchable is that if we are not careful, some data will be used many times during training and some will not, i.e. some kind of partial overtraining that could distort the resulting LSVM. This will be the case if we add MWUs and also use the words they contain during the training. The alternative to remove the separate words are for some applications not acceptable. In IR for example, we need to be able to search for “Clinton” even if it is a part of the MWU “Bill Clinton”. One way to avoid the risk for partial overtraining is to use all $n$-grams. Now every word in the training data will be used the same number of times.

I hope that this discussion has answered most questions of why I am choosing a simple way of finding MWU-candidates, despite that it is clearly not as successful as statistical or syntactical phrases in the general case of phrase finding. However, phrase finding is not a goal in itself for me, but rather to improve latent semantic vector models. Even if we are going to use the LSVM in an application where high precision of real phrases is important, using $n$-grams followed by filtering of the resulting list could be an alternative.

Some of the arguments mentioned, especially **Efficiency**, **Language independence**, and **Simplicity**, have led me to the conclusion of not using more sophisticated tools like:

- A tokenizer that handles difficult cases like acronyms.
- A topic to query converter that chooses what to include and not before calculating the query vector.\(^2\)

These tools should be tested in future experiments.

9.2 **Models**

Let me now define the models I am actually using in the evaluation. The MWUs will for some of the models be added and this will happen already in the preprocessing step. After that, I have to define what parameters we have to set for the SVD-step.

\(^2\)The most obvious parts, like “The document should mention”, are removed from the query.
9.2.1 Latent semantic vector models

As I have said earlier, LSVMs are obtained by SVD. Lavelli et al. (2004) show that word×word-co-occurrences almost always outperforms the document×word-representation when the results are used for term categorization and term clustering. A word×word matrix tends to be less sparse and document lengths are less important, which is an advantage if the training data contains many types of documents with very different lengths. I have chosen to work with a word×word-matrix of co-occurrence data (see section 2.3.1) as an input to the SVD program. The word order between co-occurring word-pairs is not kept, so the initial matrix is always symmetric. In practice, the matrix is truncated in one dimension. This makes the processing much less memory and time consuming and it also seems to improve the performance. It is however important to note that in the dimension that keeps the vocabulary, we keep as many words as possible. With this kind of LSVM, a lookup of a document is performed as the sum of the vectors for each word in the document, since the documents are not in the model directly.

9.2.2 Adding n-grams to an LSVM

I am trying to handle n-grams in the same way during training and lookup. The processing below can be seen as a preprocessing performed on data in both cases before the resulting strings are sent to the LSI software for training or lookup. In the experiments I have used two different ways of adding n-grams to LSVMs:

1. Build one vector model with both words and tuples

2. Build separate vector models for words and each tuple length, calculate separate similarities in each model and weight them together

I have tried to add just bigrams or both bigrams and trigrams. These variations sum up to a set of models described in table 9.1 for each language where the last two are not used directly. Each model in table 9.1 is trained with the same program: The Infomap NLP System. The difference is the preprocessing that is exemplified in table 9.2. I start with the tokenized document in figure 9.1. Note that tokens that contain non-words are removed before sent to the Infomap NLP System and that all tokens are converted to lower case. The models $M_{T2sep}$ and $M_{T3sep}$ are not included in table 9.2 since they are just combinations of the other models.

```
Bill, Clinton, ',', the, former, president, of, the, United, States, of, America, ',', likes, computer, science, '.'
```

Figure 9.1: A tokenized version of the example document presented in section 2.1.1
Table 9.1: The different LSVMs used to find out how \( n \)-grams change the performance

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_W )</td>
<td>The LSVM trained on words with no ( n )-grams added</td>
</tr>
<tr>
<td>( M_{T2} )</td>
<td>Trained on words and bigrams. Both the words and the bigrams are trained into the same vector space. The pairs are treated as words in the model.</td>
</tr>
<tr>
<td>( M_{T3} )</td>
<td>Trained on words, bigrams, and trigrams into the same vector space.</td>
</tr>
<tr>
<td>( M_{T2sep} )</td>
<td>Trained on words and bigrams. The words and bigrams are trained into two separate models (( M_W ) and ( M_{T2only} )). A lookup is calculated as a weighted sum of two similarity values, one from each model. See section 9.2.3.3 for the used weighting scheme.</td>
</tr>
<tr>
<td>( M_{T3sep} )</td>
<td>Trained on words, bigrams, and trigrams. The words, bigrams, and trigrams are trained into three separate models (( M_W ), ( M_{T2only} ) and ( M_{T3only} )). A lookup is calculated as a sum of three similarity values, one from each model.</td>
</tr>
<tr>
<td>( M_{T2only} )</td>
<td>Trained on only bigrams. Single words are not present in this model so it has to be used in combination with ( M_W ) to be useful.</td>
</tr>
<tr>
<td>( M_{T3only} )</td>
<td>Trained on only trigrams. Single words or bigrams are not present in this model so it has to be used in combination with ( M_W ) and preferably also ( M_{T2only} ) to be useful.</td>
</tr>
</tbody>
</table>

### 9.2.3 Using the enriched LSVM

Defining an LSVM is not just defining the training process. The lookup strategy is also important and not at all trivial for models containing \( n \)-grams. In the evaluation tasks we need to be able to look up different object types in all the defined LSVMs:

- Words from the synonym test (both queries and alternatives)
- Phrases from the synonym test (both queries and alternatives)
- Topics from the document retrieval test suites
- Documents from the document retrieval test suites

I try to handle each object type in a similar way, so single words are seen as special cases of phrases, but they contain only one word. In fact, I look at both words and phrases as special cases of documents, so all three are
Table 9.2: The preprocessed version of the document in figure 9.1 for each model sent to the Infomap NLP System

<table>
<thead>
<tr>
<th>Name</th>
<th>Training text sent to the Infomap NLP System</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_W$</td>
<td>bill clinton former president united states america likes computer science</td>
</tr>
<tr>
<td>$M_{T2}$</td>
<td>bill bill_clinton clinton the_former former former_president president_of of_the the_united united states states_of_of_america america likes likes_computer computer_science science</td>
</tr>
<tr>
<td>$M_{T3}$</td>
<td>bill bill_clinton bill_clinton_the clinton the_former the_former_president former former_president_of president_of_of_the of_the_of_the_united the_united the_united_states united united_states united_states_of states_of_states_of_america america america likes likes_computer likes_computer_science computer computer_science science</td>
</tr>
<tr>
<td>$M_{T2only}$</td>
<td>bill_clinton the_former former_president president_of_of_the the_united united_states states_of_of_america likes_computer computer_science</td>
</tr>
<tr>
<td>$M_{T3only}$</td>
<td>bill_clinton_the the_former_president former former_president_of_of_the_of_the_united the_united_states united_states_of states_of_america america likes_computer_science</td>
</tr>
</tbody>
</table>

seen as the same kind of object even if they normally have very different lengths. A topic is a little more complicated since it has a structure. A topic consists of three parts: the title, a description, and a narrative (see section 11.2.2 for examples). The systems competing in TREC do many clever things to form query strings. I have chosen not to focus on this part of the retrieval, but just add the three parts together and treat this resulting string as a document during lookup. In the initial experiments I tried to use just the title and description, but even if the narrative sometimes contains information that one would expect to degrade the results, it turned out that including all three parts gave clearly better results. Now the four object types above have been turned into just one single object type: a document. The problem of normalization of vectors could be a problem when we compare vectors for very short and very long documents, but this is not the case for me since I am using the cosine as a similarity measure. A document containing many words will typically give a longer vector, but this will not affect the cosine of the angle between vectors.
9.2.3.1 Lookup in an LSVM enriched with n-grams

Looking up a document in the LSVMs is done in different ways depending on the model. The easiest case is the model \( M_W \), where I just look up each word and calculate the sum, see equation 9.3. \(|d|\) is the length of document \( d \) and \( \text{lookup}_{\text{low}}(w, M) \) is the low-level lookup of the word \( w \) in the LSVM \( M \). The result of such a lookup is an \( n \)-dimensional vector in the projected vectorspace of \( M \).

\[
\text{lookup}(d, M_W) = \sum_{i=1}^{\left|d\right|} \text{lookup}_{\text{low}}(w_i, M_W)
\]

(9.3)

A model containing both words and n-grams has to be handled a bit differently. Since the model was trained with both the single words and the n-grams, we need to do the lookup with both words and n-grams as well. One could do this in many different ways, but at this stage I have chosen to keep things simple. Just create the string containing both words and n-grams, look up each token and calculate the sum, see equation 9.4. \([w_{i-1}, w_i]\) is the token created by concatenating \( w_{i-1} \) and \( w_i \). The concatenation results in a bigram which can be looked up in \( M_{T2} \).

\[
\text{lookup}(d, M_{T2}) = \sum_{i=1}^{\left|d\right|} \text{lookup}_{\text{low}}(w_i, M_{T2}) + \sum_{i=2}^{\left|d\right|} \text{lookup}_{\text{low}}([w_{i-1}w_i], M_{T2})
\]

(9.4)

Note that if \( d \) is a tokenized text, equation 9.4 may create bigrams where one of the words is a punctuation mark. In that case, let us assume that \( \text{lookup}_{\text{low}}(w, M) \) returns a 0-length vector, i.e. no contribution to the sum. In fact we can generalize the formulas for any maximal \( n \)-gram length, which is done in equation 9.5.

\[
\text{lookup}(d, M_{T<n>}) = \sum_{l=1}^{\left|n\right|} \sum_{i=l}^{\left|d\right|} \text{lookup}_{\text{low}}([w_{i-l+1}, \ldots, w_i], M_{T<n>})
\]

(9.5)

The variable \( n \) (in the expression \( M_{T<n>} \)) is the maximum length of the \( n \)-grams. Note that \( M_W \) can be defined as \( M_{T1} \), so equation 9.5 also handles the word-based model.

9.2.3.2 Lookup in a model composed by LSVMs containing n-grams of different lengths

The difference between \( M_{T<n>} \) and \( M_{T<n>.sep} \) (\( \text{sep} \) stands for separated) is that \( M_{T<n>.sep} \) is built up by \( n \) different LSVMs containing \( n \)-grams of different lengths. Therefore, a lookup will return a list of \( n \) vectors, for example a lookup of a document \( d \) in \( M_{T3.sep} \) will return a triple of vectors.
where the first one is the vector for \( d \) in \( M_W \), the second is the vector for the new document \( d_2 \) containing all word pairs from \( d \) in \( M_{T2only} \) and the third vector is similarly the vector for \( d_3 \) in \( M_{T3only} \). The fact that we have three vectors for each document makes the similarity calculation a bit more complicated. I have chosen to use a linear combination of the, in this case, three vector-pairs. The calculation is described in section 9.2.3.3.

### 9.2.3.3 Calculating similarities in different kinds of LSVMs

In word models or \( n \)-gram models, similarity between two objects \( d_1 \) and \( d_2 \) in an LSVM \( M \) is defined as the cosine of the angle between the vectors (\( v_1 \) and \( v_2 \)) looked up (in the model \( M \)) for each object (equation 9.6), which is calculated as the dot-product between the normalized vectors.

\[
sim(d_1, d_2, M) = \cos(\text{lookup}(d_1, M), \text{lookup}(d_2, M))
\]

(9.6)

Where:

\[
\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|}
\]

The models where each \( n \)-gram length is separated is a bit different. The similarity is the sum of the similarities for each length. I have chosen to put a weight on each length to be able to give the \( n \)-grams more or less importance compared to the single words. The similarity for models of the type \( M_{T3sep} \) is defined in equation 9.7.

\[
sim(d_1, d_2, M_{T\text{l<sep}}} = \sum_{i=1}^{l} w_i \cdot \cos(\text{lookup}(d_1, M_{T\text{l<only}}), \text{lookup}(d_2, M_{T\text{l<only}}))
\]

(9.7)

Where the weights are defined as:

\[
w_i = \frac{i^c}{\sum_{j=1}^{l} j^c}
\]

In the equation, \( l \) is the maximum tuple length of the model and the constant \( c \) decides how big the difference in importance should be between the lengths. The special case \( c = 0 \) leads to the same weight for all lengths and if \( n \to -\infty \) the model converges to \( M_W \). As an example \( c = -2 \) for \( M_{T3sep} \) leads to the weights:

\[
w_1 = \frac{36}{49} \approx 0.73, w_2 = \frac{9}{49} \approx 0.18, w_3 = \frac{4}{49} \approx 0.082
\]

If we choose a value where \( c > 0 \), the tuple models will get a higher weight the longer they are.
9.3  Software for LSI calculation

In many places in this thesis I mention problems with lack of RAM memory and the long time of some computations. This section is a brief description of the software package I am using and a summary of various performance problems due to the large computations, I have included this section.

9.3.1  Introduction

I am using a revised version of the Infomap NLP System: An Open-Source Package for Natural Language Processing (Takayama et al., 1999), developed by the Computational Semantics Lab at Stanford University’s Center for the Study of Language and Information. It is an implementation of singular value decomposition (SVD) and interface software to calculate a word by word matrix from a corpus, running the SVD, and functions to calculate and compare vectors in the new projection. The interface is easy to use and also work very well for the kind of functionality I need. Some parts of the program have been rewritten to remove limitations due to internal memory and hard-disk usage. My version can handle up to 100 times bigger matrices than the original 0.8.6 version of the Infomap NLP System.

To be able to test the various combinations of corpora and parameters, I have written awk-scripts for:

- Building corpus files in the Infomap format
- Combine the different corpora and settings to build the vector models
- Run the synonym test for each model and store the result

9.3.2  Performance and complexity problems

Since SVD is a well established method, one would hope that existing software already handled complexity problems well enough for full scale use of SVD in LSI, but this is unfortunately not the case. Most researchers do experiments on limited sized data, and the commercial implementations are not available for academic researchers. However, with my revised version of the Infomap NLP System 0.8.6, most of the wanted experiments have been possible to perform.

9.3.2.1  Singular value decomposition

As I have mentioned in section 2.3.2, the SVD calculation is time consuming. The time complexity for the Lanczos SVD is $O(N^2 k^3)$ where $k$ is the dimensionality after reduction and $N$ is the summed number of rows and columns in the training matrix (Berry et al., 1995). $N$ will grow with the size of the data collection if we want to cover the full vocabulary. The growth is smaller than linear since many words are reused. For a bigger data collection it seems like a higher $k$ gives better results. In practical use the RAM-memory consumption is a bigger problem. The Infomap NLP
system needs to store all non-zero elements of the co-occurrence matrix in RAM-memory, and it also allocates a large amount of memory for the iterations during the calculation.

In the end of section 2.3.2 I discuss more efficient algorithms for SVD or other dimensional reduction methods, so in real life use of large tuple-enriched LSVMs, I do not worry too much about the performance problems in the future.

### 9.3.2.2 Tuples and words

When we include both words, bigrams, and trigrams, the number of tokens will increase with a factor of three. This is not a very big problem since it is just a linear factor. The number of repetitions of \( n \)-grams is not at all as high as for the single words, which will lead to a much higher number of types. On the other hand, we do not have to worry about types with only one occurrence since they will not affect the vector space. The high number of types is therefore not a problem during the SVD calculation, but it leads to more work during the preprocessing.

### 9.3.2.3 Implementation details

Depending on parameter settings and properties for the training data, different problems will arise, and depending on the implementation, the computation may take too long time or consume too much RAM-memory. “Too much” in this case means that each training of an LSVM takes more than 50 hours or that the need of RAM exceeds 10 gigabytes with the computers I have access to, but a more powerful computer is not a solution since there are always more data we could use. Generally, this task could be implemented to use less RAM if I had a faster computer, or the opposite.

The major changes from the original version of the Infomap NLP System are the following:

- The full co-occurrence matrix was stored in RAM to get the fastest processing. I have rewritten it using a hash-table instead, which saves RAM-memory but also slows down the processing a bit.
- To save hard disk space and to avoid bigger files than the system can handle, I changed some file formats from text to a more compact binary format.

### 9.4 Parameter settings

There are some choices to make before we can compare the LSVMs with or without \( n \)-grams. To get an idea of the best settings, I have looked at results for the SweHP560 synonym test collection. This is just to get a first idea of how the parameters should be set. These parameters are not a complete set of all possible LSVM parameters, but rather the set of parameters I use in this thesis.
9.4 Parameter settings

9.4.1 Upper and lower case
A not so complicated but still important option is whether we should keep upper and lower case letters or convert all alphabetic characters to lower case. The abbreviation Case is used in the result section with the possible values keep and lower.

9.4.2 Corpus choice
I strongly believe in the old mantra of statistical NLP “there’s no data like more data” since LSI relies to a high extent on redundancy. Another important thing to keep in mind is that my evaluation sets, especially the synonym test data, contains words very rarely found in newspaper text. On the other hand, this is just an evaluation and it is not a goal in itself for me to construct LSVMs that solves synonym tests. I will now describe the corpora used for training. The term types is used for distinct words and tokens for running words. Note that I am using a Swedish synonym test only, and therefore all training data mentioned in this chapter is in Swedish.

- Newspapers
  I have a collection of around 300 000 newspaper articles, in total 80 million tokens. From this collection, parts of various size are created. In this kind of collection we can observe the fact that the number of types can be very high in a Swedish corpus compared to an English one, where compounds are written with space between the parts. The number of word types in this corpus is more than one million types. The corpus named TT is a document collection of 140 000 short news stories containing 34 million tokens, which has been used as evaluation data by the Cross-Language Evaluation Forum (CLEF) in 2002, and GPHD is a corpus of 160 000 newspaper articles from two Swedish newspapers,\(^3\) in total 54 million tokens. The content of this newspaper collection is described more in detail in section 11.2.

- Bring thesaurus
  Bring is a Swedish thesaurus from 1930. It is inspired by an earlier version of Roget’s thesaurus (Roget, 2002) and just like Roget, it has around 1000 main categories containing a main word and groups of related nouns, verbs, and adjectives. I have not used these structures at all so each category is counted as one document. In total, Bring contains 60 000 types and 140 000 tokens.

- Lexin
  Lexin is a dictionary for language learners of Swedish. It contains 19 000 main words with short descriptions. Each main word and its description is counted as a document. In total, 200 000 tokens.

\(^3\)See section 11.2.1 and table 11.1 for more information on the TT and GPHD document collections.
• The Bible
  This is just a machine readable version of the Swedish Bible. It contains 800 000 tokens and 50 000 types, divided into 1200 documents – one for each chapter.

• Swedish Parole
  The Parole corpora is a result of a project financed by the European Union. This Swedish part contains newspaper text and novels, in total 20 million tokens and 600 000 types.

9.4.3 Combining corpora
I have used the corpora one by one for training but also combined them. One problem is that when using large newspaper corpora, the number of types becomes bigger than my software can handle, so if newspaper texts are combined with Lexin and Bring, which contain many of the important words for synonym tests that are missing in newspaper texts, many important words will be filtered out due to low frequency. More about this problem can be found in section 9.4.4.

9.4.4 Input matrix
My revised version of the Infomap software uses a hash-table instead of the complete matrix, but large vocabularies combined with a large context window still consume much more RAM-memory, so it is in many cases impossible to use the complete vocabulary on the second dimension of the input matrix. In the result section, I use Vocab for vocabulary size and CoVoc for the number of cells in the other dimension of the matrix.

9.4.5 Context size
The context size decides how close two words have to occur to be seen as a co-occurrence. If the context size is large, all words in the current document will be seen as context words to each other. I have chosen to look at the values: ±1000, ±300, ±100, ±30, ±10, and ±3 words. A context is limited to the same document. The abbreviation Con is used in the result section.

9.4.6 Number of dimensions after projection
This is the number of dimensions in the new vector space obtained by the SVD. The number of dimensions is much lower than the vectors in the

---

4In the hash-table we do not have to store all empty cells as in the complete matrix.
5To understand the CoVoc parameter, recall from section 2.3.1 that for term x term training we start with is a quadratic 2-dimensional matrix in the optimal case with the full vocabulary in both dimensions. A term is represented as a vector that reflects which terms it co-occurs with. When the vocabulary size is big, this matrix grows too big for a reasonable RAM-memory need and processing time. This is where CoVoc comes in. All terms in one dimension is kept to keep the vocabulary but in the other dimension, the most low-frequent terms are removed. The number of words kept is the CoVoc parameter, i.e. the number of co-occurring terms.
original co-occurrence matrix, but it is easy to calculate the vector in this space for any known word or combination of words, using the information from the SVD. It is reasonable to think that a bigger training corpus would need a higher dimensionality for all topics to fit into the vector space and earlier articles about LSI propose a dimensionality from 50 to 500. I have chosen to look at: 10, 30, 100, 300, 1000 and 3000. The abbreviation $\text{Dim}$ is used in the result section.

9.4.7 Combining the parameters

A new computer performance problem arises when combining all the different parameter settings. Apart from the different training corpora the different settings result in an explosion of combinations, and with some compositions of training corpora, the number of LSVMs to train easily reaches thousands. Each training is rather computation intensive since it contains calculations on a huge matrix and also requires hundreds of megabytes of disk space to store the model. I did not have the computer power to perform calculations of all combinations of parameter settings but exploring the parameter space works quite well even without all data points. By varying one or two parameters at a time at least a good local optimum can be found.
Chapter 10

The synonym test evaluation task

In this section I am going to use the SweHP560 ("Högskoleprovet" synonym test) presented in section 6.2 to evaluate the performance of LSVMs with and without \( n \)-grams in the model. First of all I will look at various parameter settings where the final one is the preprocessing that contains the adding of bigrams and trigrams, i.e. \( n \)-grams according to my definitions.

After this overall evaluation I will go into detail to find out for which type of examples I get differences between a word-based LSVM and an \( n \)-gram enriched one.

10.1 Experimental setup

I use the revised version of Infomap NLP System (see 9.3) to perform these experiments. With various parameter settings, including the preprocessing that enables \( n \)-grams, SVD-parameters and corpus compositions, I train a set of LSVMs. The corpora are preprocessed to contain the desired tokens and then sent to the Infomap NLP System, which builds and stores the LSVMs as files on the hard disk. Each LSVM is tested on the SweHP560 to see how well it handles the test overall and for each kind of query depending on whether the query or alternatives contain phrases or just single words. This results in overall scores and specialized scores for each LSVM.

10.2 Exploring one parameter at a time

Here I will go through the parameter space, dimension by dimension, but still keep the other dimensions in mind since they may interfere with each other. Note also that the baseline, which chooses randomly among the five answers, has an expected result of 20% (112 correct answers out of the 560). Note that the second result column in the tables shows the absolute number of correct answers.

I should mention a few things about the notation used both in the text and in the tables in chapter 10 and 11:

- I am using the notation \( k \) for 1000 (kilo) and \( M \) for million (Mega), for example 10k or 1M, to save space and to make the text and tables...
10.2 Exploring one parameter at a time

• In the vocabulary columns I mark the number of word types with * if the full vocabulary is used, for example 129k* if the training data contains 129 000 word types and all are included in the model.

10.2.1 Upper and lower case

I will start with the question if we should keep the case of the characters or convert all to lower case. The context size and the number of dimensions is here set to 100, CoVoc is 10 000 (10k) and the maximum Vocab is set to 1M, i.e. big enough to keep the full vocabulary. So let us take a look at the results for a few corpora in table 10.1.

Table 10.1: Results for the two case options

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>Case</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>keep</td>
<td>55.5%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>lower</td>
<td>55.7%</td>
</tr>
<tr>
<td>TT</td>
<td>617k*</td>
<td>keep</td>
<td>27.5%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>lower</td>
<td>28.6%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>657k*</td>
<td>keep</td>
<td>29.5%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>lower</td>
<td>30.9%</td>
</tr>
</tbody>
</table>

• The first value in the result-field is the percentage of correct answers and the second the absolute number of correct answers

• The * marks that Vocab is the complete vocabulary for this corpus

The subcorpora Bring and Lexin are almost completely lower case, which is also true for the SweHP560 collection. We can note that the number of word types (the Vocab column) is the same\(^1\) for keep and lower, but it changes a bit when the TT corpus is included where we also get a small improvement when converting to lower case. There is no significant difference between keep and lower, but the fact that the vocabulary gets smaller if we use lower case, together with some initial experiments in the document retrieval domain, made me decide to use lower case from now on. This choice has to be verified again for the document retrieval task.

10.2.2 Find the optimal corpus composition

There are many possible combinations of the available corpora. I have tried the ones in table 10.2 with Vocab set to fit all the word types. Case is set

\(^1\)There is a small difference: If we count types case sensitive we get 93 856 of them, and for lower case there are 93 851.
to lower, $Dim$ and $Con$ to 100, and $CoVoc$ to 10k. The table starts with models trained on each subcorpus alone.

The best single subcorpora are clearly Bring and Lexin and these two combined to Bring+Lexin give the best result. Bring alone gives lower results (significant at the 0.00001 level according to the McNemar test and at the 0.025 level with the $\chi^2$-test), followed by Lexin alone (significantly worse than Bring alone at 0.1 level with $\chi^2$-test and 0.07 with McNemar). The Bible is almost as bad as the baseline (random choice) and the newspaper corpora TT and GPHD are not very good either, but significantly better than the baseline (0.01 level with $\chi^2$-test). When adding the Bible or newspaper documents to the high performing Bring+Lexin, the result clearly gets significantly worse. Maybe we should reconsider the claim that more data is always better. On the other hand, it is not surprising that such a heterogeneous collection as the Bible combined with newspaper articles, a dictionary, and thesaurus results in a bad performing LSVM.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bible</td>
<td>29k*</td>
<td>21.8%</td>
</tr>
<tr>
<td>Bring</td>
<td>52k*</td>
<td>48.4%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>28.2%</td>
</tr>
<tr>
<td>GPHD</td>
<td>934k*</td>
<td>32.0%</td>
</tr>
<tr>
<td>Lexin</td>
<td>57k*</td>
<td>43.2%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>93k*</td>
<td>55.7%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>108k*</td>
<td>37.5%</td>
</tr>
<tr>
<td>Bring+Lexin+GPHD</td>
<td>630k*</td>
<td>30.7%</td>
</tr>
<tr>
<td>Bring+Lexin+TT+GPHD+Bible</td>
<td>1000k</td>
<td>30.2%</td>
</tr>
</tbody>
</table>

### 10.2.3 Verify that bigger vocabulary is better

A useful LSVM for the synonym test task should contain as many words as possible, but what happens with the overall results for different vocabulary sizes? Here the $Case$ option is set to lower, $Dim$ and $Con$ to 100, and $CoVoc$ to 10k. In table 10.3 I am now testing two corpus compositions in combination with some different values for $Vocab$. I will keep three corpus compositions since the results may vary in a different way for a large vocabulary corpus like TT (or Bring+Lexin+TT) than for the Bring+Lexin corpus, and my aim is of course to find the general behaviour rather than just for the optimal corpus for this evaluation task.

There is a significant difference between 94k and 30k vocabulary for Bring+Lexin and also between 10k and 30k. For TT and Bring+Lexin+TT we get significant on the 0.05 level with the $\chi^2$-test from 30k to the full vocabulary. The McNemar test gives even stronger significance. Therefore
I conclude that a bigger vocabulary is better. If possible, one should use the full vocabulary to get good coverage which shows in the results from the synonym test evaluation.

### 10.2.4 Find an optimal CoVoc value

In table 10.4 Case is set to lower, Dim and Con to 100. Now I will test some values for CoVoc to see how important the parameter is. Just as for the Vocab variable, I will keep three corpus compositions to get a more general conclusion.

For TT and Bring+Lexin+TT there are few significant differences, but we can at least see a trend that a higher CoVoc gives better and better results until CoVoc=3k, where the next value CoVoc=10k gives roughly the same result. The McNemar test gives a significant difference between 100 and 10k at the 0.05 level for these corpora. For Bring+Lexin it is not very clear either. However, the difference between CoVoc=10k and CoVoc=10k is clearly significant (McNemar). When we raise CoVoc above 10k, the results get slightly worse and the 100k-result is significantly worse than for 30k. A lower CoVoc makes the calculation less time consuming so there is no reason to go above 10k. If necessary we could lower CoVoc to 3k with almost no change. From 10k to 1k the result gets worse but the change is not significant so 1k is also a possibility if 3k takes too long time.

---

2The bigger CoVoc settings did not work with the TT and Bring+Lexin+TT corpora due to the high number of non-zero elements in the co-occurrence matrix this result in.
10.2.5 Find an optimal number of dimensions

I now want to explore what the best value for Dim might be. Here we cannot expect that a higher value is always better, since a higher value will get closer to a traditional vector model which will not be able to find similarities between words that do not co-occur. Instead I expect to find an optimal level where a lower or higher dimensionality performs worse.

Case is set to lower, Con to 100, and CoVoc to 10k. In Table 10.5 we have the results for some different values for Dim. The three corpora are kept even in this table to get a more general conclusion.

Bring+Lexin becomes a bit better with Dim=1000 than Dim=100 and significantly better for Dim=100 than for Dim=30 at the 0.005 level with the χ²-test. We also get the χ²-value 4.41 between Dim=1000 and Dim=3000, which is significant at the 0.05 level (McNemar gives p=0.0002). So for Bring+Lexin there seems to exist a level for Dim where we get a maximum, and the result gets worse for higher Dim. This is in line with the articles about LSI (Dumais, 1995; Landauer and Dumais, 1997), but the optimal dimensionality 1000 is a bit higher than reported earlier. However as I mentioned, the improvement from 100 to 1000 dimensions is not significant. Bring+Lexin+TT does not give a significant difference, except for the jump between 100 and 300 dimensions which gives a significant improvement at
the 0.05 level, but at least there is a tendency of the expected effect that there is a level of dimensionality where the performance is best and small differences around that maximum. This dimensionality is of course different for different corpora.

### 10.2.6 Find an optimal context size

The $Con$ variable is the most unclear of the variables investigated so far. The random indexing experiments by Sahlgren & Karlsgren are performed with a small context size for the best results while the old LSI experiments by Dumais (1995); Landauer and Dumais (1997) use the full documents as context size.

$Case$ is set to lower, $Dim$ to 100, and $CoVoc$ to 10k. In table 10.6 we have the results for some different values for $Con$. The three corpora are kept even in this table to get a more general conclusion.

Between $Con$=3 and $Con$=100, we get a significant difference at the 0.01 level for Bring+Lexin from the $\chi^2$-test, but for TT and Bring+Lexin+TT, there are no significant differences, not even tendencies. McNemar gives a significant difference between 100 and 300 for Bring+Lexin ($p=0.03$), not between 100 and 10, but between 100 and 3 we get $p<0.0001$. One should also note that the context may never stretch over document boundaries, which makes $Con$=300 and $Con$=1000 almost identical if the documents are not very long, as for Bring and Lexin that contain only short “documents”.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>Dim</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>10</td>
<td>36.4%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>30</td>
<td>45.9%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>100</td>
<td>55.7%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>300</td>
<td>56.6%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>1000</td>
<td>58.0%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>3000</td>
<td>51.8%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>10</td>
<td>26.1%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>30</td>
<td>28.4%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>100</td>
<td>28.2%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>300</td>
<td>28.6%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>10</td>
<td>27.0%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>30</td>
<td>30.2%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>100</td>
<td>30.7%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>300</td>
<td>36.4%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>1000</td>
<td>34.1%</td>
</tr>
</tbody>
</table>
Table 10.6: Results for different Con-values

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>Con</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>3</td>
<td>44.6%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>10</td>
<td>52.5%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>30</td>
<td>56.2%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>100</td>
<td>55.7%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>300</td>
<td>52.9%</td>
</tr>
<tr>
<td>Bring+Lexin</td>
<td>94k*</td>
<td>1000</td>
<td>52.9%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>3</td>
<td>25.9%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>10</td>
<td>23.9%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>30</td>
<td>25.0%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>100</td>
<td>28.2%</td>
</tr>
<tr>
<td>TT</td>
<td>590k*</td>
<td>1000</td>
<td>26.4%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>3</td>
<td>30.5%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>10</td>
<td>30.2%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>30</td>
<td>30.2%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>100</td>
<td>30.7%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>300</td>
<td>29.5%</td>
</tr>
<tr>
<td>Bring+Lexin+TT</td>
<td>630k*</td>
<td>1000</td>
<td>28.9%</td>
</tr>
</tbody>
</table>

10.2.7 Adding bigrams and trigrams

Now I have explored the parameter space, one parameter at a time, and we have a rather good knowledge about how to set the parameters to obtain a good model for the synonym test evaluation. We should keep in mind that the particular values may change for other training data, but I think that the overall behaviour of the results when parameters are varied, is similar. So now it is time to test one of the main questions for this thesis: What will happen to the accuracy in the synonym test evaluation when we add n-grams to the word-based model?

In table 10.7 I have tried to add first bigrams, and then also trigrams. Adding bigrams and trigrams is a way to obtain models containing both words and n-grams. The Words+bigrams+trigrams contains all n-grams up to a length of three tokens but, as I have discussed in section 9.2, I am adding all occurring bigrams and trigrams occurring in the training corpus, not just the ones that we would call “collocations”, “phrases” or n-grams. The corpus used is Bring+Lexin, Case is set to lower, Con to 100, and CoVoc to 10k. The Vocab value is set big enough to fit the full vocabulary. Since I suspect that the richer model obtained by adding the n-grams could need a higher dimensionality to reach the optimal score, I am running the evaluation for many different levels of dimensionality.

Recall from section 9.2.2 that MW is the Only words-model, MT2 is trained on Words+bigrams and MT3 on Words+bigrams+trigrams. To investigate whether the differences are significant or not I give p-values be-
Table 10.7: Results after preprocessing for the Bring+Lexin corpus

<table>
<thead>
<tr>
<th>LSVM</th>
<th>Vocab</th>
<th>Dim</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_W$</td>
<td>93k*</td>
<td>100</td>
<td>55.0%</td>
</tr>
<tr>
<td>$M_W$</td>
<td>93k*</td>
<td>300</td>
<td>56.6%</td>
</tr>
<tr>
<td>$M_W$</td>
<td>93k*</td>
<td>1000</td>
<td>58.2%</td>
</tr>
<tr>
<td>$M_W$</td>
<td>93k*</td>
<td>3000</td>
<td>51.8%</td>
</tr>
<tr>
<td>$M_T2$</td>
<td>323k*</td>
<td>100</td>
<td>56.2%</td>
</tr>
<tr>
<td>$M_T2$</td>
<td>323k*</td>
<td>300</td>
<td>59.1%</td>
</tr>
<tr>
<td>$M_T2$</td>
<td>323k*</td>
<td>1000</td>
<td>60.4%</td>
</tr>
<tr>
<td>$M_T3$</td>
<td>551k*</td>
<td>100</td>
<td>54.8%</td>
</tr>
<tr>
<td>$M_T3$</td>
<td>551k*</td>
<td>300</td>
<td>61.4%</td>
</tr>
<tr>
<td>$M_T3$</td>
<td>551k*</td>
<td>1000</td>
<td>61.2%</td>
</tr>
</tbody>
</table>

Table 10.8: $p$-values from the McNemar test between $M_W$, $M_T2$, and $M_T3$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$M_W$ (+,−)</th>
<th>$M_T2$ (+,−)</th>
<th>$M_T3$ (+,−)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_W$</td>
<td>-</td>
<td>0.08 (26,14)</td>
<td>0.02 (32,15)</td>
</tr>
<tr>
<td>$M_T2$</td>
<td>0.08 (14,26)</td>
<td>-</td>
<td>0.41 (14,9)</td>
</tr>
<tr>
<td>$M_T3$</td>
<td>0.02 (15,32)</td>
<td>0.41 (9,14)</td>
<td>-</td>
</tr>
</tbody>
</table>

...between the best $M_W$, $M_T2$, and $M_T3$ respectively in table 10.8. Unfortunately, I am not able to calculate $M_T2$ and $M_T3$ for Dim=3000. The best scores are received when Dim=1000 for $M_W$ and $M_T2$ so I will use Dim=1000 even for $M_T3$.

We can see in table 10.8 that the difference between $M_W$ and the models with n-grams added are significant while the difference is not significant between $M_T2$ and $M_T3$. I can now conclude that $M_T3$ is clearly better than $M_W$. For completion I should mention that the difference between $M_W$ and the best performing version of $M_T3$ (Dim=300) is also significant, but $p$ is only 0.045. The reason for this is that the total number of differing cases is bigger between these two models than between the $M_W$ and $M_T3$ used in table 10.8.

10.3 Comparing LSVMs in detail using SweHP560

Except for the difference in average percentage of correct answers, I want to look in detail for differences between two LSVMs when they are used to solve the SweHP560. The kind of data we are looking at are the same as

---

3 The (+,−)-numbers indicates the number of improvements and changes for the worse between the compared models.

4 $M_T3$ with Dim=300 is better than $M_W$ in 45 cases and worse in 27 cases, which result in $p=0.045$. 

---
Figure 10.1: Four answer combinations between $M_W$ and $M_{T3}$

- $=0$ Both models ($M_W$ and $M_{T3}$) are wrong
- $=1$ Both models are right
- $+$ Improvement: $M_{T3}$ is right and $M_W$ is wrong
- $-$ Change for the worse: $M_W$ is right and $M_{T3}$ is wrong

the McNemar test—pairwise differences for each query. I will take a look at two different LSVMs to see to what extent they give different answers. The two LSVMs use the same training data: the Swedish thesaurus Bring from 1930 with the same structure as the well known Roget’s Thesaurus, and a dictionary for language learners of Swedish called Lexin. In total, the training data contains some 340 000 tokens. They are trained with the same parameters: reduction to 1000 dimensions, context size of ±100 words and the same number of co-occurring words,\(^5\) but there is one difference:

- $M_W$ is trained the normal way using bags of words
- $M_{T3}$ is trained using bags of $n$-grams, where the $n$-grams are all 1-3 word sequences (unigrams, bigrams, and trigrams) in each document

The purpose with this comparison is to get a more fine-grained view of the effect of adding $n$-grams to an LSVM. I will look at $n$-grams and single words separately for queries and the answer alternatives, and also present differences between the model’s answers query by query.

### 10.3.1 Difference types between LSVMs

When comparing $M_W$ and $M_{T3}$ on SweHP560 we end up in four different cases (table 10.1).

<table>
<thead>
<tr>
<th></th>
<th>$M_W$ correct</th>
<th>$M_W$ wrong</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{T3}$ correct</td>
<td>311 (=1)</td>
<td>32 (+)</td>
<td>343</td>
</tr>
<tr>
<td>$M_{T3}$ wrong</td>
<td>15 (−)</td>
<td>202 (−0)</td>
<td>217</td>
</tr>
<tr>
<td>Total</td>
<td>326</td>
<td>234</td>
<td>560</td>
</tr>
</tbody>
</table>

Table 10.9: The share of each difference type between $M_W$ and $M_{T3}$

The $+$ and $−$ cells in table 10.9 show the differences between the models, in total $32 + 15 = 47$ queries, so let us take a look at the 47 examples behind these numbers. The results for the other $202 + 311 = 513$ queries are the same for $M_W$ and $M_{T3}$ so let us not worry about them. I will look

\(^5\)The co-occurrence matrix grows too big if we use all words in both dimensions. The results may also become better if we skip the most infrequent words in the second dimension. This is the so called CoVoc described in section 9.4.4.
at the 47 examples divided into groups depending on the type of queries and alternatives—single words or $n$-grams.

### 10.3.2 Single word queries and answers

Table 10.10 shows the cases where the models give a different result,\(^6\) and both the queries and answers are single words. The correct alternative is marked by italics. Note that for single words, we have 11 improvements and 6 changes for the worse (i.e. $M_W$ is correct and $M_{T3}$ is wrong). The result is a bit surprising. I would not expect such a big improvement as 65\% of the cases for queries where $n$-grams are not directly involved, in a model where only $n$-grams have been added.

### 10.3.3 Single word queries and MWU answers

In table 10.11 we have the cases where the query is a single word and at least one of the selected answer alternatives contains MWUs. We can see an improvement in 13 cases out of 19, i.e. 68\%. The share of improvement for $M_{T3}$ compared to $M_W$ is similar to the improvement for the queries where no MWUs are involved.

### 10.3.4 MWU queries and answers

For the MWU queries I would expect the highest level of improvement. Unfortunately, there are very few examples of this in the data collection. Let me start with the cases when all the alternatives are single words but the query contains MWUs. As we can see in table 10.13 we get one improvement and one change for the worse where the models differ. In table 10.15 there are nine examples of MWU queries with MWU answers. $M_{T3}$ gave the correct answer in 78\% of the cases and $M_W$ only 22\%. Table 10.14 shows the results for MWU-queries with at least one alternative that contains MWUs, and now we get an improvement in six out of eight cases, but as we can see in table 10.15, the two models give the same answer in 48 cases so they agree most of the time for this kind of examples.

### 10.4 Discussion

This section will summarize both the overall results and the more detailed discussion on specific examples of differences between a word-based and an $n$-gram enriched LSVM, looking at the different kinds of queries in the SweHP560 evaluation set. Note that I have not given priority to test the models with separate vector spaces for words, bigrams and trigram. The combined model is more interesting so I chose not to run these tests.

\(^6\)In the case when they give different answers but both are wrong, the result for the models are considered as the same, i.e. wrong answer.
Table 10.10: The cases of single word queries and answers where the answers from $M_W$ and $M_{T3}$ differ

<table>
<thead>
<tr>
<th>Query</th>
<th>$M_W$-answer</th>
<th>$M_{T3}$-answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>periferi/periphery</td>
<td>gränslinje/boarder</td>
<td>yttrområde/outskirts</td>
</tr>
<tr>
<td>ograverad/unspoiled</td>
<td>orörd/untouched</td>
<td>ojämnn/uneven</td>
</tr>
<tr>
<td>aviseras/announce</td>
<td>rådgöra/consult</td>
<td>underrätta/inform</td>
</tr>
<tr>
<td>storvulen/grand</td>
<td>fumlig/fumbling</td>
<td>mäktig/powerful</td>
</tr>
<tr>
<td>fromleri/hypocrisy</td>
<td>avund/envy</td>
<td>skenhelighet/hypocrisy</td>
</tr>
<tr>
<td>baxna/be taken (dumbfounded)</td>
<td>häpnna/be amazed (astonished, astounded)</td>
<td>kikna/choke</td>
</tr>
<tr>
<td>jämka/adjust</td>
<td>avlägsna/distant</td>
<td>anpassa/adapt</td>
</tr>
<tr>
<td>allmoge/country people</td>
<td>bondebefolkning/peasant society</td>
<td>adel/nobility</td>
</tr>
<tr>
<td>avans/gain</td>
<td>gäva/gift</td>
<td>vinst/profit</td>
</tr>
<tr>
<td>raljera/banter</td>
<td>skämta/joke</td>
<td>skratta/laugh</td>
</tr>
<tr>
<td>prosaisk/prosaic</td>
<td>uppsluppen/merry</td>
<td>vardaglig/everyday</td>
</tr>
<tr>
<td>aningslöst/unsuspecting</td>
<td>naiiv/naiive</td>
<td>spontan/spontaneous</td>
</tr>
<tr>
<td>beting/piece-work</td>
<td>försök/attempt</td>
<td>arbetssuppgift/job</td>
</tr>
<tr>
<td>mellanhavande/unsettled matters</td>
<td>olyckshändelse/accident</td>
<td>tvist/dispute</td>
</tr>
<tr>
<td>göl/water surface in moss areas</td>
<td>vattensamling/plash</td>
<td>skogsdunge/copse</td>
</tr>
<tr>
<td>vattendelare/watershed</td>
<td>flöde/torrent</td>
<td>skiljelinje/division</td>
</tr>
<tr>
<td>kausalitet/causality</td>
<td>betydelse/meaning</td>
<td>orsakssammanhang/causal relationship</td>
</tr>
<tr>
<td>Correct</td>
<td>6 (35%)</td>
<td>11 (65%)</td>
</tr>
</tbody>
</table>

10.4.1 Overall results

We have seen that with the right combination of parameters, which is a model with bigrams and trigrams added, an ordinary LSVM trained using just single words seems to be outperformed. The best result, 61.4% is close to the average student, taking the word synonym test, which is quite fun, even if it is not very meaningful, seen as an evaluation, to compare these figures with the ones for humans. For the 465 cases where the query and at least one alternative contains known words the result is as high as 69.3%. It looks like the optimal dimensionality is higher for the models containing n-grams, which is not surprising since they have a much higher number of word types.

In total, all the 560 queries and answer terms contain 3387 types (5798
Table 10.11: The cases of single word queries and MWU answers where the answers from $M_W$ and $M_{T3}$ differ

<table>
<thead>
<tr>
<th>Query</th>
<th>$M_W$-answer</th>
<th>$M_{T3}$-answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>förvisso/</td>
<td>utan tvivel/without doubt</td>
<td>efter bästa förstånd/</td>
</tr>
<tr>
<td>certainly</td>
<td></td>
<td>with good sense</td>
</tr>
<tr>
<td>hagla/hail</td>
<td>dimpa ner/drop</td>
<td>komma i stor mängd</td>
</tr>
<tr>
<td>ringakta/</td>
<td>skämmas för/be ashamed of</td>
<td>se ned på/look down on</td>
</tr>
<tr>
<td>despise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vurm/</td>
<td>stor långtan/big yearning</td>
<td>starkt intresse/great interest</td>
</tr>
<tr>
<td>passion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>debitera/</td>
<td>föra upp som skuld/take</td>
<td>avsluta affär/close deal</td>
</tr>
<tr>
<td>debit</td>
<td>as debt</td>
<td></td>
</tr>
<tr>
<td>distingerad/</td>
<td>stilfullt förnäm/distinguished</td>
<td>artigt uppmärksam/politely</td>
</tr>
<tr>
<td>dignified</td>
<td></td>
<td>attentive</td>
</tr>
<tr>
<td>traktera/</td>
<td>behandla någon illa/to</td>
<td>hjuda på mat och dryck/offer</td>
</tr>
<tr>
<td>to treat</td>
<td>treat someone badly</td>
<td>food and beverage</td>
</tr>
<tr>
<td>intuition/</td>
<td>känsla av obehag/feeling</td>
<td>förmåga till spontan bedömming/an ability to spontaneous judgement</td>
</tr>
<tr>
<td>intuition</td>
<td>of discomfort</td>
<td></td>
</tr>
<tr>
<td>dommera/</td>
<td>avbryta tvårt och otrevligt/</td>
<td></td>
</tr>
<tr>
<td>shout and</td>
<td>interrupt abrupt and in an</td>
<td></td>
</tr>
<tr>
<td>swear</td>
<td>disagreeable manner</td>
<td>tala aggressiv och högljutt/speak aggresively and loudly</td>
</tr>
<tr>
<td>galghumor/</td>
<td>elak och nedlåtande humor/mean and condescending humor</td>
<td>bister och ironisk humor/grim and ironic humor</td>
</tr>
<tr>
<td>gallows hu-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mour</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

tokens) from which 13% (448 types) are unknown to the best model trained with Bring and Lexin, and the most difficult ones are of course when the query term is unknown. When that happens all models take a random guess. The models trained on Bring, Lexin and TT know more words that result in only 5% unknown word types but they are much worse anyway as we have seen.

I earlier mentioned in section 9.4.2 the old mantra: “there’s no data like more data”, but the results in table 10.2 (section 10.2.2) suggests that this is not true. Instead it seems that adding the wrong kind of data can destroy a well functioning model since the evaluation accuracy falls from 55.7% down to 30.7%. A possible reason for this in the case of adding the TT subcorpus to Bring+Lexin is that the vocabulary from the much larger TT subcorpus replaces the important words from the Bring+Lexin part. In my case this cannot be the problem since the model keeps the full vocabulary. A more plausible explanation is that the added data in such large quantity takes over most of the dimensions resulting in a richer model with much
Table 10.12: Table 10.11, continued

<table>
<thead>
<tr>
<th>Query</th>
<th>$M_W$-answer</th>
<th>$M_{T3}$-answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>falang/wing</td>
<td>riktning inom politiskt parti eller annan organisation/direction within a political party or another organisation</td>
<td>grupp av ämnen vid universitet eller högskola/group of subjects at university</td>
</tr>
<tr>
<td>nyansera/make subtle distinctions</td>
<td>överbrygga åsiktsskillnader/bridge differences of opinion</td>
<td>åstadkomma finare skiftningar/bring about subtle variations</td>
</tr>
<tr>
<td>brådmogen/premature</td>
<td>illa förberedd/badly prepared</td>
<td>tidigt utvecklad/early developed</td>
</tr>
<tr>
<td>grannlaga/tactfull</td>
<td>som fungerar bra/that works well</td>
<td>som kräver finkänslighet/demands discreetness</td>
</tr>
<tr>
<td>anfang/initial letter</td>
<td>långdragen berättelse/lengthy story</td>
<td>beginnelsebokstav/initial letter</td>
</tr>
<tr>
<td>panorama/panorama</td>
<td>liten tavla/small picture</td>
<td>vidsträckt utsikt/extended view</td>
</tr>
<tr>
<td>pärs/ordeal</td>
<td>tragedi/tragedy</td>
<td>prövning/trial</td>
</tr>
<tr>
<td>sober/sober</td>
<td>smakfull/tasteful</td>
<td>beslöjad/covered</td>
</tr>
<tr>
<td>moralkaka/moralizing</td>
<td>måstrande anmärkning/fault finding remark</td>
<td>dold motsägelse/hidden contradiction</td>
</tr>
<tr>
<td>lecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>6 (32%)</td>
<td>13 (68%)</td>
</tr>
</tbody>
</table>
10.4 Discussion

Table 10.13: The cases of MWU and single word queries answers where the answers from $M_W$ and $M_{T3}$ differ

<table>
<thead>
<tr>
<th>Query</th>
<th>$M_W$-answer</th>
<th>$M_{T3}$-answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>black om foten/impediment</td>
<td>stöd/support</td>
<td>hinder/obstacle</td>
</tr>
<tr>
<td>låta påskina/insinuate</td>
<td>antyda/hint</td>
<td>liuga/lie</td>
</tr>
<tr>
<td>Correct</td>
<td>1 (50%)</td>
<td>1 (50%)</td>
</tr>
</tbody>
</table>

less quality for the synonym test task. A higher dimensionality may enable the important dimensions to be kept in the model but as we can see in table 10.5, a model with 1000 dimensions still gives a bad result when it is trained with Bring+Lexin+TT.

For the other parameters we get the expected results in most cases:

- To keep the case or to convert the corpus to lower case before training is not very important for the synonym test task, probably because the test data in itself is only lower case. To convert to lower case limits the number of word types and since the results are slightly better than if we keep the case, it is better to use only lower case. This has to be tested in the document retrieval task as well.

- Bigger Vocab is better, preferably the full vocabulary. This is intuitively right since unknown words are not handled at all by the LSVM.

- A bigger CoVoc gives a better result up to a certain level. It is a bit surprising that the full vocabulary as CoVoc is not optimal, but a possible explanation is that the least frequent words act more like noise that actually makes the result worse. From a computational point of view it is very good that a truncation of the matrix does not worsen the results but rather improves them.

- The expected optimum for Dim is, based on the literature, something like 300 dimensions. The performance should rise up to that maximum and then fall slightly when we have passed the optimum. This is what we also have found, but more evaluation data may give a clearer result. The actual value is different for different corpora.

- The optimal value for Con is difficult to guess since earlier research have been using context sizes from ±2 words up to the full document. The synonym test evaluation shows an improvement of the results up to a context size of ±30 or ±100, and then an unclear worsening, but since the best working training data for this evaluation contains only short documents, we will have to test different context sizes even for the document retrieval task.

---

7See chapter 3 for a discussion on the optimal number of dimensions.
Table 10.14: The cases of MWU queries and MWU answers where the answers from $M_W$ and $M_{T3}$ differ

<table>
<thead>
<tr>
<th>Query</th>
<th>$M_W$</th>
<th>$M_{T3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta skruv/do the trick</td>
<td>få nog/get enough</td>
<td>ha verkan/have effect</td>
</tr>
<tr>
<td>utgjuta sig/vent one’s feelings</td>
<td>känna djup ång/feel deep remorse</td>
<td>tala vitt och brett</td>
</tr>
<tr>
<td>inte skräda orden/not mince matters</td>
<td>vara faordig/being silent</td>
<td>säga sin upprätta mening/give one’s honest opinion</td>
</tr>
<tr>
<td>vurma för/have a passion for</td>
<td>ha ett starkt interesse</td>
<td>vara överdrivet rädd för/being to scared of</td>
</tr>
<tr>
<td></td>
<td>för/take a great interest in</td>
<td></td>
</tr>
<tr>
<td>vinlägga sig/take pains to</td>
<td>skynda sig/hurry</td>
<td>anstränga sig/make an effort</td>
</tr>
<tr>
<td>i parti och minut/wholesale and retail</td>
<td>med starkt stöd/with strong support</td>
<td>i stor mängd/in large quantities</td>
</tr>
<tr>
<td>sitta emellan/be the sufferer</td>
<td>tränga sig på/thrust oneself</td>
<td>bli lidande/be the sufferer</td>
</tr>
<tr>
<td>ligga i sin linda/be in its infancy</td>
<td>vara i början</td>
<td>vara uppenbar/be obvious</td>
</tr>
<tr>
<td>prängla ut/sell with unfair methods</td>
<td>försöka överlista/try to outwit</td>
<td>försöka sälja/try to sell</td>
</tr>
<tr>
<td>Correct</td>
<td>2 (22%)</td>
<td>7 (78%)</td>
</tr>
</tbody>
</table>

I have now explored all parameters including the possibility of adding $n$-grams to the training phase of an LSVM. We have a good picture of how to tune the parameters before going to the document retrieval evaluation task. For the synonym test evaluation task we can conclude that the performance gets better when $n$-grams are added to the model just by inserting all occurring bigrams and trigrams into the documents to be interpreted as single words. The improvement from the word-based LSVM to $M_{T3}$ is significant at the 0.05 level according to the McNemar test.

10.4.2 Summary of the detailed comparison of a word-based and an $n$-gram enriched LSVM

Table 10.15 gives an overview of the differences between $M_W$ and $M_{T3}$. We can clearly see that $M_{T3}$ handles MWUs better than $M_W$. The number of cases of improvement or change for the worse for each row is too low to be significant, but as we saw in table 10.8 the improvement is significant if we use all changes together.

Even between $M_{T2}$ and $M_{T3}$ we can see a strong tendency that $M_{T3}$ is better, but the number of differences between the models is too small to
give significant results. There is an improvement in 14 out of 23 cases which corresponds to 61% of the cases where the models differ, but we also have 537 cases where the models agree.

Table 10.16: Number of improvements for each model depending on queries/answer type, comparison of $M_T^2$ and $M_T^3$

<table>
<thead>
<tr>
<th>Query</th>
<th>Answer</th>
<th>Changed</th>
<th>No change</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$M_T^2$</td>
<td>$M_T^3$</td>
<td>Right</td>
</tr>
<tr>
<td>Single word</td>
<td>Single word</td>
<td>3</td>
<td>7</td>
<td>168</td>
</tr>
<tr>
<td>Single word</td>
<td>MWU</td>
<td>4</td>
<td>4</td>
<td>127</td>
</tr>
<tr>
<td>MWU</td>
<td>Single word</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MWU</td>
<td>MWU</td>
<td>1</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>9</td>
<td>14</td>
<td>329</td>
</tr>
</tbody>
</table>

We can now conclude that $M_T^3$ gives a significant improvement compared to $M_W$. Using the scale of improvement where we count improvement separated depending on whether MWUs are involved in the task, gives positive scores in both cases.
Chapter 11

Document retrieval evaluation

The experiments in this chapter are performed in the same way for both English and Swedish, except for some of the parameter settings. I will start to define the experimental setup and then present the two test suites. The two sections following the definition section describe the two series of experiments: document retrieval for English and Swedish. The Swedish synonym test showed an improvement when the $n$-grams were added. I will now perform the document retrieval evaluation for the different models to find out if we get an improvement here as well.

11.1 Definition of the document retrieval task

This section gives a definition of the evaluation task based on document retrieval. It also describes how the LSVMs are used to perform the task used in an evaluation.

11.1.1 Experimental setup

Just as for the synonym test evaluation in chapter 10 I use the revised version of the Infomap NLP System described in section 9.3. Making sure that the parameters (see section 9.4) are set in the right way, I try to find out how well an LSVM enriched with $n$-grams performs compared to a word-based LSVM. We have some experience of how to set the parameters from the synonym test, but some settings may have different optimal values for document retrieval than for synonym tests. The most obvious one is the corpus choice, but we have to be careful with the other parameters as well.

The evaluation test suites consists of the following parts:

- A database of documents $D$, where each document is, for example, a newspaper article.

- A set of topics $T$. Each topic consists of a title, a narrative, and a description.

- For each topic $t$ there is a pool $P_t = \{d_1,...,d_p\}$ of relevance-judged documents such that $P_t \subseteq D$. 

130
The Swedish test suite is developed with the English TREC data in mind, so the structure and amount of data is similar, except for the smaller set of relevance-judgements, as we will see.

To get a better comparison of the different IR-models in the evaluation that does not favour any of the IR-models, I have chosen to define the task as:

- \( \forall t \in T : \) Calculate a complete ranking list of all documents in \( P_t \) with aspect to relevance to \( t \).

These ranking lists will then be evaluated based on the gold standard (the relevance-judgements). Note that in the TREC environment a top-1000 list from each system is considered for evaluation—not the full pools.

### 11.1.2 The used evaluation metrics

I have studied existing evaluation metrics in section 5.3.1 and section 5.3.3, and also tested them in 7.4.1. In the document retrieval experiments I am going to use a subset of the metrics defined in section 7.4.1.

- **MAP**, equation 5.5, since it is the most used in document retrieval and has also been proved to work well.

- **bpref-10**, equation 5.7, since it is a metric that handles incomplete relevance data and correlates strongly with **MAP**. **bpref-10** is an adjusted version of the **bpref** metric, and both are introduced in Buckley and Voorhees (2004).

- **RankEff**, equation 7.1, which also handles incomplete relevance data, but seems to do a better job than the **bpref**-metrics (see section 7.6).

These metrics form a good mix of established and new, more advanced metrics. I have chosen to omit **bpref** and use only the improved **bpref-10**. **WRS** is also not used for two reasons: it is not yet tested very much and its definition is very close to **RankEff**.

### 11.1.3 Models used in the evaluations

These LSVMs were defined in detail in chapter 9, so I will just mention them here. I am going to evaluate two ways to include \( n \)-grams in LSVMs and compare them to a word-based LSVM. A simple keyword search model will be used as baseline. Here are all models used in the document retrieval task evaluation.

- **M\_Key**: This model is just keyword search, which is only relevant for the IR test suite where it is used as a baseline model. For the synonym test evaluation the baseline is a random choice of one alternative.

- **M\_W**: A model trained on documents treated as bags of words using LSI. Training data is different depending on evaluation task.
• $M_{T2}$: Almost the same as $M_W$, but I have added all word pairs in each bag of words, where the pairs are also conceptually treated as words in the model.

• $M_{T3}$: Similar to $M_{T2}$, but now we expand up to triples of words. Single words, word pairs, and word triples will all be treated as words in this model.

• $M_{T2sep}$: Trained on words and bigrams. The words and bigrams are trained into two separate models for words and bigrams. A lookup is calculated as a sum of two similarity values, one from each model.

• $M_{T3sep}$: Similar to $M_{T2sep}$ but now we are using three separate models for words, bigrams and trigrams.

11.2 The test suite of Swedish newspaper articles

Now it is time to use the defined framework for document retrieval evaluation on a data collection. I will first give some information about the Swedish evaluation set and then look at the results.

11.2.1 The document database

The database of documents is actually two collections, collapsed to one:

• **TT**: A collection of short news articles from “Tidningarnas Telegrambyrå”¹ (TT) published in 1994-95 and used in CLEF 2002.

• **GPHD**: A collection of news articles published 1994-95 in the two newspapers Göteborgs-Posten and Helsingborgs Dagblad and used in CLEF 2000.²

Table 11.1 gives a quantitative summary of the document collections used.

<table>
<thead>
<tr>
<th>Collection</th>
<th>Docs</th>
<th>Tokens</th>
<th>Avg. Tokens</th>
<th>Avg. Words³</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>142 819</td>
<td>34 536 077</td>
<td>242</td>
<td>212</td>
</tr>
<tr>
<td>GPHD</td>
<td>161 336</td>
<td>45 472 084</td>
<td>282</td>
<td>247</td>
</tr>
<tr>
<td>Total</td>
<td>304 155</td>
<td>80 008 161</td>
<td>263</td>
<td>230</td>
</tr>
</tbody>
</table>

Note that all articles are published in 1994-1995 and in total they contain roughly 75 million running words distributed over the 304 000 documents.

¹An agency that collects news and sells short texts to newspapers.
²More information and experiments using this collection can be found in Ahlgren (2004a).
To get an idea of the amount of data, this is actually complete files with all articles during one and a half year in the two daily newspapers, respectively, and almost the same amount of running text from TT. It is good to have data from more than one source since we at least in some cases could expect to find original texts from more than one journalist about the same news. Unfortunately, the newspapers often use articles from TT, but they will in many cases be at least a little bit rewritten.

11.2.2 The set of topics

In total I use 102 topics,\(^4\) and all of them are made to match news collections for the years 1994 and 1995. They are taken from two different sets of topics. The first one is a subset of the topics released at CLEF 2000, designed to match the GPHD document collection. The most difficult, for example topics that rely on a direct parse of the query, and the ones less probable to result in a reasonably large set of known relevant documents in the document collection, were removed. These remaining topics still vary in difficulty. The second set of topics was delivered together with the TT collection from CLEF in 2002. All topics of the original set of 50 are used, and they look similar to the GPHD topics. Here are a few examples (figure 11.1, figure 11.2, and figure 11.3) that are included among the 102 topics used in these experiments. The narratives are sometimes tricky. Topic 11.2, for example, has a second narrative sentence that contradicts the first.

**Topic number: C098**

- **Title:** Filmer av bröderna Kaurismäki
- **Description:** Sök efter information om filmer som regisserats av någon av de båda bröderna Aki och Mika Kaurismäki.
- **Narrative:** Relevanta dokument namnger en eller flera titlar på filmer som regisserats av Aki eller Mika Kaurismäki.

Figure 11.1: Topic C098 from CLEF

Note that a topic is divided into three parts: Title, Description, and Narrative, which more in detail defines what a relevant document should contain.

11.2.3 The relevance-judgements

For each topic, a set of documents, selected by the pooling process (section 5.3.2.2), are relevance-judged. I will not go into detail about the manual annotation of the document sets for each topic, but it was performed

\(^3\) A word is defined as a sequence containing at least one alphanumeric character.

\(^4\) 102 is the number of topics, but this number will be revised in section 11.2.3 to 101 because of the fact that one topic has no relevant documents.
Topic number: C102

- **Title:** Alberto Tombas segrar
- **Description:** Sök efter dokument som redogör för skidtävlingar som Alberto Tomba vunnit.
- **Narrative:** Endast dokument som omnämner tävlingar där Tomba har vunnit är relevanta. Alla dokument som tar upp någon typ av skidtävling i någon typ av mästerskap anses relevanta. Det är inte nödvändigt att dokumentet uttryckligen nämner vilken typ av tävling det rör sig om så länge Tomba nämns som vinnaren.

Figure 11.2: Topic C102 from CLEF

Topic number: C118

- **Title:** Finlands första EU-kommissionär
- **Description:** Vem utsågs att vara den första EU-kommissionären för Finland i Europeiska unionen?
- **Narrative:** Ange namnet på Finlands första EU-kommissionär. Rellevanta dokument kan också nämna sakområdena för den nya kommissãorens uppdrag.

Figure 11.3: Topic C118 from CLEF

by four persons after a short training session. Ahlgren (2004a) gives more details about the relevance-judgements.

11.2.3.1 Degrees of relevance

The document pools derived from TT are relevance-judged on a binary scale: relevant or irrelevant. The GPHD relevance-judgements on the other hand, are made as Sormunen (2002) suggests with four levels of relevance (see table 11.2).

11.2.3.2 Statistics for the relevance-judgements

It turns out that some of the topics have very few known relevant documents. Table 11.3 shows the number of topics on different levels of relevant documents. For example, we can see that 17 topics out of 102 have between 6 and 10 relevant documents and that 4 topics have between 41 and 75 irrelevant documents. What we can also see is that one topic has no relevant document at all, so this topic is removed from the collection in the following tests and tables. The metrics for retrieval effectiveness used in this thesis are not meaningful in the case when the number of relevant (or irrelevant)
Table 11.2: Four level relevance scale (Sormunen, 2002)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>1</td>
<td>Marginally relevant</td>
</tr>
<tr>
<td>2</td>
<td>Fairly relevant</td>
</tr>
<tr>
<td>3</td>
<td>Highly relevant</td>
</tr>
</tbody>
</table>

documents is zero—all rankings of the pool are just as good.

Table 11.4 gives an overview of the number of documents in each relevance category for the GPHD data. An overview of the number of documents in each relevance category for the TT data can be found in Table 11.5. Table 11.6 gives an overview of the number of documents in each relevance category for the set of topics including both the GPHD and TT topics—in total 102.

11.2.4 Training of new models for document retrieval

What I am most interested in is to see what happens to the performance when adding the \( n \)-grams to the model. Since document retrieval is a totally different task than the synonym test, we have to keep all training parameters from section 9.4 in mind. The results in section 10.2 give us some rules of thumb that we can use here, like for example that a bigger matrix gives better results. With the GPHD and TT document databases we run into more performance problems than in the synonym test experiments since these training sets are much bigger. The GPHD collection contains 934 000 lower case word types and GPHD+TT more than 1.2 million types, compared to only 93 000 for Bring+Lexin (table 10.1 in section 10.2.1). The number of tokens (running words) is much bigger now (80 million for GPHD + TT compared to 0.34 million for Bring + Lexin), but this is not that much of a problem since it does not increase the size of the matrix. The larger number of tokens will on the other hand increase the number of co-occurrences which forces the sparse matrix representation to contain much more non-zero elements. The number of word types increases when I
start to count $n$-grams as words, so I have to do something about the training data to be able to run the experiments with the computational power I have access to. I experimented with smaller training data collections extracted by randomly selecting a share of all relevance judged documents and a smaller share of the non-judged ones. Table 11.7 shows the training corpora I am going to use in section 11.2.5. The creation of these corpora is done by simply selecting each document with a probability of the wanted share, which will result in roughly the wanted shares.

11.2.5 Overall evaluation results

Now I have the tools needed to investigate what happens with the results in the document retrieval tasks when we add the $n$-grams. First I will go through the parameters to make sure to really answer the right question, but we already made some observations about the parameter settings from the synonym test experiments in chapter 10.

As I have mentioned, it is difficult to calculate if the difference between two systems is significant. One result from section 7.6 was that for the 100 topics collection used in TREC-12, an absolute difference of 3% corresponds to a significant difference at the 0.05 level. Even if this number may vary between collections, we can at least get a hint whether an absolute difference is significant or not, by comparing it to the value 3%. The size of my collection, 101 topics, is almost the same as TREC-12.
Table 11.4: Overview of relevance-judgements for the GPHD data

<table>
<thead>
<tr>
<th>Tag</th>
<th>Relevance</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Irrelevant</td>
<td>153.0</td>
<td>69.7</td>
<td>7</td>
<td>296</td>
<td>7 958</td>
</tr>
<tr>
<td>1</td>
<td>Marginally relevant</td>
<td>19.5</td>
<td>22.4</td>
<td>0</td>
<td>104</td>
<td>1 013</td>
</tr>
<tr>
<td>2</td>
<td>Fairly relevant</td>
<td>9.1</td>
<td>9.7</td>
<td>0</td>
<td>40</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>Highly relevant</td>
<td>7.7</td>
<td>9.9</td>
<td>0</td>
<td>44</td>
<td>402</td>
</tr>
<tr>
<td>1-3</td>
<td>Relevant</td>
<td>36.4</td>
<td>36.2</td>
<td>2</td>
<td>140</td>
<td>1 890</td>
</tr>
<tr>
<td>1-4</td>
<td>Total</td>
<td>189.4</td>
<td>54.2</td>
<td>110</td>
<td>308</td>
<td>9 848</td>
</tr>
</tbody>
</table>

- Relevant in this table means: at least marginally relevant.

Table 11.5: Overview of relevance-judgements for the TT data

<table>
<thead>
<tr>
<th>Tag</th>
<th>Relevance</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Irrelevant</td>
<td>232.9</td>
<td>83.2</td>
<td>96</td>
<td>387</td>
<td>11 645</td>
</tr>
<tr>
<td>1</td>
<td>Relevant</td>
<td>23.9</td>
<td>25.6</td>
<td>0</td>
<td>106</td>
<td>1 196</td>
</tr>
<tr>
<td>0 &amp; 1</td>
<td>Total</td>
<td>256.8</td>
<td>77.7</td>
<td>98</td>
<td>390</td>
<td>12 841</td>
</tr>
</tbody>
</table>

11.2.5.1 Corpus choice

Since the full training corpora of TT and GPHD are so large, I will try to use a smaller portion from them, namely the corpora defined in table 11.7. We will now see how well LSVMs trained with the different subcorpora perform compared to the full TT+GPHD. If the performance for the smaller corpora is roughly the same, I will not have to use the full corpus, which enables me to run more tests.

In table 11.8 the LSVMs were trained using only words and a corpus converted to lower case only, Con set to 100 or 1000, Dim set to either 100 or 300. Vocab and CoVoc are varied to keep a reasonably large matrix and at the same time keep the data small enough to be able to calculate the SVD. Note that the values for RankEff, MAP and bpref-10 are given as the percentage of the maximum value 1.0, and to save space I have written CV instead of CoVoc, BP10 for bpref-10 and REff for RankEff in the table.

I will rely on the RankEff metric since it seems to be a better metric according to chapter 7, but the MAP and bpref-10 are also included in table 11.8. One thing we can see now is that the only subcorpus that is clearly worse than the full TTGPHD all is TTGPHD 0, so I can stick to TTGPHD 1, which is much more handy than the full TTGPHD all for experiment series. An alternative to TTGPHD 1 is to use the full TTGPHD all but with a smaller CoVoc that also seems to give good results.
11 Document retrieval evaluation

Table 11.6: Overview of relevance-judgements for the full data set where GPHD and TT are merged together

<table>
<thead>
<tr>
<th>Tag</th>
<th>Relevance</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>irrelevant</td>
<td>192.2</td>
<td>83.8</td>
<td>7</td>
<td>387</td>
<td>19 603</td>
</tr>
<tr>
<td>1</td>
<td>Relevant</td>
<td>30.3</td>
<td>31.8</td>
<td>0</td>
<td>140</td>
<td>3 086</td>
</tr>
<tr>
<td>0 &amp; 1</td>
<td>Total</td>
<td>222.4</td>
<td>71.0</td>
<td>98</td>
<td>390</td>
<td>22 689</td>
</tr>
</tbody>
</table>

Table 11.7: Overview of subcorpora for Swedish document retrieval

<table>
<thead>
<tr>
<th>Name</th>
<th>Judged</th>
<th>Not j.</th>
<th>Documents</th>
<th>Tokens</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTGPHD$_{all}$</td>
<td>100%</td>
<td>100%</td>
<td>304k</td>
<td>80M</td>
<td>1208k</td>
</tr>
<tr>
<td>TTGPHD$_5$</td>
<td>100%</td>
<td>5%</td>
<td>34.2k</td>
<td>9.6M</td>
<td>327k</td>
</tr>
<tr>
<td>TTGPHD$_2$</td>
<td>40%</td>
<td>2%</td>
<td>13.5k</td>
<td>3.7M</td>
<td>189k</td>
</tr>
<tr>
<td>TTGPHD$_1$</td>
<td>20%</td>
<td>1%</td>
<td>6.7k</td>
<td>1.9M</td>
<td>129k</td>
</tr>
<tr>
<td>TTGPHD$_0$</td>
<td>2%</td>
<td>0.01%</td>
<td>0.45k</td>
<td>0.13M</td>
<td>22k</td>
</tr>
</tbody>
</table>

11.2.5.2 Upper and lower case

The case-option was a little bit difficult to verify for the synonym test since almost all test and training data were in lower case already, so I will just verify that lower case gives better result than keeping both upper and lower case, or at least not worse results.

In table 11.9 I have compared a few models trained in lower case only or with case differences kept. Dim is set to 100 and Con to 1000.

The difference between the $\text{RankEff}$-values for lower case models and case sensitive models is high enough to be significant both for TTGPHD$_1$ and TTGPHD$_{all}$, so I will use lower case from now.

11.2.5.3 Adding bigrams and trigrams

I have now looked at the corpus choice and case options for the Swedish document retrieval task. Table 11.8 shows that the results are roughly the same for 100 and 300 dimensions, and Con=1000 seems to be a good setting.

I have trained many more combinations than the ones presented here and a fair combination of settings that is not too time or RAM consuming is: Vocab=1000k, CoVoc=10k, Con=1000, Dim=100 or 300 and to use the TTGPHD$_1$ corpus. Table 11.10 shows the results for $M_W$, $M_T2$ and $M_T3$.

For $M_W$ and $M_T2$ a Vocab of 1000k was enough but for $M_T3$ I had to try a bigger maximum Vocab to catch all word types, but this did not improve the result. As we can see in the table, the results get worse when we add the bigrams and even worse with bigrams and trigrams added. Another option to try is to separate the vector models for each tuple length, to get $M_{T3sep}$ and $M_{T2sep}$ that were defined in section 9.2. Table 11.11 shows the results for these models compared to the word-based model. The parameters are the same as in table 11.10 except that Dim is set to 100 for all models.
### Table 11.8: Results for subcorpora of various sizes

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>CV</th>
<th>Dim</th>
<th>Con</th>
<th>REff</th>
<th>MAP</th>
<th>BP10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTGPHD_{all}</td>
<td>300k</td>
<td>5k</td>
<td>100</td>
<td>100</td>
<td>81.2</td>
<td>44.8</td>
<td>45.9</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>100k</td>
<td>5k</td>
<td>300</td>
<td>100</td>
<td>81.7</td>
<td>47.0</td>
<td>49.6</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>300k</td>
<td>5k</td>
<td>300</td>
<td>100</td>
<td>82.2</td>
<td>48.6</td>
<td>50.6</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>300k</td>
<td>10k</td>
<td>300</td>
<td>100</td>
<td>82.2</td>
<td>48.5</td>
<td>50.6</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>1208k*</td>
<td>1k</td>
<td>300</td>
<td>1000</td>
<td>82.0</td>
<td>49.3</td>
<td>51.0</td>
</tr>
<tr>
<td>TTGPHD_{5}</td>
<td>327k*</td>
<td>1k</td>
<td>100</td>
<td>1000</td>
<td>81.3</td>
<td>47.4</td>
<td>49.0</td>
</tr>
<tr>
<td>TTGPHD_{5}</td>
<td>327k*</td>
<td>1k</td>
<td>300</td>
<td>1000</td>
<td>81.7</td>
<td>49.4</td>
<td>50.7</td>
</tr>
<tr>
<td>TTGPHD_{2}</td>
<td>189k*</td>
<td>1k</td>
<td>300</td>
<td>1000</td>
<td>80.2</td>
<td>47.0</td>
<td>48.8</td>
</tr>
<tr>
<td>TTGPHD_{2}</td>
<td>129k*</td>
<td>10k</td>
<td>1k</td>
<td>1000</td>
<td>80.4</td>
<td>48.1</td>
<td>49.9</td>
</tr>
<tr>
<td>TTGPHD_{1}</td>
<td>129k*</td>
<td>10k</td>
<td>300</td>
<td>1000</td>
<td>81.8</td>
<td>48.8</td>
<td>50.8</td>
</tr>
<tr>
<td>TTGPHD_{1}</td>
<td>129k*</td>
<td>10k</td>
<td>100</td>
<td>1000</td>
<td>82.3</td>
<td>47.0</td>
<td>49.5</td>
</tr>
<tr>
<td>TTGPHD_{0}</td>
<td>22k*</td>
<td>1k</td>
<td>100</td>
<td>1000</td>
<td>71.1</td>
<td>32.0</td>
<td>32.8</td>
</tr>
<tr>
<td>TTGPHD_{0}</td>
<td>22k*</td>
<td>1k</td>
<td>300</td>
<td>1000</td>
<td>71.6</td>
<td>34.0</td>
<td>34.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M_{Key}</th>
<th>low</th>
<th></th>
<th></th>
<th></th>
<th>66.4</th>
<th>30.4</th>
<th>30.0</th>
</tr>
</thead>
</table>

### Table 11.9: Comparison of results for lower case and case sensitive models

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Case</th>
<th>Vocab</th>
<th>CoVoc</th>
<th>RankEff</th>
<th>MAP</th>
<th>bpref</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTGPHD_{1}</td>
<td>low</td>
<td>129k*</td>
<td>10k</td>
<td>82.3</td>
<td>47.0</td>
<td>49.5</td>
</tr>
<tr>
<td>TTGPHD_{1}</td>
<td>keep</td>
<td>132k*</td>
<td>10k</td>
<td>75.8</td>
<td>35.3</td>
<td>36.8</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>low</td>
<td>1208k*</td>
<td>1k</td>
<td>82.0</td>
<td>49.3</td>
<td>51.0</td>
</tr>
<tr>
<td>TTGPHD_{all}</td>
<td>keep</td>
<td>1239k*</td>
<td>1k</td>
<td>76.8</td>
<td>36.2</td>
<td>38.3</td>
</tr>
<tr>
<td>M_{Key}</td>
<td>low</td>
<td></td>
<td>-</td>
<td>66.4</td>
<td>30.4</td>
<td>30.0</td>
</tr>
<tr>
<td>M_{Key}</td>
<td>keep</td>
<td></td>
<td>-</td>
<td>66.4</td>
<td>30.5</td>
<td>30.2</td>
</tr>
</tbody>
</table>

Note that Vocab now has up to three separate values—one for each tuple length. The weights used in the similarity calculation of $M_{T2sep}$ and $M_{T3sep}$ is set to $c = -2$. The value of $c$ is not optimized except that $c = -2$ gives reasonable results and the $n$-grams get enough weight to make a difference compared to $M_W$.

Unfortunately, all the tuple-enriched models get worse results than the word-based $M_W$. The separated models ($M_{T2sep}$ and $M_{T3sep}$) get a little bit better than the combined ($M_{T2}$ and $M_{T3}$), but they are still slightly worse than $M_W$.

#### 11.2.6 Detailed observations

During the work of developing better models there is a need for comparing more aspects than just the overall relevance score. I need to compare an experimental model both to the baseline and to a more stable best-so-far
Table 11.10: Comparison of results for word-based and n-gram enriched models

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Model</th>
<th>Vocab</th>
<th>Dim</th>
<th>RankEff</th>
<th>MAP</th>
<th>bpref-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTGPHD1</td>
<td>$M_W$</td>
<td>129k*</td>
<td>100</td>
<td>82.3</td>
<td>47.0</td>
<td>49.5</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_W$</td>
<td>129k*</td>
<td>300</td>
<td>81.8</td>
<td>48.8</td>
<td>50.8</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_T2$</td>
<td>805k*</td>
<td>100</td>
<td>78.2</td>
<td>41.7</td>
<td>44.1</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_T2$</td>
<td>805k*</td>
<td>300</td>
<td>78.1</td>
<td>44.4</td>
<td>46.6</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_T3$</td>
<td>1000k</td>
<td>100</td>
<td>76.3</td>
<td>40.7</td>
<td>42.3</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_T3$</td>
<td>1865k*</td>
<td>100</td>
<td>75.7</td>
<td>40.2</td>
<td>41.8</td>
</tr>
</tbody>
</table>

Table 11.11: Evaluation of the models based on separate LSVMs for each n-gram length

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Model</th>
<th>Vocab</th>
<th>RankEff</th>
<th>MAP</th>
<th>bpref-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTGPHD1</td>
<td>$M_W$</td>
<td>129k*</td>
<td>82.3</td>
<td>47.0</td>
<td>49.5</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_{T2}$</td>
<td>805k*</td>
<td>78.2</td>
<td>41.7</td>
<td>44.1</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_{T2sep}$</td>
<td>129k* 805k*</td>
<td>81.1</td>
<td>45.8</td>
<td>47.9</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_T3$</td>
<td>1M</td>
<td>76.3</td>
<td>40.7</td>
<td>42.3</td>
</tr>
<tr>
<td>TTGPHD1</td>
<td>$M_{T3sep}$</td>
<td>129k* 805k* 1M</td>
<td>81.2</td>
<td>45.4</td>
<td>47.8</td>
</tr>
</tbody>
</table>

model. It may be interesting to find out if the experimental system finds a higher number of relevant documents, even if the overall score is the same, or if there are other differences that do not show up in the score.

### 11.2.6.1 Comparing a keyword model and the word-based LSVM

One example of a more detailed comparison between two models is table 11.12, where we have the RankEff differences topic by topic. Only the example topics presented in section 7.5 are shown here. The same kind of statistics could be calculated for all topics and for any evaluation metric. We can see that the $M_W$ is better than $M_{Key}$ in almost all cases.

We may also want to go down to document level—what is happening with particular documents when we switch from one model to another? Let us take a look at a topic with a reasonable number of relevant documents. T20 has 18 relevant and 199 irrelevant documents among the relevance-judged. Table 11.13 shows the movements for relevant documents, and since we are comparing $M_W$ to the baseline $M_{Key}$ we get more improvements than changes for the worse. If we are interested in not just the relevant document movements, the same statistics can be calculated for all documents in the pool. Table 11.14 shows statistics for all movements, summed over all topics, and we can see that the models are very different. A high number of documents have moved more than 50 positions.
11.2 The test suite of Swedish newspaper articles

Table 11.12: \textit{RankEff} difference between two models calculated per topic

<table>
<thead>
<tr>
<th>Topic</th>
<th>\textit{RankEff}</th>
<th>Abs.diff</th>
<th>Rel.diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>T120</td>
<td>0.765</td>
<td>0.167</td>
<td>21.8%</td>
</tr>
<tr>
<td>T57</td>
<td>0.569</td>
<td>0.425</td>
<td>74.8%</td>
</tr>
<tr>
<td>T61</td>
<td>0.582</td>
<td>0.393</td>
<td>67.6%</td>
</tr>
<tr>
<td>T18</td>
<td>0.554</td>
<td>0.277</td>
<td>50.0%</td>
</tr>
<tr>
<td>T58</td>
<td>0.798</td>
<td>0.079</td>
<td>9.8%</td>
</tr>
<tr>
<td>T20</td>
<td>0.821</td>
<td>0.097</td>
<td>11.8%</td>
</tr>
<tr>
<td>T72</td>
<td>0.829</td>
<td>-0.021</td>
<td>-2.5%</td>
</tr>
<tr>
<td>T7</td>
<td>0.520</td>
<td>0.415</td>
<td>79.8%</td>
</tr>
<tr>
<td>T45</td>
<td>0.579</td>
<td>0.068</td>
<td>11.7%</td>
</tr>
<tr>
<td>T21</td>
<td>0.521</td>
<td>-0.071</td>
<td>-13.6%</td>
</tr>
<tr>
<td>T48</td>
<td>0.472</td>
<td>0.469</td>
<td>99.2%</td>
</tr>
</tbody>
</table>

Table 11.13: Individual document movement for \(M_W\) compared to \(M_{Key}\), for relevant documents in topic T20

<table>
<thead>
<tr>
<th>Better</th>
<th>Worse</th>
<th>Position</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>Not top-50</td>
<td>51 or more</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Top-50</td>
<td>1-5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Top-50</td>
<td>11-20</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Not top-50</td>
<td>21-50</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Top-50</td>
<td>6-10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Top-50</td>
<td>21-50</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Top-50</td>
<td>51 or more</td>
</tr>
</tbody>
</table>

11.2.6.2 Comparing the word-based LSVM and an LSVM enriched with n-grams (\(M_{T3sep}\))

I will now look at the differences between the word-based \(M_W\) and the tuple enriched \(M_{T3sep}\). The overall score is better for \(M_W\) but can we find any cases where \(M_{T3sep}\) actually gives an improvement? Let us look at the improvements per topic in table 11.15. I have chosen to set \(c = -2\) for this run. The rows are sorted by relative difference between the models and the topics where the difference is less than 5% are excluded. I have included the number of relevant and irrelevant documents since the task of ranking topics with only a few relevant documents is different from the case where there are a high number of relevant documents.

Since \(M_W\) is a better performing model, it is not surprising that the number of topics where \(M_W\) is better than \(M_{T3sep}\) is higher than the opposite.
Table 11.14: Individual document movement for MW compared to MKey, for all documents in all pools

<table>
<thead>
<tr>
<th></th>
<th>Better</th>
<th>Worse</th>
<th>Position</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5289</td>
<td>3109</td>
<td>Not top-50</td>
<td>51 or more</td>
<td></td>
</tr>
<tr>
<td>2789</td>
<td>2088</td>
<td>Not top-50</td>
<td>21-50</td>
<td></td>
</tr>
<tr>
<td>1049</td>
<td>924</td>
<td>Not top-50</td>
<td>11-20</td>
<td></td>
</tr>
<tr>
<td>569</td>
<td>498</td>
<td>Not top-50</td>
<td>6-10</td>
<td></td>
</tr>
<tr>
<td>534</td>
<td>616</td>
<td>Not top-50</td>
<td>1-5</td>
<td></td>
</tr>
<tr>
<td>286</td>
<td>948</td>
<td>Top-50</td>
<td>21-50</td>
<td></td>
</tr>
<tr>
<td>268</td>
<td>382</td>
<td>Top-50</td>
<td>11-20</td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>249</td>
<td>Top-50</td>
<td>1-5</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>220</td>
<td>Top-50</td>
<td>6-10</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>Top-50</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2279</td>
<td>Top-50</td>
<td>51 or more</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>Not top-50</td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I have tried to find patterns, for example important phrases in the topics, that explain why the result for certain topics improve and other get worse, but I have not found any consistent patterns.

11.2.6.3 Results for different degrees of relevance

As a small test of how the LSVM-based models handles highly relevant documents, I present separate results for the degrees of relevance in figure 11.4 using RankEff and 11.5 with bpref-10. Note that separate relevance-judgement scores for different degrees of relevance are only available for the GPHD-topics. The relevance scores are separated for each topic, ranging from the best to the worse topic results according to the RankEff (figure 11.4) or bpref-10 (figure 11.5) values for relevance level 1-3. We can see that the topic ranking is rather different between the relevance thresholds, especially for bpref-10. The average scores for each metric and degree of relevance can be seen in table 11.16, and we can see that the two metrics disagree a bit about whether the results get better or worse when we raise the relevance limit.

11.2.7 Conclusion of the Swedish document retrieval results

Unfortunately, we can only see a small change for the worse when going from the word-based MW to a n-gram enriched model (MT3 or MT3sep). The change is not very big, but it seems to be significant. We can observe improvements for some topics but I have not found any explanations why...

---

50=irrelevant, 1=Marginally relevant, 2=Fairly relevant, 3=Highly relevant
11.2 The test suite of Swedish newspaper articles

Figure 11.4: RankEff for different degrees of relevance for each topic

Figure 11.5: bpref-10 for different degrees of relevance for each topic
Table 11.15: *RankEff* difference between $M_W$ and $M_{T3sep}$ calculated per topic, sorted by relative difference

<table>
<thead>
<tr>
<th>Topic</th>
<th>$RankEff$ $M_W$</th>
<th>$RankEff$ $M_{T3sep}$</th>
<th>Abs.diff</th>
<th>Rel.diff</th>
<th>Rel.s</th>
<th>Irrel.s</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0.566</td>
<td>0.361</td>
<td>-0.204</td>
<td>-36.1%</td>
<td>3</td>
<td>155</td>
</tr>
<tr>
<td>120</td>
<td>0.858</td>
<td>0.716</td>
<td>-0.142</td>
<td>-16.6%</td>
<td>1</td>
<td>162</td>
</tr>
<tr>
<td>133</td>
<td>0.789</td>
<td>0.714</td>
<td>-0.076</td>
<td>-9.9%</td>
<td>17</td>
<td>284</td>
</tr>
<tr>
<td>70</td>
<td>0.912</td>
<td>0.833</td>
<td>-0.078</td>
<td>-8.6%</td>
<td>56</td>
<td>54</td>
</tr>
<tr>
<td>107</td>
<td>0.828</td>
<td>0.760</td>
<td>-0.068</td>
<td>-8.2%</td>
<td>10</td>
<td>242</td>
</tr>
<tr>
<td>39</td>
<td>0.676</td>
<td>0.620</td>
<td>-0.055</td>
<td>-8.2%</td>
<td>20</td>
<td>154</td>
</tr>
<tr>
<td>24</td>
<td>0.818</td>
<td>0.756</td>
<td>-0.062</td>
<td>-7.5%</td>
<td>23</td>
<td>106</td>
</tr>
<tr>
<td>37</td>
<td>0.764</td>
<td>0.717</td>
<td>-0.047</td>
<td>-6.2%</td>
<td>39</td>
<td>123</td>
</tr>
<tr>
<td>132</td>
<td>0.729</td>
<td>0.685</td>
<td>-0.044</td>
<td>-6.0%</td>
<td>33</td>
<td>226</td>
</tr>
<tr>
<td>26</td>
<td>0.764</td>
<td>0.722</td>
<td>-0.042</td>
<td>-5.5%</td>
<td>57</td>
<td>142</td>
</tr>
<tr>
<td>92</td>
<td>0.766</td>
<td>0.726</td>
<td>-0.041</td>
<td>-5.3%</td>
<td>60</td>
<td>108</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.664</td>
<td>0.701</td>
<td>0.037</td>
<td>5.5%</td>
<td>140</td>
<td>54</td>
</tr>
<tr>
<td>71</td>
<td>0.860</td>
<td>0.907</td>
<td>0.048</td>
<td>5.6%</td>
<td>13</td>
<td>187</td>
</tr>
<tr>
<td>137</td>
<td>0.561</td>
<td>0.616</td>
<td>0.054</td>
<td>9.7%</td>
<td>6</td>
<td>358</td>
</tr>
<tr>
<td>121</td>
<td>0.814</td>
<td>0.948</td>
<td>0.134</td>
<td>16.5%</td>
<td>1</td>
<td>97</td>
</tr>
<tr>
<td>Avg</td>
<td>0.823</td>
<td>0.812</td>
<td>-0.0112</td>
<td>-1.5%</td>
<td>30.6</td>
<td>194.1</td>
</tr>
</tbody>
</table>

The results for these topics are improved and not the rest of them. The fact that English compounds are written with spaces separating their parts may favour the results for $n$-gram enriched LSVMs for English compared to Swedish.

### 11.3 The English test suite for document retrieval

In this section I will define the test suite for English. This data collection is very similar to the Swedish collection, but the documents are not just newspaper articles.

The data used form a subset of the TREC data used in section 7.6.1 where I used the test suite and results from TREC competitions. As document collection I use the data from TREC-8 and TREC-12. Both these TREC collections are built up from the same 500 000 documents collection containing roughly 2 gigabytes of text (see table 7.10).\(^6\) I will call this collection...

---

\(^6\)The document collections may be ordered from NIST on two CDs and are called “NIST TREC Document Database: Disk 4 & 5”.  

144
11.3 The English test suite for document retrieval

Table 11.16: Average scores for each degree of relevance

<table>
<thead>
<tr>
<th>Degree of relevance</th>
<th>RankEff ± StdDev</th>
<th>bpref-10 ± StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MW</td>
<td>MKey</td>
</tr>
<tr>
<td>1-3</td>
<td>0.840±0.179</td>
<td>0.658±0.205</td>
</tr>
<tr>
<td>2-3</td>
<td>0.843±0.181</td>
<td>0.693±0.227</td>
</tr>
<tr>
<td>3</td>
<td>0.841±0.188</td>
<td>0.717±0.245</td>
</tr>
</tbody>
</table>

TREC812 since it contains data from TREC-8 and TREC-12.

### 11.3.1 The document database

The documents are mostly newspaper articles as we can see in table 11.17. These documents are on the average almost twice as long as the documents in the Swedish document collection—400 words compared to 230, which is not surprising since the documents for each language are of different types. In total the number of documents is 70% higher than for the Swedish TT+GPHD.

Table 11.17: Overview of the English document database TREC812

<table>
<thead>
<tr>
<th>Collection</th>
<th>Docs</th>
<th>Words</th>
<th>Avg.Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federal Register (1994)</td>
<td>55 631</td>
<td>44.2M</td>
<td>662</td>
</tr>
<tr>
<td>Financial Times (92-94)</td>
<td>210 158</td>
<td>80.7M</td>
<td>384</td>
</tr>
<tr>
<td>Foreign Broadcast Information Service</td>
<td>130 471</td>
<td>65.7M</td>
<td>504</td>
</tr>
<tr>
<td>Los Angeles Times (89-90)</td>
<td>131 896</td>
<td>64.2M</td>
<td>487</td>
</tr>
<tr>
<td>Total</td>
<td>528 156</td>
<td>211M</td>
<td>400</td>
</tr>
</tbody>
</table>

### 11.3.2 The topics and relevance-judgements

The topics are designed in the same way as the Swedish CLEF topics, which is not surprising since the CLEF setup is inspired by the TREC competitions. So each topic contains three parts as in the example topics below. For some topics ordinary keyword search is enough but some topics are really difficult even for advanced IR-systems.

- **Title:** foreign minorities, Germany
  Description: What language and cultural differences impede the integration of foreign minorities in Germany?
  Narrative: A relevant document will focus on the causes of the lack of
integration in a significant way; that is, the mere mention of immigration difficulties is not relevant. Documents that discuss immigration problems unrelated to Germany are also not relevant.

- Title: toxic chemical weapon
  Description: Gather any information that mentions ricin, sarin, soman, or anthrax as a toxic chemical used as a weapon.
  Narrative: To be relevant, a document must pertain to the use of toxic chemicals as weapons.

From the two TREC s I also get 150 topics and totally more than 200 000 relevance-judgements, on the average 1400 per topic, so this data collection has a much better coverage than the Swedish collections used in section 7.5. When the two collections are collapsed into one set of topics, it turns out that some of the topics are used both in TREC-8 and TREC-10 so the resulting number of distinct topics is 133. The number of relevance-judgements also shrinks as a result of the duplicates, to 184 092 distributed on a total of 121 639 distinct documents. The remaining 406 517 documents are not relevance-judged, but used only for training purposes.

### 11.3.3 Training of new LSVMs

I have the same goal with the English TREC test suite as for the Swedish TTGPHD—how will the performance change when we add \( n \)-grams to a word based LSVM? We will have to be careful with the various parameters to be able to measure what we want rather than effects of bad parameter settings. I will skip the step of verifying that converting to lower case gives a better score and instead rely on the results for the Swedish data. The dimensionality and corpus choice is the most crucial parameters, so we will have to verify these parameters first, just like for the Swedish case. I have created subcorpora for TREC812 in the same way, see table 11.18. For this collection, that is bigger than the Swedish collection, it is even more important to be able to run on smaller data. The subcorpora are created in the same way as for the Swedish ones in table 11.7.

<table>
<thead>
<tr>
<th>Name</th>
<th>Judged</th>
<th>Not j.</th>
<th>Documents</th>
<th>Tokens</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC812_all</td>
<td>100%</td>
<td>100%</td>
<td>528k</td>
<td>211M</td>
<td>1036k</td>
</tr>
<tr>
<td>TREC812_5</td>
<td>100%</td>
<td>5%</td>
<td>141.9k</td>
<td>82.3M</td>
<td>698k</td>
</tr>
<tr>
<td>TREC812_2</td>
<td>40%</td>
<td>2%</td>
<td>56.7k</td>
<td>32.9M</td>
<td>365k</td>
</tr>
<tr>
<td>TREC812_1</td>
<td>20%</td>
<td>1%</td>
<td>28.5k</td>
<td>16.7M</td>
<td>251k</td>
</tr>
<tr>
<td>TREC812_0</td>
<td>2%</td>
<td>0.01%</td>
<td>2438</td>
<td>1.41M</td>
<td>64.1k</td>
</tr>
</tbody>
</table>
11.3.4 Overall evaluation results

To ensure that we really measure whether \(n\)-grams give an improvement or not, I will first test the other parameters—just as in the Swedish case in section 11.2. Let me start with the corpus choice.

11.3.4.1 Corpus choice and number of dimensions

\(TREC812_{alt}\) is really a large collection for this kind of calculation. Especially when it comes to adding \(n\)-grams. Table 11.19 gives a comparison of the results for the different subcorpora of \(TREC812_{alt}\). I was not able to use all of it for training. We can see that the results are similar for all sizes of subcorpora except for \(TREC812_0\) that gives clearly worse results. The keyword-based model gave the worst result of all models, but still better than a random ranking (\(RankEff \approx 50\%\)).

Table 11.19: Results for subcorpora of various sizes

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Vocab</th>
<th>CV</th>
<th>Dim</th>
<th>Con</th>
<th>REff</th>
<th>MAP</th>
<th>BP10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TREC812_5)</td>
<td>698k*</td>
<td>10k</td>
<td>100</td>
<td>100</td>
<td>66.7</td>
<td>11.3</td>
<td>9.3</td>
</tr>
<tr>
<td>(TREC812_5)</td>
<td>698k*</td>
<td>10k</td>
<td>300</td>
<td>100</td>
<td>68.7</td>
<td>13.9</td>
<td>11.8</td>
</tr>
<tr>
<td>(TREC812_2)</td>
<td>365k*</td>
<td>3k</td>
<td>100</td>
<td>100</td>
<td>73.3</td>
<td>15.6</td>
<td>14.0</td>
</tr>
<tr>
<td>(TREC812_1)</td>
<td>263k*</td>
<td>1k</td>
<td>100</td>
<td>100</td>
<td>72.7</td>
<td>15.4</td>
<td>13.8</td>
</tr>
<tr>
<td>(TREC812_1)</td>
<td>263k*</td>
<td>3k</td>
<td>100</td>
<td>100</td>
<td>73.1</td>
<td>15.7</td>
<td>14.2</td>
</tr>
<tr>
<td>(TREC812_0)</td>
<td>64k*</td>
<td>1k</td>
<td>100</td>
<td>1k</td>
<td>61.0</td>
<td>9.3</td>
<td>7.0</td>
</tr>
<tr>
<td>(M_{Key})</td>
<td>*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>55.8</td>
<td>6.2</td>
<td>3.1</td>
</tr>
</tbody>
</table>

11.3.4.2 Adding bigrams and trigrams

In this comparison of word-based and models enriched with \(n\)-grams, I have used the \(TREC812_1\) to get rid of the performance problems during the training phase. Unfortunately a new problem arises. A weakness for my version of the Infomap NLP System becomes critical when we are using the full set of topics for TREC-8 and TREC-12, totally 184 000 relevance-judgements in combination with the more complex \(M_T^2\) and \(M_T^3\). I did not realize early enough that the model had to be read once for each relevance-judgement which makes it extremely time-consuming to run. I had to limit the set of topics to 10, but there are still 8 240 relevance-judgements used. Table 11.20 shows results for \(TREC812_1\) with \(\text{Dim=Con}=100\) and \(\text{Covoc}=3k\).

\(RankEff\) shows a slightly better result for \(M_W\) than for \(M_T^2\) and \(M_T^3\), but it is definitely not significant. Actually \(MAP\) shows the opposite, i.e. the worst result for \(M_W\).
<table>
<thead>
<tr>
<th>Model</th>
<th>Vocab</th>
<th>REff</th>
<th>MAP</th>
<th>BP10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_W$</td>
<td>263k*</td>
<td>73.8</td>
<td>13.8</td>
<td>12.2</td>
</tr>
<tr>
<td>$M_{T2}$</td>
<td>332k*</td>
<td>73.0</td>
<td>14.9</td>
<td>11.8</td>
</tr>
<tr>
<td>$M_{T3}$</td>
<td>1M</td>
<td>72.3</td>
<td>14.8</td>
<td>11.3</td>
</tr>
<tr>
<td>$M_{Key}$</td>
<td>*</td>
<td>56.1</td>
<td>6.5</td>
<td>3.2</td>
</tr>
</tbody>
</table>

### 11.3.5 Conclusion of the English document retrieval results

The most important results for the English document retrieval experiments are that $M_{T3}$ and $M_{T2}$ have roughly the same results as $M_W$. The differences are not significant, but a comparison for the combined TREC-8 and TREC-12 as a whole may show a significant difference, but that is not very likely because the results for the smaller subset of topics used in section 11.3.4.2 seems to be big enough to give stable results compared to the full set used in 11.3.4.1.

### 11.4 Summary of the evaluation results for document retrieval

I have compared the $n$-gram enriched $M_{T3}$ and $M_{T2}$ to the word-based $M_W$, both for English and Swedish document retrieval test collections. For Swedish we have seen a noticeable change for the worse, but for English, the differences are not significant. All LSVMs are however clearly better performing than the keyword baseline model. The fact that LSVMs with $n$-grams added seem to work better for English can be a result of the English way of forming compounds compared to Swedish, where no white-spaces are inserted between the parts of the compounds.

---

7 Whether it is significant or not is not entirely clear. The usual statistical tests do not give significance, but the error rate calculations indicate that at least the difference between $M_{T3}$ and $M_W$ is significant.
Part IV

Conclusion
Part IV: Conclusion

The final part of this thesis consists of two chapters. Chapter 12 gives a summary of the contributions from this thesis. Then chapter 13 gives some suggestions for future research.

Chapter 12
The contribution chapter starts with section 12.1 about the synonym test task. The contributions are summarized, including the new SweHP560 synonym test evaluation-set. In section 12.1.2 the results from experiments where LSVMs trained with various parameter settings, including the models with $n$-gram added, are summarized. The conclusion is that adding $n$-grams to an LSVM improves the results for the synonym test task. Section 12.2 concludes the document retrieval-task. For English the results are almost unchanged, but for Swedish we get a clear change for the worse when $n$-grams are added. Section 12.3 is a summary of how the performance of an LSVM changes when $n$-grams are added.

Chapter 13
The results presented so far raise questions and ideas of future research. This chapter is a summary of these ideas and questions. In section 13.1 questions about evaluation metrics are discussed. A large part of this thesis is an investigation about document retrieval metrics for ranking effectiveness, but there are still more to investigate. In section 13.2 I discuss remaining questions about how MWUs can be added to an LSVM. More experiments are needed to find out if there are better ways than the $n$-gram approach to add phrases to an LSVM, and to finally conclude if we can get a general improvement by adding some kind of phrases to an LSVM.


Chapter 12

Contributions

This chapter contains a short summary of the findings that answer the questions formulated in the introduction. I will answer the main question by looking at the results from the two evaluation tasks:

\textit{How will the performance of an LSVM change when MWUs are added?}

12.1 Synonym test evaluation

I will start with the synonym test evaluation. Here I will present both the results and the new SweHP560 synonym test evaluation set.

12.1.1 The SweHP560 synonym test evaluation set

I have developed a new evaluation set that is suitable for LSVM evaluation with or without \( n \)-grams. It contains 560 Swedish synonym queries taken from “Högskoleprovet”. For each query there are five alternatives and exactly one of these is a close synonym to the query. Both queries and alternatives may be phrases, i.e. contain more than one token. See section 6.2 for more details.

For this evaluation set, the choice of evaluation metric is much easier than in the document retrieval case. Percentage of correct answers is a good metric that measures what we want. One baseline and what we can call a “topline” are:

- The expected number of correct answers achieved by a random choice, which for SweHP560 is 20% corresponding to 112 correct answers.

- The average result for humans that perform the test, which range from 55%-60% in 1996–1998 (Stage et al., 1998).

12.1.2 Results

This thesis contains results from the synonym test evaluation for LSVMs trained with various parameter settings, where the initial experiments were needed in order to isolate the effect of \( n \)-grams without mixing it with the
other parameters. In table 10.7 in section 10.2 we can see that a word-based LSVM ($M_W$) achieves a result of 58.2%. When adding bigrams to get model $M_{T2}$ the accuracy rises to 60.4%, and finally the bigram and tri-gram enriched model ($M_{T3}$) gets a result of 61.4%. The improvement from $M_W$ to $M_{T3}$ is significant according to the McNemar-test with $p = 0.02$, but the difference between $M_W$ and $M_{T2}$ is not big enough to be significant ($p = 0.08$). It looks like the optimal dimensionality is higher for the models containing $n$-grams, which is not surprising since they have a much higher number of word types. The negative side of this is that more word types and more dimensions make the computation much heavier, but as I have mentioned in section 2.3.2, there is good hope for more efficient implementation of the training process. We should not rely on more powerful computers in the future since the amount of data to search will grow bigger and bigger at the same time.

In the detailed comparison of $M_W$ and $M_{T3}$ in section 10.3 it turns out that for 61 out of the 560 queries the two models get a different score, i.e. one of them is right and the other one is wrong. 44 (72%) of these are in favour of $M_{T3}$ which is a promising result. More surprising is that roughly the same share of queries gives an improvement for $M_{T3}$ even for single word queries with single word alternatives. This suggests that the $M_{T3}$ finds synonymy better than $M_W$ not just for MWUs but also for single word terms. This suggests that an LSVM enriched with $n$-grams actually is a better semantic space even for single words. The reason for this can be that the $n$-grams give some help by disambiguating ambiguous words, but I have no proofs for that.

For the synonym test I did not use the models with separate vector spaces for words, bigrams and trigram. The combined model is more interesting so I chose not to run these tests.

### 12.2 Document retrieval evaluation

This section presents the results from the document retrieval evaluation, but I will start with some information about my newly developed evaluation metrics.

#### 12.2.1 New evaluation metrics

As a complement to the standard metrics for document retrieval evaluation, there is a need for metrics that are aware of missing relevant judgements in typical IR test suites like the ones used in TREC and CLEF. The $bpref$-metrics presented in Buckley and Voorhees (2004) work to some extent, but my metrics $RankEff$ and $WRS$ seem to work better in many cases. If this is really what we want to measure is difficult to say, but $RankEff$ (and $WRS$) have many of the properties that the traditional metrics do not have.
12.2 Document retrieval evaluation

- *RankEff* uses more information than *bpref*, since *each* known irrelevant document is taken into consideration.

- It handles topics with a small (or extremely large) number of relevant documents better than *bpref*-10 in the sense that unreasonably large differences in measurement values between models are prevented.

- Compared to *WRS*, it is designed to handle the problem with missing relevance-judgements introduced when we go from the ideal Cranfield environment to more realistic evaluation like TREC and CLEF.

- If a relevant document swaps position with a higher ranked irrelevant one, the score always rises, which is not the case for the *bpref*-metrics

Baselines to use for document retrieval evaluations using *RankEff* or *WRS* are:

- The expected value for each metric achieved by a random ranking. For *RankEff* and *WRS* it will be 0.5.

- A more competitive baseline is to calculate the metrics for a simple keyword search (here called *MKey*) where the documents are ranked by the number of relevant terms in each document.

The significance testing is difficult since the conditions needed for normal distribution based tests are not fulfilled. Experiments performed in Zobel (1998) and Buckley and Voorhees (2000) show that the Wilcoxon test can be used anyway together with *bpref* and the traditional metrics. For the typical IR test suites a big difference in accuracy is needed to be significant. I hope that *RankEff* can be shown to be more stable, which is indicated by the experiments in section 7.6.2 on error rates, and therefore a smaller difference will be significant, but there is more theoretical work needed here.

### 12.2.2 Evaluation results

The experiments have been performed both for English and Swedish. The Swedish test data consist of two document collections and topic sets from CLEF, totally 101 topics and more than 22 000 relevance judgements (3 000 of these were relevant), and the English runs are made on a document collection and two topic sets from TREC. I call this combined collection TREC812, containing 133 topics and 184 000 relevance judgements from which 9 000 are relevant. The results show that for the Swedish data there is a small, but noticeable change for the worse when adding bigrams or bigrams and trigrams. This is disappointing but the models can still be useful since the change is not very big. For English the results are a bit better. They show roughly the same scores for the word-based model and the *n*-gram enriched ones, which means that we now have an LSVM that contains *n*-grams without getting a worse result than for a word-based LSVM.
We can conclude that the synonym test evaluation gives a significant improvement for the \( n \)-gram enriched LSVM, but for document retrieval we get changes for the worse, especially for Swedish. In next chapter I will discuss possible reasons for these poor results. On the other hand, an LSVM that contains \( n \)-grams can be more useful even if it is not performing better than a word-based model since it gives the possibilities of searching for MWUs, that are not captured by the individual words. Topics described by such MWUs are impossible to find in a word-based LSVM.

### 12.3 N-grams in LSVMs

We can conclude that in the results for the SweHP560 synonym test collection, there is a tendency of improvement for all kinds of queries and answers, i.e. single words, MWUs, or mixed, when we go from a word-based LSVM to the LSVM enriched with up to three tokens long \( n \)-grams. The difference between the word-based model and the one with \( n \)-grams up to length 3 is significant according to McNemar’s test. One reason for this can be that the \( n \)-grams help during the training to resolve ambiguities.

For the document retrieval task I have been working on both Swedish and English. It may be a problem that I do not have access to a completely tuned retrieval system. For example the topic to query conversion is very rudimentary, which could affect the results. The overall result show a change for the worse when I include \( n \)-grams in the model for both the \( M_{T3} \) and \( M_{T3sep} \) models compared to \( M_W \), but this is not the case for English where \( M_{T3} \) and \( M_W \) give similar results.

At this point we should recall the arguments that made me choose \( n \)-grams instead of syntactic or statistical phrases (section 9.1.3). What can we say now when the results are available?

1. **Completeness**
   The improvement for the synonym test task shows that the model gains something from the \( n \)-grams. Many of the queries and answer alternatives in section 10.3 (mainly in table 10.13) are actually of the kind that they would not be considered as phrases by the rivaling methods. Some of them are not complete units and some have a far to low frequency in the training data. The worse results for document retrieval may indicate that non-MWU \( n \)-grams worsen the models, but we do not have enough evidence for that.

2. **Efficiency** and **Stop criteria**
   The importance of these issues can be tested when we have access to the different alternative implementations of phrase-extraction.

3. **Language-independence** and **Simplicity**
   I think that these arguments still hold, but further experiments can of course show that better results can be obtained with syntactic or statistical phrases.
4. **Training artifacts**

The *Training artifacts* argument is still an open question, until we can compare the results to other method of extracting MWUs.

Even if the overall results from the evaluations do not give a clear improvement, it is interesting to be able to include \( n \)-grams in an LSVM. There are contents we may want to search for in the model that are simply not expressible with single words, for example full names where both the given name and the family name are rather common. In this case \( n \)-grams will enable disambiguation between different people, but since these cases are not common enough in any of the evaluation data, this feature will give no significant improvement. The lack of improvement can also depend on the fact that adding \( n \)-grams to the model is too coarse and that a more elaborated way of finding the right MWUs to add can give these improvements.


Chapter 13

Future research

The work presented in this thesis raises many new questions. Some main interesting areas are presented in this chapter. I should also mention that more evaluation test suites covering many languages and domains are always important contributions to the research community.

13.1 Evaluation metrics

I have made a substantial investigation of modern document retrieval effectiveness metrics and also presented two new ones: \textit{RankEff} and \textit{WRS}.

13.1.1 Different information needs

Even if my metrics seem to measure what I want, there is still a need for tailored metrics for various kinds of information needs. We can see two extremes among information needs:

1. A user that wants to find the answer to a question as quickly as possibly. He wants just one document that contains the answer—the rest of the list is not interesting at all.

2. A user that wants all information sources on a specific topic. He needs all existing relevant documents.

Sufficient evaluation metrics for the two extremes are not very difficult to invent, but there is a continuum of information needs between these. A general metric of retrieval effectiveness, like \textit{RankEff} or \textit{MAP}, that suits most information needs is useful but probably we also need to investigate what kind of information needs there are and also how common they are among users of document retrieval systems like Google all over the world.

13.1.2 Calculating average over \textit{RankEff}

One thing not yet investigated is how well \textit{RankEff} works as an average over topics. The question of how much impact each topic of different difficulty levels and amount of relevant documents should get in an average value based on \textit{RankEff} should be further investigated. The same question comes
with the $b_{\text{pref}}$-metric while $MAP$ handles the problem in a way easier to understand.

### 13.1.3 Significance testing

More work could also be done on a significance test that is useful for the retrieval effectiveness metrics used in this thesis, but of course also more testing of RankEff and $WRS$ on other test suites to see if it really outperforms $MAP$ and $b_{\text{pref}}$-10 as a metric to use in environments like TREC and CLEF.

### 13.2 LSVMs enriched with n-grams

This section takes up some aspects of MWUs that are not yet enough investigated. I still believe in the idea of adding $n$-grams to get more powerful LSVMs, but more research is needed.

#### 13.2.1 Different ways to add MWUs

What I did not have time to do in this thesis work was to test more different ways to find the MWUs to include, and how to add them to the LSVM. I argue in section 9.1.3 that a strength for $n$-grams in favour of more elaborated phrases, is that all phrases are present among the $n$-grams. The problem is that we do not know how much damage the rest of the $n$-grams do to the models, which has to be further investigated.

Various statistical collocations and maybe a dependency parser could be used, for example with the fast and freely available MaltParser developed by the MALT-group at Växjö University (Nivre et al., 2006). A strength with the parser approach is that we will then be able to find non-juxtapositional phrases, i.e. phrases that do not consist of an unbroken sequence of words. These phrases are not even found by the $n$-gram approach.

#### 13.2.2 Different ways to train the LSVMs

A problem during this work has been the lack of computational power and enough RAM-memory. The solution is not to buy a more powerful computer, but testing better ways to do the training of the models. More efficient algorithms could make it possible to run on bigger materials without limitations of the maximum vocabulary size and the number of dimensions.

#### 13.2.3 More testing is needed

I think that the evaluation data used in this thesis in combination with the evaluation metrics form a good basis for a much more extensive investigation. Processing time and lack of RAM-memory have limited the number of runs possible in this work, but there is also a need of better tokenization, maybe stemming, different ways to add the MWUs and better query
construction from the topics. More knowledge and time to improve these things in combination with more tests will probably give very interesting results. Until I have seen such tests, I will not give up the idea of LSVMs enriched with MWUs or $n$-grams.
Bibliography


