Using Contextualized Lexical Expansion for Information Retrieval

Master-Thesis von Richard Steuer
September 15, 2013

TECHNISCHE UNIVERSITÄT DARMSTADT

UBIQUITOUS KNOWLEDGE PROCESSING

Using Contextualized Lexical Expansion for Information Retrieval

vorgelegte Master-Thesis von Richard Steuer

Supervisor: Prof. Dr. Chris Biemann
Coordinator: Dr. Jungi Kim

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. September, 2013

(R. Steuer)

Zusammenfassung

In Information-Retrieval-Systemen (IR) drückt der Benutzer sein Informationsbedürfnis durch eine Suchanfrage (engl. *query*) aus. Diese Anfrage wird dann mit einer Sammlung von Dokumenten abgeglichen, um für den Benutzer relevante Informationen zu finden. Die Dokumente sind von dem IR-System vorverarbeitet worden. Sie sind für gewöhnlich sehr groß und repräsentieren Informationen, die von anderen Menschen zum Ausdruck gebracht worden sind. Das IR-System gleicht diese beiden Repräsentationen der Texte miteinander ab, die möglicherweise sehr verschieden formuliert sind. An dieser Stelle ist oft die Vokabularlücke (engl. *vocabulary gap*) ein Problem, die daraus resultiert, dass die gleichen Konzepte auf verschiedene Art und Weise zum Ausdruck gebracht werden können. Das heißt, die Dokumente können vom gleichen Thema wie die Suchanfrage handeln, werden aber aufgrund einer geringen lexikalischen Überschneidung vom System nicht gefunden. Synonymie (einzelne Wörter haben die gleiche oder ähnliche Bedeutung) und Polysemie (einzelne Wörter haben mehrere Bedeutungen) sind zwei weitere Probleme dabei, herauszufinden, was der Benutzer meint. Um diese Probleme anzugehen, werden in der vorliegenden Arbeit drei lexikalische Expansionsquellen angewandt, um diese lexikalische Lücke durch Erweiterung der lexikalischen Repräsentationen zu überbrücken und dadurch das Suchergebnis zu verbessern. Die Ergebnisse werden auf zwei Korpora evaluiert, um jegliche Änderung zu messen. Die Quellen reichen von distributionellen Ressourcen für jede Wortart bis hin zu kontextuellen, die sich auf eine beschränken. Die lexikalischen Expansionen (oder Ersetzungen) werden auf Seite der Anfrage und auf Seite der Indexdokumente angewandt und unterschieden nach der Wortart (soweit möglich), pro Wort und Satzkontext, nach verschiedenen Gewichtungen und der Anzahl an hinzugefügten Expansionen. Die Ergebnisse zeigen insgesamt nur kleine Verbesserungen, lassen aber vielversprechende Möglichkeiten für künftige Forschung erkennen.

Abstract

In Information Retrieval (IR) systems, a user expresses an information need by a so called *query*. This query is then matched against a collection of documents to retrieve information relevant to the user's request. The documents have been pre-processed by the system. They usually are very large and represent information expressed by many other people. The IR system matches both the representations, expressed by possibly many different words and phrases. It is here, where the *vocabulary gap* often is a problem, resulting from the fact that concepts have several ways to be expressed in terms. That is, both the query and a document might be about the same issue, but due to a low lexical overlap are not discovered by the IR system. Synonymy (single words having the same or similar meaning) and polysemy (single words having multiple meanings) are two more problems when identifying what the user meant. To address these problems, in this thesis three lexical expansion resources were applied to bridge that gap by enhancing each of those lexical representations and thereby to improve the system's retrieval performance. The results are evaluated on two corpora to assess any changes. The resources vary from distributional all-words to contextual ones limited to a certain part-of-speech (POS). The lexical expansions (or substitutions) provided are applied at the query and the index documents, distinguished by POS (as far as possible), per term and sentence context, by different weights and the amount of expansion terms added. The results show only small overall improvements, but reveal promising avenues for future investigation.

Contents

1 Introduction

Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it.

(Samuel Johnson)

Looking for information on the world wide web today (be it news, about people or of encyclopedic nature) is usually starts with navigating to a search engine. That is seen as a convenient way which doesn't require much manual effort, compared to browsing an encyclopedia for instance. The search engine offers a small input field to the user in what several keywords or sentences can be entered. The system then returns a list of results, which is presented to the user page by page. Additionally, the results are expected to be ordered by importance as to the input. Of course, the results are based on the terms the user has supplied in the first place. Entering different terms is supposed to result in different answers from the system. The underlying search engine seems to know what the user is looking for, but internally it compares the user's terms with the information it "knows". That is, it tries to compare the user's request representation (expressed by its query) with the representations of all the documents and resources it has pre-processed. There are multiple ways this matching can be done by the system. But all of them are representing the concepts (the user has in mind, the documents) in terms and words. That is, in its simplest form they try to match the exact same words in both the representations. This is a source of many problems. The reason is the nature of words itself.

First, a language may offer many closely related terms to express the same or similar issue (e.g. *buy* and *purchase*, *big* and *large*). On the other hand, each word in turn may have multiple meanings (polysemy, e.g. *bank* can either be a financial institution or the edge of a river). Thirdly, even if there should be no complexity in verbally expressing an issue, a user might not have any experience in doing so. In the following section, some mechanism ("models") are introduced to perform that matching between the query and the documents. Many of the models often have problems dealing with synonymy (words having the same or similar meaning) and related terms. For example, one would expect a search for the term *aircraft* also match the terms *airplane* and *plane*. This is called the *vocabulary problem* [Furnas et al., 1987], also called *lexical gap*, and is caused by concepts having several ways to be expressed in words. Additionally, words can have multiple meanings (ambiguity) and different words can have the same meaning (synonymy, e.g. "tv" and "television"). Polysemy may cause erroneous retrieval results in terms of irrelevant documents, and synonymy may in effect lead to a decrease of relevant documents. Several approaches have been proposed to address this problem, including word sense disambiguation (WSD), interactive query refinement or even the possibility of the user giving feedback on the search for the system to re-calculate a better representation of what the user meant. But since the users tend to avoid putting in extra work, the goal is to do this automatically, i.e. with as little user interaction required as possible. That is why we want to explore the possibilities of what is called *lexical expansion*, that is enhancing the textual basis on what to work with, in our case on the query and document side. The idea of lexical expansion is to add extra terms to the original words that are considered to be related, associated or even equivalent (synonymous). For example, consider the term *meeting*. Associated terms might be *hearing*, *session*, *conference*, *summit* or *workshop*. The purpose of finding related terms is to better capture the actual intent behind the given words. In the search domain (called *Information Retrieval* in science), the expectation is to engage the vocabulary problem and therefore to cover more results more accurately, since they may be phrased in distinct words. So, the overall goal from the viewpoint of the user is to improve the retrieval in quantity (number) or quality (order). Still, to come up with extra terms (expansions) some kind of resource is needed to tell which terms are related (and how much). Usually, "handmade" lexical-semantic resources are employed for such a purpose. Such a resource may group and connect terms and provide additional information (e.g. definitions or examples).

To address these problems, this thesis applies three complex lexical expansion resources that are composed in very different ways. In particular, this includes resources from the *statistical semantics* domain. The research question of this thesis is to examine the hypothesis, if lexical expansion can bridge the lexical gap in Information Retrieval and measurably improve the retrieval performance. To achieve this, experiments with and without lexical expansions are conducted and evaluated on two different data sets. As pointed out, the search process roughly has to starting points, that is on the one hand the documents to be searched against and second the user entered search string. Two problems arise from that. The first is the collection of the documents being large, and the second the queries being short. According to *Hitwise*[1], the average query length in 2009 was 2.3 terms. [Taghavi et al., 2012] analysed about 40 million queries in the period from June 2010 to February 2011, collected by web proxy logs and directed to over 50 search engines. They make up a slight increase in average query term length, resulting in about three terms. A study by *Chitika Insights*[2] from January 2012 considers the most popular search engines in the United States (Google, Bing, Yahoo, Ask, AOL) and constitutes an increase to 4-5 words per query over the years before. With that, the query still is relatively short (e.g. compared to whole sentences). Adding extra terms to a short query may increase the matching between query and documents, implicitly disambiguate the terms within the query or move the search towards the most popular or representative meaning of the query. With the query becoming slightly more verbose, there will be less confusion about pinpointing different senses of a term within the query and contextual query expansion methods may become more accurate.

1.1 Information Retrieval

Information retrieval (IR) is a computer science research area with highly practical applications. It is concerned with the storage, search and retrieval of information. A widely recognized definition is provided by [Manning et al., 2008, p. 1]:

> *Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*

Usually dealing with digitally available texts makes it a sub area in the wide field of natural language processing (NLP). As the definition points out, a user with an information need serves as the initiation of the IR process. The user information need in textual IR is expressed in a so called *query*. The documents are computationally pre-processed and stored in efficient data structures. Given the query on the one hand and the documents on the other, the IR system has to perform a matching between them and return a list of ranked documents to the user. Figure 1.1 gives overview over this process [Broder, 2002], incorporating the basic concepts in IR. Optionally, the user can give some sort of feedback to the retrieval results. This could either include global methods such as query reformulation or spelling correction, or local methods such as *relevance feedback*. That is, after the system has returned an initial set of retrieval results, the user marks some returned documents as relevant or non-relevant. Based on the user feedback, the system then computes a better representation of the user's information need.

The returned documents may either be *relevant* to the query or they may be *non-relevant*. The goal of an IR system is to retrieve all the documents relevant to a user's query and keeping the retrieved non-relevant ones to a minimum [Baeza-Yates and Ribeiro-Neto, 1999]. Evaluation and assessing relevance will be explored in Chapter 4.3. The matching between the documents representation and the query representation is done by an IR model. In the following, some popular models and paradigms are described exemplarily.

1 http://www.hitwise.com/us/press-center/press/releases/2009/google-searches-oct-09/
2 http://chitika.com/insights/2011/ask-com-users-are-the-most-verbose-internet-searchers

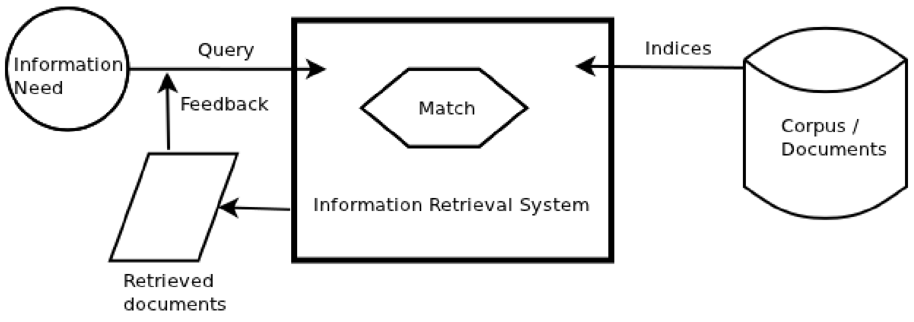


Figure 1.1: The classic model for IR (after [Broder, 2002, p. 2]).

1.1.1 Vector Space Model

One of the oldest and most well known methods for text similarity is the Vector Space Model (VSM) [Salton et al., 1975]. In the VSM documents are represented as vectors in a high-dimensional space. In that space each vector corresponds to a separate term and each vector entry is non-zero with the term occurring in the document. These entries are called term weights and there are several ways to compute them. One of the most known is the *tf-idf* term weighting scheme which is even widely used outside of the vector space model. The tf-idf weighting takes into account the term frequency (tf) and the inverse document frequency (idf) as explained in the following. The term frequency is defined as in Equation 1.1 and denotes the number of times a certain term *t* appears in a certain document *d*. To prevent a bias towards longer documents, the term count is usually normalized.

$$tf_d(t) = \text{term frequency of term t in document d} \tag{1.1}$$

In addition, the document frequency counts the number of documents the term t appears in and is defined as shown in 1.2.

$$df(t) = \text{document frequency of the term t in the entire corpus} \tag{1.2}$$

The inverse document frequency (idf) on the other hand measures the general importance of the term within the document collection. It is inversed, because terms occurring in many documents (e.g. determiners) tend not to be as meaningful as rare terms are [Luhn, 1958]. It is computed as show in Equation 1.3.

$$idf(t_j) = log(\frac{N}{df(t_j)}) \tag{1.3}$$

Since it is computed over the entire collection of documents (its size is denoted as *N*), it can be regarded as a global weighting of a term, while the term frequency by definition is a local (i.e., document based) weighting factor. Usually, this value is logarithmized to deal with small values. Putting the two parts together, we get the combined tf-idf weighting as shown in Equation 1.4.

$$w_{TF\text{-}IDF(t_j,d_i)} = tf_{di}(t) \cdot log(\frac{N}{df(t_j)}) \tag{1.4}$$

So, overall tf-idf defines a vector representation for all documents with a document d_i being represented as a vector $\vec{d_i}$ as exemplified in formulation 1.5.

$$\vec{d_i} = (w_{TF\text{-}IDF(t_1,d_i)}, w_{TF\text{-}IDF(t_2,d_i)}, w_{TF\text{-}IDF(t_3,d_i)}, ...) \tag{1.5}$$

Obviously, the VSM has some drawbacks. Since it relies on lexical overlapping, the terms must precisely match the documents. This is problematic especially for short texts. Furthermore, the word order is lost in the representation and it assumes that the terms are statistically independent. This is exactly where other methods are applied, using lexical-semantic resources.

1.1.2 BM25

Due to the vagueness and ambiguity of language within the documents and the query, there is no guarantee that a retrieval document is relevant to a query. Probabilistic approaches now model relevance probabilities. There are several probabilistic retrieval models in information retrieval. One of the more successful systems was developed for the Okapi system by the London City University [Robertson et al., 1996]. They tried different weighting models prefixed with *BM* (short for *Best-match weighting function implemented in Okapi*). The BM25 setting turned out to be best and now represents state-of-the-art retrieval functions used in document retrieval, such as in web search engines. BM25 is the name of the ranking function, but it is usually referred to as *Okapi BM25*. Since BM25 and one of its variations are used throughout this thesis, we will now explain it. Following the notations of Section 1.1.1, BM25 uses the length l of a document d, as defined as follows.

$$l(d) = \text{number of words in document d} \tag{1.6}$$

Additionally, l_{avg} is the average length over all documents of the collection. Two more parameters are k_1 and b with $k_1 \geq 0$ and $b \in [0,1]$. The entire retrieval function is all about weighting the terms and can be seen in Equation 1.7 [Gottron, 2010]. The sum iterates over all terms appearing in both the document and the query.

$$\rho_{BM25}(d,q) = \sum_{i \in T(q) \cap T(d)} log(\frac{N}{df(t_j)}) \cdot \frac{(k_1+1) \cdot tf_d(t_i)}{k_1 \cdot \left((1-b) + b \cdot (\frac{l(d)}{l_{avg}})\right) + tf_d(t_i)} \tag{1.7}$$

The first part is the global weight as introduced in the vector space model (see Definition 1.3). The second factor combines the term frequencies with the document lengths, with k_1 controlling the impact of the term frequencies. Setting $k_1 = 0$ results in them not having any influence at all. The parameter b however is the normalization factor of the weighting in relation to the document length. According to [Manning et al., 2008, p. 233], experimental best-practice results were setting k_1 between 1.2 and 2, and $b = 0.75$.

1.1.3 Divergence From Randomness

The Divergence from Randomness (DFR) paradigm [Amati, 2003] is based on the hypothesis that the level of treatment of the words considered to be informative is witnessed by a small set of documents (the "elite set"), in which these words occur much more often than in the rest of the documents. Other terms not possessing an elite set are assumed to follow a random frequency distribution. The DFR models are based on the idea, that the more the frequency of a term within a document diverges from its frequency within the document collection, the more informative the word t is in the document d. As formalized in Equation 1.8, the weight of a term t is inversely proportional to the probability of its term frequency within the document d obtained by a model of randomness M.

$$weight(t|d) \propto -log\ Prob_M(t \in d|Collection) \tag{1.8}$$

Whereat $Prob_M$ is a basic IR randomness model, which is the first out of three steps of obtaining a DRF model: Selecting a basic randomness model (e.g. BM25), applying the first normalization and normalizing the term frequencies. The first normalization simulates a *risk* component in the DFR models. If a rare term (in the collection) has a high frequency in a document, then it is almost certain to be informative for the content described by the document. That would be reflected by formula 1.8 in such in case. In return, such a minimal risk may also reflect in a minimal information gain. That is why a smoothing factor P_{risk} is applied to the weight. The amount of information gained with the term is shown in 1.9.

$$gain(t|d) = P_{risk} \cdot (-log\ Prob_M(t \in d|Collection)) \tag{1.9}$$

The smaller the probability P_{risk}, the more frequent the term t is in the elite set of its documents, that is, the more the terms frequency diverges from randomness.

$$P_{risk} = 1 - Prob(t \in d|d \in Elite\ set) \tag{1.10}$$

For computing the information gain of a term within a document, the search engine framework used in this thesis (see Chapter 3) uses two models, the Laplace succession model and a binomial model. Thirdly, the term frequencies are normalized. Before using the generating formula (such as in 1.9), the document length dl is normalized to a standard length sl and the term frequencies tf are also recomputed with regard to the standard document length. The normalized term frequencies result as shown in formula 1.11.

$$tfn = tf \cdot log(1 + \frac{sl}{dl}) \tag{1.11}$$

Since version 2, Terrier also provides a more flexible formula called *Normalisation2*, enabling a hyper-parameter factor value in front of the fraction. Overall, Terrier includes many DFR models, among them Bernoulli-Einstein, Inverse Term Frequency and a Poisson model.

1.2 Natural Language Processing and Statistical Semantics

Since this work is an application of the computer science research field of *natural language processing* (NLP) with parts of the *statistical semantics* domain, both areas are introduced in this section.

1.2.1 Natural Language Processing

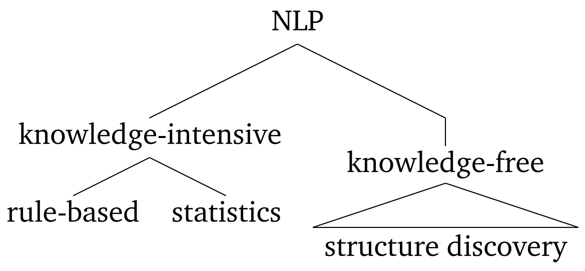
Natural (human) language is a result of the co-evolution between the innate facility for language that is possessed by the human intellect and of the development of the language itself. Compared to vision and speech however, which merely requires the appropriate physical configuration, reading and writing language is a rather unnatural process that has to be learned by any human being. In contrast, the idea behind Natural Language Processing (NLP) is the ability of computers to process and understand human language [Jurafsky and Martin, 2009]. Since the research about language is called *linguistics* and NLP presupposes the use of computers, it is also called *computer linguistics*, indicating the employment of linguistic methods and approaches. Although, NLP is a broader term, including other related computer science fields such as artificial intelligence and machine learning. There are many ways of language being available (speech or written, analogue or digital), but NLP usually focuses on processing digitally available texts. To process language or even derive meaning from it, many intermediate steps are necessary. Starting from “low-level” tasks like sentence boundary detection (to distinguish sentences) and tokenization (splitting the sentence strings into single words) to lemmatization (identifying the lemma form of a token), part-of-speech assignment and shallow parsing to identify phrases from the part-of-speech tagged tokens [Nadkarni et al., 2011]. Higher-level tasks build on the low-level ones and go on with spelling and grammatical error identification, named entity recognition (NER), word sense disambiguation (WSD) or relation extraction. On top of these building blocks, more complex NLP applications can be built. Areas of NLP application include, but are not limited to:

- automatic summarization
- automated essay scoring
- machine translation (MT)
- information retrieval (IR)
- computer-assisted language learning
- sentiment analysis

A software framework for natural language processing and chaining NLP tasks is depicted in Section 3.1. A visual example is given, too. After [Jurafsky and Martin, 2009], the following linguistic analysis levels can be distinguished in language understanding. The levels are given in order of implementation and build upon the previous one.

1. Phonetics and phonology
   Phonetics deals with the physical production, transmission and perception of speech sounds, phonology describes the way sounds encode meaning within or across languages [Carr, 1999].
2. Segmentation
   Segmentation is concerned with segmenting input streams (given as strings) in sentences, words and tokens (tokenization). Problems of tokenization are punctuation ambiguities and languages without spaces between words (e.g. Chinese).
3. Morphology
   In morphology, morphemes are identified. A morpheme is the smallest meaning-bearing unit in a language, e.g. *unbreakable* is composed of three morphemes: un- (meaning “not”), -break- (the root), and -able (signifying “can be done”).
4. Syntax
   Syntax is “the study of the regularities and constraints of word order and phrase structure” [Manning and Schütze, 1999, p. 99], that is how the words are arranged together to form sentences. A description of the valid structures in a language is called a *grammar*, by which sentences can be *grammatical* or *ungrammatical*.
5. Semantics
   Semantics is the study of the meaning of words, phrases or sentences, that is their denotation. Ambiguity (a word having multiple meanings) is a common problem in semantics.
6. Pragmatics and discourse
   Pragmatics deals with the purpose (intention) of an utterance, while discourse considers coherent groups of sentences.

However, the approaches to NLP can be quite different. One way of categorizing NLP by the intensity of knowledge used (e.g. in terms of lexical-semantic resources) is given as follows (based on [Biemann, 2012a, p. 7]).



Traditional NLP incorporates explicit knowledge (how to process language) and implicit knowledge (how to solve a task). Sources of knowledge can be lexical-semantic resources, machine-readable dictionaries, encyclopedias, Wikis, (hand-written) rules or annotations. Rule-based NLP tries to implement and apply linguistic theories by computational means. Those rules are derived by manual human work, not by data analysis. However, the vastly large size of available natural language gave rise to the field of statistical NLP, using statistics and machine learning approaches. The structure discovery paradigm works on “raw” texts and completely knowledge-free. While statistical and rule-based NLP may work supervised (annotated training examples are used) or not, structure discovery always works unsupervised. Statistical NLP can include any NLP task, such as syntactic parsing, but deriving meaning from text is associated with *statistical semantics*. In practice however, both approaches often are combined to achieve better results.

1.2.2 Statistical semantics

Statistical semantics as a sub area of statistical NLP is the “statistical study of meanings of words and their frequency and order of recurrence” [Delavenay, 1960, p. 133]. The term was first mentioned by [Weaver, 1955], arguing that WSD for MT should be based on co-occurrence frequencies of the target words context words. [Firth, 1957] supports the underlying assumption that “a word is characterized by the company it keeps”. An early contribution to the field was done by [Furnas et al., 1983], in which he had people describing objects and (due to low correlation between them) for each word a system to make one or more guesses as to what the user meant. One statistical semantics application is measuring the similarity in word meanings, as applied in the distributional resource in Chapter 1.3.1. In particular, this is called *distributional semantics* and is illustrated there.

1.3 Lexical Expansion

In this chapter, the three lexical expansion resources are introduced. It is explained how they are designed and how they work to provide lexical expansions. In this section, only the theoretical aspects will be explained.

1.3.1 Distributional Thesaurus

A thesaurus is a scientific resource that lists words grouped together according to their similarity of meaning. Usually, thesauri are put together manually by human knowledge. Since this is costly and time-consuming, it leads to extensive thesauri not being available for specific domains or small languages. [Lin, 1998] proposed a mechanism to build a thesaurus automatically. It is called *distributional* due to the fact that the words it contains have a similar distribution over contexts. The *distributional hypothesis* states that two words are similar if they appear in similar contexts [Harris, 1954]. This allows computing a distributional thesaurus automatically. After [Lin, 1998], one context for words that can be extracted within a sentence are so called *dependency triples*, that is the syntactic relationships within the sentence. For example in the sentence “I have a brown dog.”, triples are *(have subj I)*, *(I subj-of have)*, *(dog obj-of have)*, and so on. The similarities are assumed to be the more correct, the more contexts are taken into account. When applying large corpora, it is a natural step to scale the computation to a parallel architecture (i.e., to make use of distributed computing). That was done by [Biemann and Riedl, 2013] and me. We used Apache Hadoop as a software framework for distributed applications, since it supports data-intensive computing on large clusters and implements Google’s MapReduce [Lin and Dyer, 2010]. MapReduce is a programming paradigm for efficient distributed computing, operating on $< key, value >$ pairs exclusively and working on the *map* (applying a given function to each element of a list) and *reduce* (combining operation) functions known from functional programming languages. Figure 1.2 illustrates

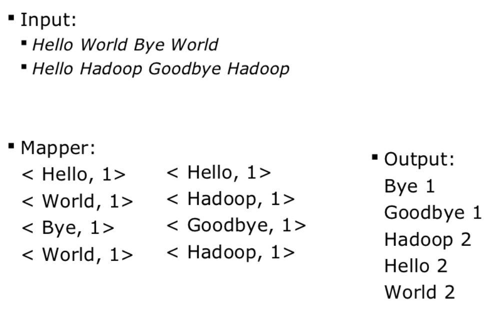


Figure 1.2: MapReduce word count example

this process with an example of counting the words in two sentences. While the mappers output a one (for one sighting) for each word, they are summed up by word in the reducing step. To build a Distributional Thesaurus with a series of MapReduce steps, [Biemann and Riedl, 2013] explain the following steps.

Context extraction

The first important stage is to extract a context (in which words co-occur) out of the raw text. We've implemented different versions of this step. The most successful one is done by using the Stanford Parser [Marneffe et al., 2006] to regard the grammatical relations of the words within the sentence. Remember that we try to apply the distributional hypothesis by this step. The dependency parser from the Stanford NLP group can look into all relations of the words within the sentence and indicate the relation name, the governor and the dependent. For example in the string ”The cat”, ”the” is the governor, ”cat” the dependent and ”det” (for *determiner*) is the relation name. We output the dependent as the word and the governor, together with the relation, as the *feature* of the word. The final result of the first steps then looks like depicted in extracts in listing 1.1.

Listing 1.1: DT: context extraction

```
politics  the#det         1276
politics  american#amod   379
politics  playing#-dobj   144
politics  national#amod   143
politics  local#amod      141
politics  party#nn        135
politics  his#poss        124
```

That is, that the word ”politics” has ”the” as determiner or ”national” as adjectival modifier as features[3]. Additionally, the occurrences of the word feature pairs are counted.

Frequency and significance

By virtue of optimizing and pruning the data, we like to indicate a significance to each of those word feature pairs. Therefore, in preparation to this step the words and the features by itself are counted. We then continue to assign a significance value to each word-feature pair. This is accomplished according to the Lexicographer's Mutual Information (LMI) measure as in [Bordag, 2007, p. 77]. For each pair, the frequency of the word, the feature and the pair itself is included. The output of this step looks like depicted in extracts in listing 1.2.

[3] A complete list of the Stanford Parser relation names can be found here: `http://nlp.stanford.edu/software/dependencies_manual.pdf`

Listing 1.2: DT: frequency and significance

```
politics  american#amod         1556.9131
politics  involved#-prep_in     655.3890
politics  national#amod         440.7464
politics  british#amod          398.1417
politics  local#amod            374.0130
politics  interested#-prep_in   294.8649
politics  business#conj_and     204.2767
```

This is followed by a pruning step with experimentally obtained parameters. The format doesn't change, the data is just reduced.

Aggregate per feature

To come closer to the similarity of two words, we first want to cumulate all the words that share the same feature(s). To do so, this MapReduce step aggregates all the words that have a certain feature. The result then looks like depicted in extracts in listing 1.3.

Listing 1.3: DT: frequency and significance

```
caring#amod       volunteers everyone person ...
caring#appos      superdome
caring#ccomp      feels
caring#conj_and   feeding trustworthiness ...
caring#dep        person
caring#dobj       stopped
caring#pcomp      between
```

Read it this way: ”caring” occurs as an adjectival modifier for ”volunteers”, ”person” etc. (Note that the list of shared features can become pretty long. That's why some very simple examples are chosen here.) They all share a feature (in this case ”caring#amod”). This is a first clue for similarity: If two words share a feature, they have something in common.

Similarity

Finally, the last step of the MapReduce chain is calculating the similarity. For every pair of two words we want to assign a weighting (representing their similarity). The first simple approach is to ask: How many features do two words have in common? The more they do, the more similar are they.

To answer this question, each list is processed quadratically whereby a one (for one sighting) is emitted. We implemented and tested multiple possibilities, including the word list length and logarithmic normalization. But emitting the simple one (in combination with parameters from other steps) turned out to achieve the best results. (Additionally, the list had to be cut to avoid data explosion to a point beyond what even parallel architectures can handle.) So here's what the final similarity lists look like depicted in extracts in listing 1.4.

Listing 1.4: DT: similarity count

```
politics  politics      590.0
politics  government    36.0
politics  journalism    36.0
politics  affairs       35.0
politics  economics     33.0
politics  politicians   32.0
politics  philosophy    31.0
```

By reason of comparison the arrangement with itself is kept in the pipeline. The number declaring the similarity is exactly the number of features the words share.

The distributional thesaurus contains such lists of similar words for each word enclosed in it.

Variants

Since each step is replaceable other contexts can be applied (provided that they comply with the output format). The first context we were experimenting with is the immediate vicinity of words. That is, that both the left and right neighbour of a word is a feature. This is a context (and hence a DT variant) I used in the experiments. It is named RL, short for *right-left* context.

For the dependency parser context there are variants of the distributional thesaurus incorporating the part-of-speech (POS) tag for each word. Furthermore there are variants carrying along the features the words have in common. This allows answering the question, why they are similar.

1.3.2 TWSI Sense Substituter

As presented by [Biemann, 2012b], the Turk Bootstrap Word Sense Inventory (TWSI) is a lexical resource created by a crowdsourcing process. Crowdsourcing is a portmanteau word[4] of *crowd* and *outsourcing*. For TWSI, Amazon Mechanical Turk (MTurk) was used. The workers were given a sentence at which a target word is marked. Their first task was to come up with substitutions for the given word in its context. Please find an example in the following listing (the target word is *capital*).

> The company manages over 10 million dollars in capital assets.

The development of TWSI was driven by substitutability and cites lexical substitution tasks and semantic search as example application scenarios. It supplies lexical substitutions for a given target word in its sentence-wide context. Lexical substitutions are provided for 1012 highly frequent English common nouns in Wikipedia. The 1012 nouns are grouped into 2443 senses with one target having 2,41 senses on average. Here is an example output for the sentence above (containing a polysemous target term):

> **The company manages over 10 million dollars in** <target="**capital**" lemma="capital" sense="capital@@3" confidence="0.98863536" substitutions="[money,27][asset,14][fund,14][investment,14][cash,13] [financial asset,10][finance,8][stock,8][wealth,6][property,5]"> **assets.**

The target retrieved substitutions with respect to its sense, the numbers denote how many people supplied the substitution for the given target word in that context. But in addition to being a lexical resource for substitution, TWSI at the same time built a sense inventory. The very fine-grained sense structures of resources like WordNet are considered a hampering issue in practice, that is the reason for a rebuild of a sense inventory from scratch. So, TWSI draws those sense distinctions according to common substitutions. That is meant by *driven by substitutability*: The sense distinctions are formed by the target words substitutions overlap. If two usages have the same or very similar substitution lists, they probably belong to the same sense. In the previous listing, the target term *capital* was used in sense number three (*capital@@3*). In the resulting sense inventory of TWSI, a sense is characterized by a

[4] That is a new word combination artificially created out of two words, e.g. *brunch* (*breakfast* and *lunch*).

gloss[5] and a list of substitutions. With an average of 2,41 senses per target term and a declining sense distribution, the majority of the targets are used in one or two senses. The target *capital* is one of the few terms with actually five senses (e.g. as in *capital letters*, *capital of Chedi*, *capital punishment* and *social capital*). The full five senses can be viewed in the following listing.

capital@@1: *The city called Suktimati is mentioned as the capital of Chedi .* (capital++34554464)
capital@@2: *This is because they build social capital , trust and shared values , which are transferred into the political sphere and help to hold society together , facilitating an understanding of the interconnectedness of society and interests within it .* (capital++50057722)
capital@@3: *Member Economic Participation : All members democratically control the capital of their cooperative .* (capital++1165690)
capital@@4: *S & P and other rating agencies have slightly different systems using capital letters and + - qualifiers .* (capital++15646388)
capital@@5: *He has attracted international attention through a disturbing video music theatre work , Those Who Speak In A Faint Voice , a project about capital punishment , and later through the multimedia music theatre projects CREDO and WINNERS .* (capital++19374025)

To make that clear, what [Biemann, 2012b] describe is twofold:

Lexical Resource
The Turk Bootstrap Word Sense Inventory was presented as a large electronic resource containing a collection of sense annotations.

Lexical Substitution System
Furthermore, a lexical substitution system trained on that resource is described. It uses a supervised word sense disambiguation techniques to provide lexical substitutions in context.

Applying the substitution system, the following steps are performed:

1. The text is split into tokens.
2. Substitutions are produced for all tokens (of a predefined list).
   a) For monosemous target words, the list of substitutions is provided.
   b) For polysemous words, the feature representation is computed and classified.
      i. The classifier returns a sense label classification and a confidence score.
      ii. Substitutions are retrieved for that sense label.

1.3.3 Delex Substituter

As described by [Szarvas et al., 2013], the Delex Substituter is a supervised all-words lexical substitution system using delexicalized (i.e., non-lexical) features. Instead of using a separate classifier per word, a global model is trained on delexicalized features. Features from lexical resources (WordNet) are used as well as features from large corpora (e.g. n-gram counts). Following previous research results, they use

[5] In the WordNet notation, glosses are brief definitions of a sense and/or example sentences. In TWSI, it is just an example sentence.

WordNet as a source for synonyms (as candidate substitutions) and n-gram frequencies for ranking as their baseline.

First, they train their model on two freely available data sets. The first is the LexSub data set, which consists of 2002 sentences with native speaker annotations of paraphrases or substitutions for the word in context. The paraphrases are assigned the frequency of how many suggested the substitute. The second data set is the larger TWSI, which contains 24.647 sentences with substitutions for nouns collected through a crowdsourcing process.

In the paper they focus on producing better ranking models by using more advanced features. As possible paraphrases, candidate substitutions are extracted from WordNet. (WordNet is organized in *synsets*, i.e. synonym sets, a group of synonymous words or collocations like *car pool*. So if you look up a word in WordNet, you find it in synsets.) Therefore, all the synonyms from all the word's synsets are taken as candidate synonyms, as well as the ones from the synsets being in a *similar to*, *entailment* or *also see* relation to one of these synsets. Then they used a Maximum Entropy (MaxEnt) classifier model from the Mallet package and trained a binary classifier to judge the validity of a given substitution in a particular context. In a lexical substitution task, *generation* and *ranking* of substitutions are the two prominent tasks. Here is the overview of that structure in the Delex system:

Generation
: The generation of substitution candidates is not provided, instead synonyms are extracted from WordNet as candidates for the substitution task.

Ranking
: The goal is to construct better ranking methods based on supervised machine learning and an advanced selection of features (delexcicalized features).

The ranking is based on various features, which are described as follows.

Lexical Resource Features
: As a source for lexical resource features, WordNet 3.0 is used as well. In particular, it is used how many senses the target word and all the substitutions have. And since the sense numbers are ordered by corpus frequency, it is informative from which sense number of the target word the substitution candidate came from.

Corpus-based Features
: Features from several steps of the Distributional Thesaurus' (DT) creation process (cf. Section 1.3.1) are used. For the DT, the source sentences are parsed with the Stanford Parser to extract features[6] for words. From the syntactic dependencies within the sentence a dependency triple $(w1, r, w2)$ is formed denoting the relation r between the two words $w1$ and $w2$. For characterizing $w1$, $(r, w2)$ serves as a feature and $(w1, r)$ for $w2$ respectively. After a significance and pruning step to reduce the data, only the 1000 most salient features per word are kept. From that intermediate result, the first ML feature is extracted: The percentage of the shared salient features among the top k entries for various k. In addition, a measure is applied on the substitutions (from the substitution candidates) features lists (from the DT) to determine to what extent the context (as elements from those features list) characterizes the substitution. Similarly to the features overlap, the percentage of shared words among the top k similar words for various k is compared. The more similar words the target word and the substitution candidate share (especially for lower k), the more they might be synonymous. Lastly, the Boolean feature is applied whether the substitution candidate is in the target words top 100 similar words list or not.

[6] In this sense, *feature* is not used as a machine learning (ML) feature, but rather in the sense of a property, an attribute or a characteristic.

Local n-gram Features

To assess the syntagmatic coherence of the substitutions in context, n-gram frequencies[7] are used in the best Semeval 2007 system. [Szarvas et al., 2013] use it as their baseline as well and use the same n-grams a features in their supervised model. They use 1-5-gram frequencies in a sliding window around the target word, once normalized with regard to the target word and once to the substitution candidates set. Additionally, the 3-5-gram frequencies are used of how often the target and a candidate are part of a comma-separated list or conjunctive phrase with *and* or *or*.

Shallow Syntactic Features

For the model to learn POS-specific patterns, part-of-speech information is also used (from the TreeTagger). Specifically, 1-3-grams of the main POS categories are used in a window around the target word. For the LexSub data set containing targets from all major part-of-speech, that is particularly useful.

As an example of the delexicalized features, the following sentence is illustrated (with the target word underlined):

He was <u>bright</u> and independent and proud.

Using WordNet as the source for substitutions candidates results in 75 potential paraphrases (11 from the synsets of *bright* and 64 from related synsets). The human annotators listed *intelligent* and *clever* as suitable substitutions, from which just one (*intelligent*) is found in the WordNet candidate set (i.e., *clever* was not found in WordNet). The remaining candidate *intelligent* was characterized by 178 active features in that context, which are composed of the following items:

- 112 features based on n-gram features (different n, variations and normalizations)
- 48 active distributional features (different POS and normalizations)
- 12 shallow syntactic features (contextual POS patterns)
- 6 resource based features (number of senses of target and candidates)

The Delex systems received its best results on the LexSub data set outperforming the baseline significantly. In a feature exploration done in their paper, all their features contribution is complementary. Still, leaving out the n-gram features results in the biggest negative effect.

[7] The n-grams are extracted from the Google Web1T corpus, which contains up to 5-grams collected from one trillion sentences.

2 Related work

There has been much work on word sense disambiguation systems in IR trying to disambiguate a terms sense before expanding. That's why we want to divide the related work into three parts. The first part covers the contribution of Word Sense Disambiguation (WSD) to IR. The second one is about query expansion, which is one fragment of the more general concept of lexical expansion. That is what we will be looking at in the third section. The last section subsumes all other related work not fitting into any of the previous categories. Alongside, the related work will illustrate further concepts associated with IR systems.

2.1 Word Sense Disambiguation

'When I use a word,' Humpty Dumpty said, in rather a scornful tone, 'it means just what I choose it to mean — neither more nor less.'

(Through the Looking-Glass, Lewis Carroll)

Word Sense Disambiguation (WSD) is the task of computationally determining the correct meaning (represented and distinguished by a particular *sense*) of a word in its context, see [Navigli, 2009] for a survey. If a computer processes natural language, ambiguity becomes a problems. WSD is a well-known approach to the lexical mismatch problem is IR [Krovetz and Croft, 1992]. There is much related work for that, but conflicting conclusions about the contribution of WSD system to Information Retrieval performance. First of all, [Sanderson, 1994] points out the belief that if all the ambiguous words used in IR texts can be disambiguated correctly, IR performance will increase in the same way. The completely automatic technique of *pseudo-words* was used to introduce sense ambiguity into and evaluate on a collection and to control its degree of quality. Pseudo-words (loosely based on [Yarowsky, 1993]) are artificial terms created by the combination of two words (e.g. *banana* and *kalashnikov* become *banana/kalashnikov*.) Since knowing the correct sense at each occurrence, evaluation becomes trivial. They reason that word sense ambiguity only is problematic to a probabilistic IR system when retrieving from very short queries. As they go on, they conclude that the disambiguation system has to have a high degree of accuracy to be of any use to an IR system. [Sanderson, 2000] goes on to outline the previous research in word sense disambiguation and presents attempts explicitly using disambiguation in IR. Lastly, a review of the efforts in information retrieval without identifying senses is given. First, the overview of disambiguation research is divided into two basic classes on which disambiguation can be based on: Manually created rules and evidence from large corpora. Manually generated rules can either be general context rules (disambiguate a sense in a certain words context), template rules (word appears in a specific location), using the grammatical category (e.g. *the train* or *to train*) or having "word experts" (which are programs) for each ambiguous word. All the approaches however require tremendous effort, causing [Kelly and Stone, 1975] to state that "... such a strategy cannot succeed on a broad scale." Using existing corpora on the other hand include textual definitions of a dictionary, whereas the sense definitions are regarded as a small collection of documents and the ambiguous words as queries to perform a ranked retrieval on co-occurring words or words overlap. Further applications include extending the dictionary definitions with commonly co-occurring words or disambiguating the whole sentence simultaneously. Another approach is to manually part-of-speech tag a corpus and train a statistical classifier to learn disambiguation features. However, the machine learning method did improve much against the already low baseline (always select the most common sense). Whereas those sense were taken from WordNet, a number of researchers used thesauri (e.g. Roget's thesaurus) in disambiguation research. Based on the semantic categories provided by those resources, they attempted to resolve an ambiguous word into one of those categories based on a set of clue words. Although the limited set of senses resulted in generally coarser senses, a high accuracy in a stand-alone WSD tasks could be achieved, but did not greatly improve IR performance. [Sanderson, 2000] summarizes a relatively small impact of ambiguity on retrieval effectiveness and terms the skewed distribution of senses of many words along with word collocation effects to account for that. Furthermore, all attempts at automatic disambiguation have failed to improve IR effectiveness.

The practical part of this thesis is motivated by the experiments of [Wolf et al., 2010]. First, they tried to increase the precision of WSD by combining the annotations of two WSD systems. Second, they intended to combine the widely used *DFR_BM25* probabilistic IR model with a monolingual translation-based model (trained on different lexical-semantic resources). Although their systems with and without word senses performed best in the monolingual RobustWSD track at CLEF 2009, not any improvements of utilizing the word sense annotations could be observed. Yet, they were successful at the second goal by showing that the combination approach always achieves better performance than the stand-alone models. [Zhong and Ng, 2012] on the other hand experimentally demonstrated significant improvements of WSD to IR. They could show a performance improvement of a supervised WSD method compared to two WSD baseline systems. Their proposed method annotates senses to terms in short queries. Therefore, a method for estimating the distribution of senses for short queries is proposed. Together with the senses predicted for words in documents, the senses and synonym relations are incorporated into the language modeling approach to IR. Evaluated on four TREC query collections, their utilization of the supervised WSD tagged word senses led to significant improvements over a state-of-the-art IR system. Lastly, as a summarization of their related work they also point out that errors in WSD (erroneous disambiguations) can easily neutralize its positive effects. This coincides with the observations gained by [Sanderson, 1994, p. 149] and [Sanderson, 2000, p. 11].

[Na and Ng, 2011] also consider word disambiguity and the lexical gap as critical problems in information retrieval. Most work so far dealt with those problems by *monolingual approaches*, that is enriching and disambiguating the queries and documents in the original source language. The authors propose a novel approach in that matter by enriching the documents by a *translation representation* of the document. Each original document is automatically translated into an auxiliary language using a simplified form of machine translation (MT). The translated document then serves as an enhanced representation of the original bag of words. By connecting the original and translated words in a many-to-many relation, the ambiguity problem is supposed to be addressed during the process of selecting the correct translation in context. Additionally, by using the auxiliary language the word mismatch problem (of the source language) is extenuated by mapping to the translated words. Instead of full-fledged MT, *monotonic translation* is applied, that is the word order of the source language does not change after translation. One part of the simplified translation method is the estimation of the *expected term frequencies* for words of the translated documents, taking the average term frequency over all possible translated documents. After producing the multilingual representations, multiple evidences of them are combined and the relevance score of a document is calculated. On standard TREC collections, the approach showed significant improvements by using English-to-Chinese translations over baseline monolingual retrieval methods.

2.2 Query expansion

The idea behind query expansion is the fact that the user of an information retrieval system has to express his information need into a textual representation of an actual query. Doing so, a great deal of information could be lost during that translation. By enhancing the query terms entered, it is aimed to address the underlying information need more accurately. There has been lots of work on query expansion in the last years. As [Carpineto and Romano, 2012] give an overview over automatic query expansion (AQE), applying various data sources, different principles and techniques. They also emphasize the importance of AQE for search effectiveness, followed by a design and implementation analysis of AQE components, concluding by a comparison of state-of-the-art and reasoning about future work. However, giving of review of recent techniques of AQE was not the only objective of their work, but also to assess the performance limitations of those techniques. For though AQE is seen as an extremely promising approach, achieving very positive experimental findings in laboratory settings, it is not yet employed in operational Web IR systems such as search engines. The reasons for that are the fast response times search engines are required to provide, the technique's instability and the emphasis of AQE to improve recall, which is less important to the users as they only look at the first page of results. They authors first point out, that with the query being expanded there is more chance for relevant documents not containing the original terms. That results in an obvious increase in recall, which is important for many professional applications (legal, medical, scientific). But improper expansion could also cause a topic drift, causing a loss of precision. The suggested taxonomy for AQE divides all the approaches into five main groups: linguistic methods, corpus-specific statistical approaches, query-specific statistical approaches, search log analysis and Web data. Linguistic analysis methods aim at providing additional linguistic information to the original query. That can either be by lexical or semantic word relationships (based on knowledge representation sources) or ontology browsing. But also a syntactic analysis between the query terms is an approach for linguistic information, e.g. learning from the relation paths induced by a parse tree. Corpus-specific global techniques are based on frequencies in the collection, two examples are concept terms and term clustering. Whereas query-specific local techniques try to take advantage of the local context provided by the query. Common methods are the analysis of feature distribution difference and model-based AQE. The idea behind analysing search logs on the other hand is to mine query associations already made by the user (i.e., extracting associated terms from past queries). In addition to those implicit measures thought to be any relatively accurate, the availability of large-scale search logs is an issue. Lastly, the approach of using web data is to exploit anchor texts as a source for AQE. Since an anchor text is a succinct description of the destination page, the authors consider them to be similar to user search queries. Finding appropriate anchor texts for a query is done by ranking the text similarity. However, that is not supposed to work well on short queries. Other types of web data can be employed, including Wikipedia documents and hyperlinks, or FAQs. Regarding computational efficiency, the authors point out that there are additional costs of generating the expansion features and costs of evaluating the expanded query. AQE runs are found to be much slower than those with the original queries, because the execution time linearly depends on the number of query terms. The cost of generating expansion features however does not affect runtime, because they usually are generated in advance and what is left at query time is just the selection of those features. In summary, the authors consider linguistic techniques less effective compared to those based on statistical analysis, because the latter ones require exact word sense disambiguation. Even if statistical analysis can not always be applied (e.g. due to the absence of co-occurrence frequencies with good expansion terms). Furthermore, local analysis seems to perform better than global corpus analysis due to the extracted features being more query-specific. The query-specific techniques however need a double run at query time. But all in all, research caters towards different requirements and hybrid methods to achieve best results, since there is no single silver bullet to solve all aspects of the vocabulary problem in IR. In general though, the advances of AQE have been proven in benchmarks and experimental tests, with gains not only in recall, but also in precision. What AQE still suffers most to become a standard component in search systems is its robustness of retrieval performance and its usability in IR systems implementing AQE.

[Metzler and Croft, 2007] proposed a query expansion technique based on the Markov random field (MRF) model, called *latent concept expansion* (LCE), aiming at combining term dependence with query expansion. Given the original query it aims at discovering the *latent concept*, that is the concept the user has in mind while expressing that query. Those can consist of single or multiple terms or some combination. This is done by defining the MRF by a graph G and a set of potential functions over the cliques in G. Then, the expanded graph H (with e.g. single term concepts) is constructed and a probability distribution over latent concepts is computed (approximated). To use this for query expansion, the k latent concepts with the highest likelihood are selected and finally the documents are ranked according to the top k expansion concepts. Applying single term concepts the authors showed significant improvements over relevance models across all five TREC data sets tested, expansion with multi-term concepts only led to negligible increases in mean average precision. Still, the authors suggest other tasks those concepts may be useful for, such as query suggestion/reformulation, summarization and concept mining.

2.3 Lexical expansion

As explained in section 1.3.1, [Biemann and Riedl, 2013] produced a high-quality and knowledge-free lexical resource in an unsupervised way. First, they introduce their new metaphor of two-dimensional text, outlining two major shortcomings of prevalent vector space representation of text: The first is the high dimensionality, i.e., size issues have to be handled with “necessarily lossy dimensionality reduction techniques”. Second, that vector space models are not generative, that is the model presupposes a list of alternatives to rank, instead of originating them from the model itself. the After, they carry out the development from distributional to contextual similarity. The evaluation of the Distributional Thesaurus (DT) is performed against WordNet. Additionally, the approach is evaluated on the official test data of the Lexical Substitution Task 2007 data set (LexSub). The LexSub data set consists of 2010 sentences with a total of 201 target words (i.e., ten sentences for each word). For each target in its sentence context, five native speaker annotators were asked to provide substitutions or paraphrases for the target in its context. It is then calculated how substitutions have been detected correctly out of ten guesses (out of ten precision). Using the DT performance as a baseline to evaluate the contextualization, the Contextual Thesaurus (CT) outperforms it in most cases. However, compared to the original lexical substitution task at Semeval 2007 [McCarthy and Navigli, 2009], they could not come close to the best-performing participating systems in any configuration. Still, their framework for lexical expansion did not use any (manually composed) lexical resources, but just a large amount of unlabeled text. [Miller et al., 2012] used that two-dimensional method and the lexical expansions from the DT as well in want to explore its contribution to purely knowledge-based word sense disambiguation. In their experiments, they explore the contribution of lexical expansion to the simplified and simplified extended variants of the Lesk algorithm. Using a new method based on word overlap between sense descriptions and the target word context, such an approach inherently suffers from the lexical gap problem, since the sense descriptions and the context might not have much vocabulary in common. Enriching the textual information (definitions are provided by WordNet) with expansions produced by the DT, they were able to successfully bridge that word mismatch by increasing the number of overlapping word pairs. Employing this resource led to a significant improvement, beating even the best-performing system of the SemEval-2007 all-words disambiguation task in both the fine-grained and coarse-grained evaluation. All in all, the authors reason about the concept of lexical expansions as a “promising avenue”. Additionally, [Bär et al., 2012] applied the DT (among other resources) for text similarity and were able to come up with the best participating system in the Semantic Textual Similarity (STS) task at SemEval-2012 in two out of three metrics by combining several text similarity measures, both simple string-based and semantic ones. The goal of the STS task is to measure the semantic similarity between pairs of sentences, with graded notions of similarity. First, various text similarity measures are outlined, from string-based and semantic similarity ones to text expansion mechanisms. It is stated, that using them in separation exhibits a major limitation, since no single measure could capture all text characteristics necessary for computing similarity. Thus, multiple text similarity measures of all kinds are combined using a supervised machine learning approach. Among the features selected is the distributional resource, although only the feature based on cardinal numbers (CD) was selected in the final models. Moreover, the authors used the lexical substitution system by [Biemann, 2013] to provide substitutions to the text.

Recently, [St. Charles, 2012] provided an extensive study on various lexical expansion methodologies in the medical domain. The goal was to improve the MiPACQ (Multi-source Integrated Platform for Answering Clinical Questions) system’s retrieval performance, evaluated using the Medpedia corpus and the MiPACQ queries. Medpedia is a collaborative encyclopedia maintained by medical professionals and contains about 8.000 articles and 600.000 paragraphs (a document is split into 79 paragraphs on average). The MiPACQ however only re-ranks the top N documents returned by the baseline IR system. Since the baseline IR system used in MiPACQ had a poor performance in terms of *recall*, the re-ranker's overall performance again was limited. With the purpose of increasing recall, three term expansion approaches are explored. First, co-occurrence based expansion methods are examined, in particular automatic thesaurus generation based on co-occurrences. The author's idea behind using co-occurrences is that the more often two words co-occur, the more likely they are semantically related. Two terms co-occur if they appear in the same document or paragraph. A global thesaurus is then constructed out of these co-occurrences (which are weighted by a *tf-idf* weight) and used for automatic document expansion and query expansion. It is aimed to provided closely related terms *in the context of the corpus*. These methods universally improved recall (i.e., they returned more relevant documents than the baseline system), while causing insignificant drops in precision. Second, a resource based approach for query expansion using the UMLS (Unified Medical Language System) Metathesaurus as an external resource is explored. The UMLS Metathesaurus is a collection of medical ontologies and serves as the most complete thesaurus and ontology of medical terms. Additionally, it includes tools to map free text (from the queries) to its medical concept entities, returning additional terms for query expansion. The top N mappings were used and evaluated, whereas $N = 1$ exhibited the best retrieval performance. However, the best resource based methods still performed worse than the best co-occurrence based ones. Finally, latent semantic indexing (LSI) as an alternative to the baseline vector space retrieval model is evaluated, since the VSM relies on term overlap to determine document relevance and ranking. LSI is a patented indexing and retrieval method first mentioned by [Deerwester et al., 1990]. Its goal is to group terms together that are semantically related. This is accomplished by constructing a term-document matrix (1 if the term exists in the document, or else 0), similar to the vector space model (section 1.1.1). Then a mathematical technique called singular value decomposition (SVD) is applied to the (large) matrix, resulting in a smaller dimensional space (down to a few hundred dimensions). The grouping of semantically related terms into these semantic vectors is supposed to help with synonymy and polysemy. As with the term-document-matrix, the query needs to be transformed into the semantic space before being compare to the document representations in the latent semantic index. Once it is transformed, it can be compared to the documents using standard cosine similarity. The author conducted several retrieval experiments with different dimensions from the SVD decomposition, observing indices created with fewer ones performing worse than indices based on a higher dimensional space. In summary, he states that in addition to the more resource intensive and slower computation of LSI, even high dimensional indices result in a poor retrieval performance compared to the two other methods, especially co-occurrence based methods. In theory though, it shall be possible to create an index with enough dimensions to obtain precision gains without loosing too much of recall. In any case, such a system would be significantly slower. So, all in all, all the three expansion approaches examined by [St. Charles, 2012] could show an improved recall performance, gaining from 8% to 15% on average. The automatic thesaurus generation could gain up to 27% more recall, while giving in significant points in precision.

2.4 Other

Because it is not feasible to calculate, most probabilistic IR models assume some kind of independence between the terms occurring in queries and documents. Most work in the past on modeling term dependencies dealt with phrases/proximity or term co-occurrences and had poor and inconsistent results. According to [Metzler and Croft, 2005], this is caused by two reasons. The first one is the fact that most dependence models are based on the *BIM* (Binary Independence Model), in which documents are represented as binary vectors (presence/absence of a term) and terms are independently distributed among the documents. That is, term dependencies must be estimated from the (often very small) available set of relevant and non-relevant documents. Second, the document collections used for the experiments in the past predominantly consisted of short documents. Thus, modeling term dependencies with only little (co-)occurrences of terms is considered not sufficient. So, despite of using larger collections, [Metzler and Croft, 2005] incorporated several arbitrary text features (types of evidence). They developed a general framework for modeling term dependencies via Markov random fields (undirected graphical models). They explore three variants of the model: Full independence (query terms are independent), sequential dependence (certain dependencies between adjacent query terms) and full dependence (capture dependencies between every subset of query terms). As a result, modeling term dependencies could show a significantly improved retrieval effectiveness across a range of collections in both the dependency variants.

3 Software

In this chapter, we will describe the software setup that has been used and developed during this work.

3.1 UIMA

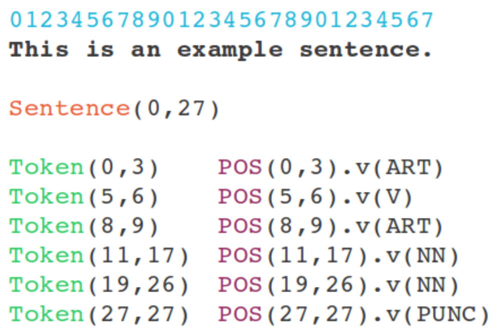
The underlying framework around all the components is UIMA, the Unstructured Information Management Architecture [Ferrucci et al., 2006]. Originally developed by IBM, it is now an Apache open source project and used in commercial as well as educational contexts. The single most known application using UIMA is the IBM Watson[1], an artificial intelligence computer system designed to take part in the quiz show Jeopardy! and answering questions posed in natural language. The following description of UIMA is partly based on the UIMA User Manual [UIMA Development Community, 2006] and the slides from the lecture *Natural Language Processing and the Web*[2].

The idea and motivation behind UIMA is the amount of unstructured information available. Information as data or knowledge comes in various forms (e.g. texts, images, tables, speech, databases) and from many different sources. Most textual information available on the World Wide Web, in books or newspapers in of unstructured nature. It is *unstructured* in the sense of not being machine-readable or automatically processable by a computer system. An unstructured information management (UIM) software is used to extract structured information and discover relevant knowledge with well-defined, computer-understandable semantics from unstructured sources. UIMA as a general UIM architecture wants to help the user to bridge that gap between unstructured and structured data and to build UIM applications with the ability to apply various related technologies as well (statistical and rule-based NLP, Information Retrieval, Machine learning, Knowledge Sources). To do so, the entire framework is based on the concept of annotations, which are created by the basic building blocks of UIMA, called Analysis Engines (AEs). An Analysis Engine (simply called *annotator*) receives the the unstructured text of a document and enriches it with structured information by performing annotations. Each annotator encapsulates a specific functionality (e.g. tokenization, sentence splitting, part-of-speech-tagging) and reads from and writes to the Common Analysis Structure (CAS), in which the document text is stored in. A CAS represents a document and serves as a container for storing the annotations. Next to the views (text, image, video, etc.) and the annotation index, the CAS also contains the type system, which forms the communication contract between the components. In addition to primitive types (e.g. integer, float, string), the type system contains some built-in complex types (e.g. arrays, lists), based on which the UIMA users can create their own annotation types with. The modularization provided by an annotator (encapsulating a specific functionality) is important for re-usability. But a too fine-grained modularization makes the whole set of annotators (called *pipeline*) complicated. The UIMA solution for this problem is the introduction of so called *Aggregate Analysis Engines* (AAEs). An AAE behaves like an atomic component, but combines several regular Analysis Engines. The annotations itself are based on their offsets in the text, covering an absolute begin and end position. In figure 3.1, three example types of annotations for a given sentence are visualized (sentence, token and POS). For the part-of-speech, its value instead of the covered text is given. Since the annotations are stored at a different layer and do not manipulate the original text in any way, multiple annotations at the same offsets or even offset overlaps do not pose a problem at all.

Although UIMA is available for Java and C++, the Java branch is much more widely used. If we talk about UIMA, we are referring to the Java version of it. Configuring UIMA components (types, annotators) is generally done by creating XML descriptor files which are tightly coupled with their Java

[1] http://www.ibmwatson.com/

[2] http://www.ukp.tu-darmstadt.de/teaching/courses/ws-1213/natural-language-processing-and-the-web/



```
0123456789012345678901234567
This is an example sentence.

Sentence(0,27)

Token(0,3)    POS(0,3).v(ART)
Token(5,6)    POS(5,6).v(V)
Token(8,9)    POS(8,9).v(ART)
Token(11,17)  POS(11,17).v(NN)
Token(19,26)  POS(19,26).v(NN)
Token(27,27)  POS(27,27).v(PUNC)
```

Figure 3.1: Visualization of UIMA annotations

implementations. To make that process simpler and to keep it more consistent, UimaFit is commonly used with UIMA as well. UimaFit [Ogren and Bethard, 2009] is a library that provides factories, injection and testing utilities for UIMA. Based on UIMA we additionally use DKPro [Gurevych et al., 2007], which builds on UimaFit and is a collection of natural language processing (NLP) software components based on the Apache UIMA framework. Having the goal of providing standard NLP components, the scope ranges from stemming, lemmatization and part-of-speech tagging over morphological analysis and syntactic and dependency parsing to spelling correction, grammar checking and working with n-grams.

3.1.1 Annotators

In this section, I will describe some non-trivial UIMA annotators worth mentioning that have been developed and/or used during the thesis.

Sentence annotation

In the data set that we chose the text is pre-processed and available as tokens (single word forms), i.e. it is *tokenized*. For our contextual resources we need to know the sentences, though. That is why we have to use a so called *sentence annotator*, also called *sentence splitter*, that performs sentence annotations out of free (token) text. There are several built-in ones available in DKPro (or made available by an interface) for at least English and German. In this thesis, the sentence segmenter of the Stanford CoreNLP[3] components is used due to its high quality and robustness. However, sentence splitting is not a straightforward, simple task and hence there will be erroneous splits (such as after abbreviations and initials). Tokenization and sentence splitting is a research field on its own. Since the contextual resources rely on the quality of the sentence annotations, that has to be mentioned because it may influence the performance of those lexical expansions resources.

WordNetRelationAnnotator

During experimentation, we were exploring if there are any particular word relations (like synonymy or co-hyponymy) that have special use for lexical expansion. For that purpose, we were using WordNet [Miller, 1995] and the Java WordNet Interface by Mark A. Finlayson [Finlayson, 2013]. Using that WordNet accessing tool, our annotator is able to denote a relation (if any is available), given two word forms, i.e. the original term appearing in a query or index document on the one hand and the expansion term retrieved on that term on the other hand. Since WordNet does not operate on words, but rather *word senses*, that is done by going through all possible senses of a surface word form and checking whether or not there is a connection between the two sense combinations.

[3] http://nlp.stanford.edu/software/corenlp.shtml

ExpansionsTerrierTagnamesIndexAnnotator

At the same time we transform the RobustWSD index documents into the tagged format of Terrier (see Section 3.2), the index-side expansions are added. Depending on the given resource, different combinations of expansions are retrieved (e.g. the top10 per term) and assigned different tag names, so that they can be allocated different expansion modes later (see Listing 5.1.1 on page 37 for a visual). Exemplarily for the DT, the tag name combinations consist of the part-of-speech, the topN limitation (top5, top10, etc.) and the per term and per sentence distinction. The goal of this annotator is to retrieve and assign expansions to different categories to build a huge index out of them, so index creation is done once and retrieval retains flexibility.

ExpansionsTerrierTagnamesQueryAnnotator

Since all the different conditions for expansion can be applied to the queries as well (with the sentence context being the whole query), such an annotator is needed for the topics in the same way. It annotates all the combinations likewise, but those in turn are used in the pipeline to output single line queries for Terrier (i.e., one query in one line of a file).

3.2 Terrier

To perform experiments in the field of Information retrieval, some kind of search engine software is needed. Among the most popular ones is *Lucene*, an Apache open source software project. Since our scientific work was based on [Wolf et al., 2010] and the framework they used, we as well use Terrier (TERabyte RetrIEverR) [Ounis et al., 2006] (now version 3.5) for indexing the documents and performing the search. Terrier is an open source search engine developed at the school of computing science at the University of Glasgow. It is used for research and experimentation in text retrieval and can be used for local and web retrieval. It supports a variety of retrieval models (including TF-IDF and BM25) and built-in query expansion. What is more, it allows manual setting of query term weights before retrieving. That will be explored in chapter 5.4. Furthermore, Terrier can even be combined with Hadoop MapReduce for indexing. Especially experiments with large collections could benefit from that in terms of saving time.

3.2.1 Fields

In Lucene and in Terrier, index terms are associated with/assigned to index **fields**. A field is an abstract categorization for terms – like a column in a database. Fields can be used to divide documents into vivid fragments like title, body and footer. But fields can also be used for rather conceptual fragmentations like word forms, e.g. the surface token and the lemmatization form. While query terms are combined by the OR operator, fields on the index side are combined by the AND operator by default[4]. Furthermore, DKPro IR applies all query terms on all given fields by default, whereat individual weights can be given for each index field. However, it is also possible to devise a more complex query with different query terms in different fields, as shown in formula 3.1. If the user does not utilise fields explicitly, the default field is used by Terrier, wherein all terms are put into.

$$(field1 : term1\ term4\ ...)\ AND/OR\ \ (field2 : term2\ term3\ term5\ ...) \tag{3.1}$$

One use case of such an elaborate formulation could be the division into title terms only matching title terms of the documents texts and so on. In summary, Terrier offers the following search possibilities:

[4] http://ir.dcs.gla.ac.uk/wiki/Terrier/QueryLanguage

1. Don't use (multiple) fields on index side (and thus, not on query side as well).
2. Use multiple fields on index side.
   a) query for all terms in all index fields likewise
   b) query for different terms in different fields

The use of all these possibilities will be explored in chapter 5.5.

3.3 Lexical Expansion

For all the lexical expansion resources mentioned in chapter 1.3 there either was an UIMA annotator or database available from which to retrieve the terms.

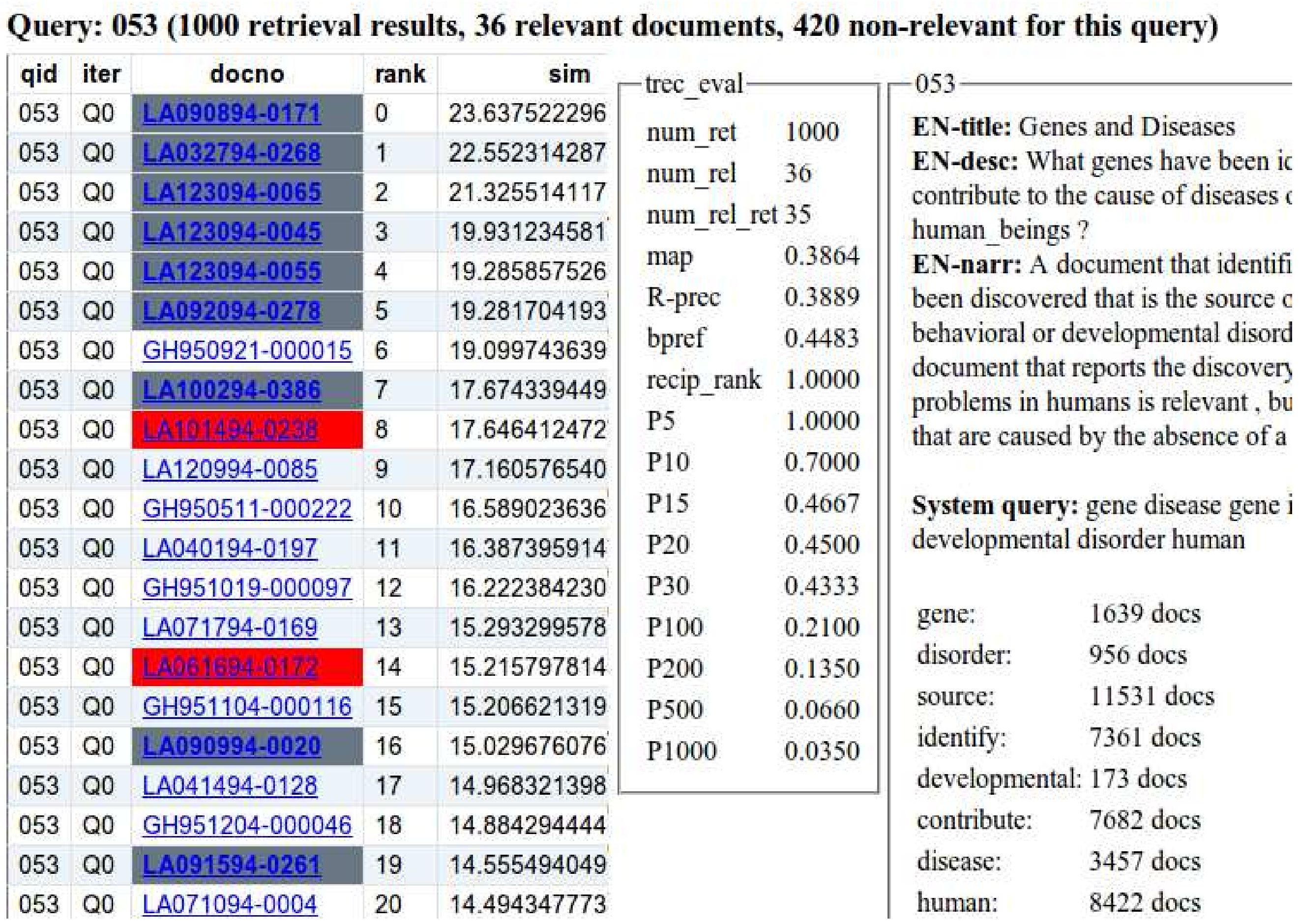
3.4 Evaluation comparison

To determine the retrieval performance of our system, the *trec_eval* tool is used[5]. Since the entire cycle (index, search, evaluation) is command-line based, there is no visual breakdown which one could use to inspect the results in a fast and easy way or even conduct comparisons between two approaches. There is a possibility to use Terrier for Web-based search[6], but that addresses a slightly different use. Since the regular indices cannot be used with the Web-based interface (because by default Terrier does not store document snippets, abstracts and meta-data) the document collections needed to be indexed from scratch, which would cause a practical overhead. That's why we developed a web browser based representation of the search result output, incorporating the *trec_eval* results. For the sake of clarity, the visualization is split into one page per query. The search results can be browsed per query and each page includes the following information:

- the recall[7] and the **number of relevant and non-relevant documents** for that query
- the **search results** (cut to the first 1000 entries, displaying the document ID and the rank)
- the full **trec_eval output** for that query
- the **textual** representation of the **query and its parts**
- the **query term frequencies** in the document collection
- the **actual query** that was used (after pre-processing, filtering, etc.)

In addition, the rank search output list contains links to all the index documents. In figure 3.2 you can see a cropped screenshot of it containing all the information mentioned. All coloured index document names are ones being manually evaluated (dark = relevant, red = irrelevant).

[5] See Chapter 4.3 for the evaluation methodology.
[6] http://terrier.org/docs/v3.5/terrier_http.html
[7] See chapter 4.3.2 for the definition of recall.



Query: 053 (1000 retrieval results, 36 relevant documents, 420 non-relevant for this query)

qid	iter	docno	rank	sim
053	Q0	LA090894-0171	0	23.637522296
053	Q0	LA032794-0268	1	22.552314287
053	Q0	LA123094-0065	2	21.325514117
053	Q0	LA123094-0045	3	19.931234581
053	Q0	LA123094-0055	4	19.285857526
053	Q0	LA092094-0278	5	19.281704193
053	Q0	GH950921-000015	6	19.099743639
053	Q0	LA100294-0386	7	17.674339449
053	Q0	LA101494-0238	8	17.646412472
053	Q0	LA120994-0085	9	17.160576540
053	Q0	GH950511-000222	10	16.589023636
053	Q0	LA040194-0197	11	16.387395914
053	Q0	GH951019-000097	12	16.222384230
053	Q0	LA071794-0169	13	15.293299578
053	Q0	LA061694-0172	14	15.215797814
053	Q0	GH951104-000116	15	15.206621319
053	Q0	LA090994-0020	16	15.029676076
053	Q0	LA041494-0128	17	14.968321398
053	Q0	GH951204-000046	18	14.884294444
053	Q0	LA091594-0261	19	14.555494049
053	Q0	LA071094-0004	20	14.494347773

trec_eval

num_ret	1000
num_rel	36
num_rel_ret	35
map	0.3864
R-prec	0.3889
bpref	0.4483
recip_rank	1.0000
P5	1.0000
P10	0.7000
P15	0.4667
P20	0.4500
P30	0.4333
P100	0.2100
P200	0.1350
P500	0.0660
P1000	0.0350

053

EN-title: Genes and Diseases
EN-desc: What genes have been ic
contribute to the cause of diseases c
human_beings ?
EN-narr: A document that identifi
been discovered that is the source c
behavioral or developmental disord
document that reports the discovery
problems in humans is relevant , bu
that are caused by the absence of a

System query: gene disease gene i
developmental disorder human

gene:	1639 docs
disorder:	956 docs
source:	11531 docs
identify:	7361 docs
developmental:	173 docs
contribute:	7682 docs
disease:	3457 docs
human:	8422 docs

Figure 3.2: Screenshot of our web-based evaluation tool

4 Dataset and evaluation methodology

In this chapter, I'll give a detailed description of the evaluation data sets and the evaluation that I used.

4.1 Dataset 1: RobustWSD

In order to gain a comparable and approved experiment setup we chose the Robust-WSD corpus from the CLEF[1] 2008 and 2009 Robust-WSD Task [Agirre et al., 2009], which is based on the widely used Los Angeles Times 1994 (LA94) and Glasgow Herald 1995 (GH95) news collections. Although the corpus originally was used to explore the contribution of Word Sense Disambiguation (WSD) to monolingual and multilingual IR, there are reference values for stand-alone retrieval on this corpus. Furthermore, it has some built-in advantages: It is already tagged with part-of-speech (POS), indicating the lemma and word form, and it comes with automatically tagged senses from WordNet (version 1.6) using two different state-of-the-art WSD systems. The POS tags stem from the Penn Treebank tagset [Marcus et al., 1993]. Making all that information available in XML format, there is much less need of individual (and error-prone) pre-processing. In listing 4.1 you can see an example how a query looks like (excerpt).

Listing 4.1: Excerpt of a query from the Robust-WSD corpus

```
1
2  <top>
3    <num>C041</num>
4    <EN-title>
5      <TERM ID="C041-1" LEMA="pesticide" POS="NNS">
6        <WF>Pesticides</WF>
7        <SYNSET SCORE="1" CODE="10749669-n"/>
8      </TERM>
9      <TERM ID="C041-2" LEMA="in" POS="IN">
10       <WF>in</WF>
11     </TERM>
12     <TERM ID="C041-3" LEMA="baby" POS="NNP">
13       <WF>Baby</WF>
14       <SYNSET SCORE="0.260057371688911" CODE="07093842-n"/>
15       <SYNSET SCORE="0.739942628311089" CODE="07094006-n"/>
16     </TERM>
17     <TERM ID="C041-4" LEMA="food" POS="NNP">
18       <WF>Food</WF>
19       <SYNSET SCORE="1" CODE="00011575-n"/>
20     </TERM>
21   </EN-title>
```

There are two problems with this data set, especially affecting the short *title* part of the query. First, due to the shortness of the title there are parser errors over and over again, declaring commons nouns (NN) as proper nouns (NNP), as can be seen in lines 12 and 17. Refer to Table 4.2 for a reinforcement of that argument. Since we are relying on those parts-of-speech in our experiments and in particular

[1] The Cross Language Evaluation Forum (CLEF) was launched as a European counterpart to the Text REtrieval Conference (TREC), which is driven by the US government's National Institute of Standards and Technology (NIST).

disregarding proper nouns (because it doesen't make sense to expand them), only taking the title into account might have an influence on the results. Because there are actual proper nouns in some query titles, there is no simple solution to that. Second, the data set include the part-of-speech of the token (*word form*) and the token's lemma, but it doesen't explicitly denote the lemma's part-of-speech. That simply requires a mapping of the two notations, which is done in chapter 5.3.1.

4.1.1 Index documents

The corpus consists of 56,472 index documents from the GH95 and 110,245 from LA94, making 166,717 documents in total. Each document consists of two parts, a headline (few terms) and a text (most of the terms). In our experiments, we do not distinguish between these two tags, but just take all the terms into account equally.

In Table 4.1 you can find some statistics about the number of terms in each of the collections to get an impression of the document base.

Table 4.1: The index documents collection

Collection	#documents	#terms	Av. #terms/doc
GH95	56,472	27,716,878	490.81
LA94	110.245	72.027.926	653.34
both	166,717	99,744,804	598.29

If you assume to comprise 400 to 700 words on a page printed on paper, you could recognize these documents as rather short ones (one page).

4.1.2 Queries

To test and evaluate an IR system, we also need given information needs, so called *topics*. The corpus provides a training dataset, consisting of 150 queries (IDs range: C041-C140, C201-C250), and a testing dataset with 160 queries (ranging from C141-C200 and C251-C350). Each query consists of three parts: title (T), description (D) and narrative (N). An example looks like this (dumped text form):

EN-title: Pesticides in Baby Food
EN-desc: Find reports on pesticides in baby food .
EN-narr: Relevant documents give information on the discovery of pesticides in baby food . They report on different brands , supermarkets , and companies selling baby food which contains pesticides . They also discuss measures against the contamination of baby food by pesticides .

The topic above is the first one from the training set (C041). As you can see, the more parts are taken, the more expressive the topic gets. Since more (or less) terms have a critical impact on the retrieval, this is a crucial part for the evaluation of the retrieval performance. Furthermore, since the part-of-speech distribution within the queries will have an impact on the query-side experiments, it is displayed in Table 4.2 (given in %). The table includes a distinction between common nouns (NN) and proper nouns (NNP).

The short title of the query hardly contains any verbs. As it gets more descriptive, whole phrases are formed using verbs. While the amount of adjectives (JJ) stays on a similar (low) level, the (often erroneous) distribution of proper nouns turns in favour of the common nouns. That statistics support our observation of many NN/NNP tagging errors in the data set.

Table 4.2: Query set POS distribution in % (RobustWSD, train)

	T	TD	TDN
JJ	7.52	7.62	11.27
NN	18.50	29.20	41.41
NNP	51.22	21.53	16.90
VB	2.85	13.49	19.07
OTHER	19.92	48.76	45.45

4.2 Dataset 2: Tipster disks 4 & 5

As a second data set, we use the *Tipster*[2] *disks 4 & 5* since it has much more relevance judgements per query. It was used as a dataset for the ad hoc retrieval tasks at TREC 7 and 8. The details of the assessments will be looked at in Section 4.3.1.

4.2.1 Index documents

Disk 4 is comprised of material from the Financial Times Limited (1991, 1992, 1993, 1994), the Congressional Record of the 103rd Congress (1993), and the Federal Register (1994). Disk 5 includes material from the Foreign Broadcast Information Service (1996) and the Los Angeles Times (1989, 1990). In Table 4.3 the collection statistics can be found [Voorhees and Harman, 2000]. We explicitly exclude the Congressional Record (CR) documents, since they come from a different domain and hence can hurt peformance [He and Ounis, 2006].

Table 4.3: Tipster document collection statistics

Collection	#documents	Mean #Words/Doc
Disk 4: Financial Times (ft)	210,158	412.7
Disk 4: Federal Register (fr94)	55,630	644.7
Disk 5: Foreign Broadcast (fbis)	130,471	543.6
Disk 5: LA Times (latimes)	131,896	526.5
total	528,155	-

What is similar to the RobustWSD collection is the mean size of the documents, resulting in about one page sized documents. As with the RobustWSD collection, the relevance judgements will be explored in section 4.3.1. The documents collection is structured in a slightly different way as the four different collections do not use the exact same set of tags and as one collection includes markup language comments. Following the official Terrier Wiki[3], five specific tags are to be indexed for that particular collection set. Table 4.4 shows a coverage of the tags availability in the four document collections.

As you can see, a text title is called differently in nearly each of the collections. The *TTL* tag could not be found in any of the collections. Next, the *doctitle* and *h3* tags contents are single-lined, while the *headline* tags contain multi-line text.

4.2.2 Data pre-processing

In contrast to the RobustWSD collection, neither the Tipster index documents nor the topics come pre-processed. Furthermore, it does not come in a separated, one-document/topic-per-file format, but rather

[2] Tipster was the name of the US government's funding program.
[3] http://ir.dcs.gla.ac.uk/wiki/Terrier/Disks4&5

Table 4.4: Index document tags coverage (Tipster)

	fr94	ft	fbis	latimes
TEXT	✓	✓	✓	✓
DOCTITLE	✓	-	-	-
HEADLINE	-	✓	-	✓
H3	-	-	✓	-
TTL	-	-	-	-

in an all-in-one-file format. That is, not the same UIMA reader component can be used and pre-processing (mainly tagging) and splitting has to be performed once in order to make that data available as input for the same processing (and expanding) pipeline used throughout the thesis (like the format shown in listing 4.1 on page 29). Additionally, individual tagging may result in different starting conditions for comparison. Listing 4.2 shows an example excerpt of one of the two topic files (containing 50 queries each) and visualizes its flaws (missing closing tags, inconsistent tag explanation verbosity).

Listing 4.2: Excerpt of a query from the Tipster disks

```
<top>

<num> Number: 351
<title> Falkland petroleum exploration

<desc> Description:
What information is available on petroleum exploration in the South Atlantic
   near the Falkland Islands?

<narr> Narrative:
Any document discussing petroleum exploration in the South Atlantic near the
   Falkland Islands is considered relevant. Documents discussing petroleum
   exploration in continental South America are not relevant.

</top>

<top>

<num> Number: 352
<title> British Chunnel impact
[..]
```

The two alternatives to process that data format are described as follows.

Transform data into RobustWSD format

The first alternative is to transform the data (that is, to write a short additional pipeline) and output it as the same format as referenced with the RobustWSD style (one topic per file, fine-grained topic tag set, POS-tags and lemmas within the tags). Then continue with the expansion pipeline. The advantage of that option is that afterwards just the same pipeline can be used as before. The disadvantage is the outsourcing into another pipeline and the extra effort.

Write separate collection reader

The second option would be just to write another UIMA collection reader for the data and use the annotations as a processing layer and use the rest of the available pipeline as given. The advantage is the integration into the complete framework by just having and switching the reader component. The disadvantage is that the one-CAS-per-file logic may get confused, as it is cleaner to separate the processing logically.

We implemented the first option (implement separate pipeline to transform the data once), since it requires a one-time transition and afterwards allows to follow the one-CAS-per-file logic. The data transformation of the Tipster topics was done in an UIMA annotation style annotating the different parts of the query and afterwards creating tags out of it. For the POS-tagging (which is necessary to do either way) the TreeTagger was used [Schmid, 1994]. The UIMA wrapper for the TreeTagger has been provided by DKPro. In order to assess whether the same NN/NNP-tagging[4] errors occur, Table 4.5 give the same overview for the POS-tags distribution within the query parts as with the RobustWSD set on page 31.

Table 4.5: Query set POS distribution (Tipster)

	T	TD	TDN
JJ	13.21	10.66	15.26
NN	56.43	37.56	41.99
NP	12.14	6.35	3.82
V	5.00	17.13	20.15
OTHER	13.21	46.07	45.11

As can be seen, not even half as much nouns are erroneously tagged as proper nouns as in the Robust-WSD query set. The percentage of proper nouns stays on a low level, even in the title representations, while the distribution of common nouns stays on an expected high level.

4.2.3 Topics

The query information needs are expressed in one hundred new topics, ranging from IDs 351-400 (TREC-7) and 401-450 (TREC-8). As with the RobustWSD topics set, the Tipster queries consist of a title, a description and a narrative. For an example, see the Listing 4.2 on page 32.

4.3 Evaluation

In order to compare our results and actually recognize improvements (or impairments), we need to verify our retrieval results to a set of assumed relevant documents, a *gold standard*. This is called the Cranfield paradigm [Cleverdon, 1997]. These judgements define for a given document and a given query whether the document is relevant to that query. The judgements may either be binary (i.e., relevant or not relevant) or graded (e.g., “excellent“/“good”/“poor”).

4.3.1 Assessing relevance

With the document collections being so large, it is not feasible (i.e., by humans) to manually judge every single document for relevance. As further described by [Agirre et al., 2009], the pooling technique is used to calculate approximate recall values. The technique of pooling means to perform runs of several IR systems in the first place and afterwords collecting the top ranked documents from each system into a so called *pool*, representing a small subset of the entire document collection. The documents in the pool

[4] In contrast to the tag set used in RobustWSD, the TreeTagger uses *NP* for a proper noun since it builds on the Penn Treebank Tagset [Marcus et al., 1993].

set are then manually judged for relevance by human jurors [Manning et al., 2008]. [Büttcher et al., 2007] even go as far as stating, that every IR evaluation based on the pooling method is inherently biased against systems that did not contribute to the pool of judged documents. While pooling might be suited to evaluate the top N search results by systems that contributed to the pool, reusing them for other systems may be difficult. By using the existing relevance set they trained a classifier to predict the relevance of documents not found in the pool of judged documents. After, they based their evaluation on the new, extended set of judgements. They were able to produce highly reliable evaluation results and even produce more accurate evaluation results than bpref[5].

RobustWSD

In the Robust WSD collection, the total number of assessments was 66,440 documents in the test set with 4.327 of them being relevant. Each topic has an average of about 28 relevant documents, seven queries have no relevant documents at all (that is 4.375% of the test set). In the train set (which is used for development) even 12,67% of the topics have zero relevant documents. Each document not judged with regard to a query (neither as relevant nor as irrelevant) is regarded as not relevant by the metrics. Table 4.6 gives an overview over the judgements available.

Table 4.6: Relevance statistics (RobustWSD)

Data set	#queries	#judgements	#relevant
train	150	67,355	2,052
test	160	66,440	4,327
total	310	133,795	6,379

Considering the amount of documents (166,717), that makes 25,007,550 possible (query,document) pairs for the train set, from which just 0,27% have been looked at.

Tipster disks 4 & 5

In the Tipster disks, the number of assessments is 167,175 with 9,402 of them being judged as relevant. Table 4.7 gives a more detailed overview.

Table 4.7: Relevance statistics (Tipster disks 4 & 5)

Data set	#queries	#judgements	#relevant
TREC-7	50	80,345	4,674
TREC-8	50	86,830	4,728
total	100	167,175	9,402

Each topic has at least 100 judgements (6 relevant), at the utmost 2992 (361 relevant) and on average 1671.75 assessments (94 relevant). So all in all, with on average three times as much relevant documents per topic as with RobustWSD, the Tipster dataset and judgements suggest a greater deal of quality assessment.

4.3.2 Metrics

In this section, I will present several IR evaluation measures used within this thesis and its evaluation tool and discuss their relevance to our retrieval task. I will differentiate between binary classification (i.e., relevant or not relevant) and ranking aware metrics, taking the order of the results into account.

5 See subsection 4.3.2 for a definition of bpref.

Binary classification

In the field of Information Retrieval *precision* and *recall* are well-known metrics. By distinguishing between *relevant documents* (that are considered as a relevant result to a particular query) and *retrieved documents* (the ones returned by the retrieval system) in a binary classification task, precision can be given as in Equation 4.1.

$$\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}|} \tag{4.1}$$

Precision basically is the number of relevant documents in relation to the retrieved documents. So, precision tells us which fraction of the documents contained in the result set actually is relevant (and how much noise is in the results set). In contrast to that, the fraction of the relevant documents contained in the result set is called *recall* (see equation 4.2) and tells us how much is missing.

$$\text{Recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \tag{4.2}$$

Precision and recall are connected with each other, in fact they are opposed. Typically considered in the range [0,1] with high values being better. While a recall of 1.0 can always be obtained (by having a large result set), even high precision values can be influenced (by having a small results set with many of them known as relevant). Known from other classification tasks (such as part-of-speech tagging) there are measures to combine both of them, such as fall-out, accuracy and the F-measure, especially the F1-measure. Since quality in IR is much more about ranked results than rather about result sets, they are of little use in IR and hence we will neither consider those.

Ranking aware metrics

Precision accounts for all retrieved documents, but it can also be evaluated at a fixed cutoff rank, regarding only the top k results returned by the system. This measure is called precision at k or short *P@k*. While p@k will yield interesting insights in web retrieval for low values of k, the case of having less than k relevant documents is a general problem of that measure (i.e., it is considered *unstable*). Having a flexible cutoff at the number of relevant documents for a given query is called *R-precision*. In other words, it is precision at the R-th position in the ranking of the results for a query having exactly R relevant documents (recall and precision are equal at the R-th position). [Kishida, 2005] formalize it as given in equation 4.3.

$$\text{R-precision} = \frac{1}{R}\sum_{i=1}^{R} x_i \tag{4.3}$$

Let R be the total number of relevant documents and with x_i being 1 if the ith document is relevant to the current query and 0 otherwise. A measure correlated to that is *Average Precision* (AP), measuring the system quality at all recall levels by averaging the precision for a single query. [Buckley and Voorhees, 2000] define it as "*the mean of the precision scores obtained after each relevant document is retrieved, using zero as the precision for relevant documents that are not retrieved*". Following equation 4.3 it can be formulated as follows:

$$\text{Average precision (AP)} = \frac{1}{R}\sum_{i=1}^{n} x_i \bar{x}_i \tag{4.4}$$

In equation 4.4, n is the number of documents included in the list (usually n = 1000) and by $\bar{x}_i$ the average of values $x_1, ..., x_n$ is denoted. While R-precision and average precision indicate values for single queries, the mean for a set of queries is call *Mean Average Precision* (MAP) and simply divides the sum of the AP for all queries by the number of queries.

$$\text{Mean average precision (MAP)} = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{4.5}$$

With MAP being one of the most common measures in IR research it will also be the most important one in our thesis.

All the metrics introduced so far assume to know the relevance of each and every document. As we have seen in subsection 4.3.1, only a tiny fraction of all possible (query, document) pairs is actually evaluated. One metric that doesen't need this assumption is called *bpref* [Buckley and Voorhees, 2004], short for *binary preference*. It is a preference-based measure considering how often (known) irrelevant documents are ranked above (known) relevant ones, disregarding documents not judged. Similar to R-precision R relevant documents are assumed. Then, all pairs of relevant and not relevant documents are determined. As displayed in equation 4.6, it is then summed up over all relevant documents (r) how often a not relevant (n) document is ranked above.

$$\text{bpref} = \frac{1}{R} \sum_{r} \left(\frac{|\text{n is ranked above r}|}{R} \right) \tag{4.6}$$

It could be shown that bpref correlates with MAP quite often, but is more stable on a small judgements base. We will not focus on it here, but still keep that in the back of our mind.

4.3.3 Evaluation tool

The Terrier package incorporates a built-in evaluation tool, which is said to be compatible with the commonly used *trec_eval* tool. However, while experimenting I found some inconsistencies which led us to abandon that built-in tool. First, in their evaluation they only took into account so called *effective queries*, i.e. queries that do have at least one relevant document assigned to it. With those topics being 19 in the train set and seven in the test set, retrieval results changed significantly (precision values usually increase). Second, the built-in tool only outputs a fraction of the the measures that *trec_eval* is capable of (the fraction does not include bpref).

5 Experiments

In the following sections the experiments conducted will be described. Results of the approaches are presented in each section separately. We tried to base our setup upon the one used by [Wolf et al., 2010]. Due to older software library versions, that was not always possible.

Furthermore, note that it is crucial to explicitly state which parts of the query are used (as explained in chapter 4.1.2). Therefore, that will always be mentioned along with every result shown.

5.1 Pipelines and pre-processing

Following the UIMA concept explained in chapter 3.1, we apply two separate pipelines. The first one is used for indexing the document collection. The second one performs the search and outputs results ready for evaluation. Despite the conditions the data set provides us with, we additionally apply some pre-processing steps. For the sake of completeness, the whole pipelines are given in table 5.1.

Table 5.1: Both pipelines displaying the pre-processing steps

Indexing	Searching
read the collection	read the collection
AnnotationRemover	AnnotationRemover
CharacterSplitReannotator	CharacterSplitReannotator
	SentenceSplitter
StopWordRemover	StopWordRemover
SnowballStemmer	SnowballStemmer
Lexical expansion (optional)	Lexical expansion (optional)
	AnnotationRemover
IndexTermAnnotator	IndexTermWeightAnnotator
AnnotationRemover	AnnotationRemover
	TerrierQueryGenerator
	TerrierSearcher
	SearchResultConsumer

As can be seen, both procedures essentially are the same. Since the query pipeline incorporates employing the search and outputting the results, there are some extra steps needed. The SentenceSplitter component, which performs sentence annotations, is required to make use of the contextualized lexical expansion resources. All the AnnotationRemover calls basically are necessary cleanups to remove all annotations from terms that could cause problems, including special characters for the Terrier search. Those calls are necessary after each step producing unanticipated output (like the expansions).

5.1.1 Index documents pre-processing

For two out of the three methods involving index side expansions, the index documents need to be enhanced before retrieval. Since indexing a large collection of documents is a very time-consuming computational task, it is advisable to do it as less often as possible. That is why we try to include as much information as possible within the index and only use certain parts for different experiments. When expanding index-side, one can only weight using index fields (as discussed in section 3.2.1). That is, once in the index, there is no more possibility to weight individual terms. That is why we need to take care of the manner and the number of terms to put into the expansions field. In conclusion, to save indexing time and still keep flexibility using expansion terms, one could incorporate multiple expansions fields into the index semantically displaying multiple expansions modes. Listing 5.1 shows an example (excerpt) of how the expanded document representations look. In this example the DT is used since it is the most comprehensive resource we utilized and thus will result in much more tag names.

Listing 5.1: Expanded document representation for indexing (DT)

```
<DOC>
        <DOCID>...</DOCID>
        <TOKEN>...</TOKEN>
        <LEMMA>...</LEMMA>
        <STEM>...</STEM>
        <EXPANSION_DT_TOP1_PERTERM_ALL>...</EXPANSION_DT_TOP1_PERTERM_ALL>
        <EXPANSION_DT_TOP3_PERTERM_ALL>...</EXPANSION_DT_TOP3_PERTERM_ALL>
        <EXPANSION_DT_TOP5_PERTERM_ALL>...</EXPANSION_DT_TOP5_PERTERM_ALL>
        <EXPANSION_DT_TOP10_PERTERM_ALL>...</EXPANSION_DT_TOP10_PERTERM_ALL>

        <EXPANSION_DT_TOP1_PERTERM_NN>...</EXPANSION_DT_TOP1_PERTERM_NN>
        <EXPANSION_DT_TOP3_PERTERM_NN>...</EXPANSION_DT_TOP3_PERTERM_NN>
        <EXPANSION_DT_TOP5_PERTERM_NN>...</EXPANSION_DT_TOP5_PERTERM_NN>
        <EXPANSION_DT_TOP10_PERTERM_NN>...</EXPANSION_DT_TOP10_PERTERM_NN>

        [...]

        <EXPANSION_DT_TOP10_PERDOC_ALL>...</EXPANSION_DT_TOP10_PERDOC_ALL>
        <EXPANSION_DT_TOP50_PERDOC_ALL>...</EXPANSION_DT_TOP50_PERDOC_ALL>
        <EXPANSION_DT_TOP100_PERDOC_ALL>...</EXPANSION_DT_TOP100_PERDOC_ALL>
        <EXPANSION_DT_TOP500_PERDOC_ALL>...</EXPANSION_DT_TOP500_PERDOC_ALL>

        [...]

</DOC>
```

All is short for *all part-of-speech*. Next to NN (common nouns), we included JJ (adjectives) and VB (verbs). Similar holds for TWSI and Delex. Even though they are contextual resources and operate *per sentence* instead of *per term*, the tag names will still be *perterm*. Second, as for the TWSI we only work with nouns, so the part-of-speech adding to the tag name is not needed. Apart from that, we will use the same quantitative limitations (top 1, 3, 5 and 10 per sentence, top 10, 50, 100 and 500 per document). Listing 5.2 shows the entire structure of the TWSI expanded documents.

Listing 5.2: Expanded document representation for indexing (TWSI)

```
<DOC>
        <DOCNO>GH950102-000001</DOCNO>
        <TOKEN>...</TOKEN>
        <LEMMA>...</LEMMA>
        <STEM>...</STEM>
        <EXPANSION_TWSI_TOP1_PERTERM>...</EXPANSION_TWSI_TOP1_PERTERM>
        <EXPANSION_TWSI_TOP3_PERTERM>...</EXPANSION_TWSI_TOP3_PERTERM>
        <EXPANSION_TWSI_TOP5_PERTERM>...</EXPANSION_TWSI_TOP5_PERTERM>
        <EXPANSION_TWSI_TOP10_PERTERM>...</EXPANSION_TWSI_TOP10_PERTERM>
        <EXPANSION_TWSI_10_PERDOC>...</EXPANSION_TWSI_10_PERDOC>
        <EXPANSION_TWSI_50_PERDOC>...</EXPANSION_TWSI_50_PERDOC>
        <EXPANSION_TWSI_100_PERDOC>...</EXPANSION_TWSI_100_PERDOC>
        <EXPANSION_TWSI_500_PERDOC>...</EXPANSION_TWSI_500_PERDOC>
</DOC>
```

Since that resource focuses on a particular part-of-speech (nouns), the size of the resulting document collection is about a third compared to the DT.

Listings 5.1 and 5.2 display the tagged format of the text data which is suitable for indexing with Terrier. Both the baseline and all expansion experiments are represented in this way. Data set 1 was available in the RobustWSD format, data set 2 (Tipster) had to be transformed into the same format to follow the pipeline. Figure 5.1 gives an overview over the data transformations performed to be able to index the collection using Terrier in our setting.

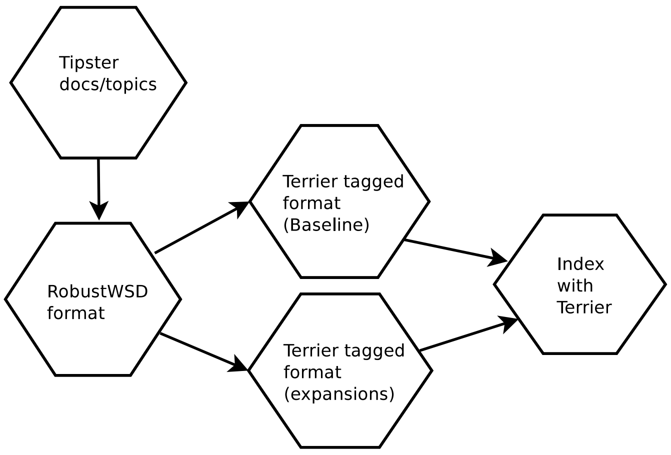

Tipster docs/topics
RobustWSD format
Terrier tagged format (Baseline)
Terrier tagged format (expansions)
Index with Terrier

Figure 5.1: The data transformations (index side)

5.2 Baseline

In the paper by [Wolf et al., 2010], before their own experiments, they report a stand-alone retrieval baseline of 0,4054 in MAP on the RobustWSD collection. They used the lemma for indexing and the *DFR_BM25* model for retrieval. From the query, they used TD (title and description). We tried to follow their pre-processing steps and parameters, which led us to the baseline results in table 5.2. Additionally, [Pérez-Agüera and Zaragoza, 2008] use the RobustWSD corpus and report a baseline retrieval value of 0,3614. (Still it is unclear, which parts of the query they used.)

To display a baseline and conduct improvements based upon that, you have to consider the following questions: What types to use (Token, Lemma, Stem)? Which parts of the query to use (title, description, narrative)? This is why we will first display a combination of all of them and thereby will argue for a way to proceed. Table 5.2 shows our baseline values of all the combinations on the RobustWSD data set. We used the *DFR_BM25* retrieval model [Amati, 2003] as well, if not denoted otherwise.

First of all, one can see the impact between using concrete tokens and abstract forms (lemma, stem). Since the lemma provides a clean abstract mapping for many word forms and the terms from the lexical expansion resources come as lemma, we will use the lemma from now on. To compare against a baseline in the following chapters, in which we will work with the train set, Table 5.3 will provide detailed baseline results on the train set and serve as our final baseline to assess all expansion experiments.

Table 5.2: Baseline for different word forms (test set, TD), RobustWSD

	T[1]	L	S	T,L	L,S
recall	0.8239	0.8756	0.8738	0.8167	0.8690
map	0.3351	0.3507	0.3571	0.3289	0.3440
r-prec	0.3259	0.3353	0.3426	0.3202	0.3343
bpref	0.3219	0.3303	0.3373	0.3183	0.3314
p@5	0.4225	0.4475	0.4450	0.4325	0.4300
p@10	0.3450	0.3644	0.3663	0.3463	0.3569

[1] T = Token, L = Lemma, S = Stem

Table 5.3: RobustWSD baseline results (train set, Lemma, TD)

	recall	map	r-prec	bpref	p@5	p@10	p@15	p@20
train set	0.8748	0.3146	0.2995	0.3747	0.3147	0.2487	0.2209	0.1950

As explained in section 4.3.3 on page 36, those baseline results on the train set were produced by the *trec_eval* tool. The same holds for the experiments on the Tipster data set, for which the baseline results are displayed in Table 5.4. While the overall mean average precision is at a relatively low level, the first ranking-aware metrics are at remarkably high ones.

Table 5.4: Tipster baseline results (Lemma, TD)

	recall	map	r-prec	bpref	p@5	p@10	p@15	p@20
Tipster baseline	0.4899	0.1876	0.2405	0.2073	0.4700	0.4200	0.3840	0.3520

[Zhai and Lafferty, 2001] used the data set in their experiments as well, although they once use the title (T) of the topics only, and once the long version (title, description, narrative). They report MAP values ranging from 0.167 to 0.186 (Trec-7, title), 0.239 to 0.256 (Trec-8, title), 0.204 to 0.224 (Trec-7, long version) and 0.248 to 0.265 (Trec-8, long version) for different smoothing methods. That is, our baseline MAP values (with the experiments including the topic description) join in well, as they are placed between the lower and upper bound.

5.3 Lexical expansion sources

In this section it is explained in detail how the lexical expansion resources are applied. In addition to the introduction of those resources in chapter 1.3, some practical aspects are highlighted.

5.3.1 Distributional Thesaurus

The basic concept of using the Distributional Thesaurus (DT) for lexical expansion is quite simple - one enters a term and looks up all the similar terms associated with it. But there are several conditions to consider: Expand by how many terms, which terms of the query (or index document) to expand (e.g. filtered by their part-of-speech) and how to set their weights.

There are different version of the DT. We use the version abbreviated by *DT_120M_NP_POS* (which is constructed out of 120 million sentences), which only uses the lemma form, but distinguishes between NP and NN. The reason for the latter is not to get associated terms for proper names, which is not helpful for IR. Here is an overview of the most frequent parts-of-speech contained in the DT we use:

Since our data set is based upon tokens and hence provides the token parts-of-speech, we had to carry out a manual mapping between the token POS (which are available to us) to the DT POS (which we

Table 5.5: Displaying the parts-of-speech within the DT

DT lemma POS	Comment	Documents/Topics token POS
NP	proper nouns	NNP, NNPS
NN	all noun forms, except NP	NN, NNS
VB	all verb forms	VB, VBD, VBG, VBN, VBP, VBZ
JJ	adjectives	JJ, JJR, JJS
IN	prepositions	IN
RB	adverbs	RB, RBR, RBS
...	...	...

just intend to use). In the third column of table 5.5 you can find the token POS that are mapped to the lemma form in the first column.

Basically, there are two methods of retrieving and restricting DT terms for expansion. First, there is the most simple and intuitive one of getting expansions *per term*. That is the basic concept of the DT, picking a term and inspecting associated terms for that particular source token. Before doing so, of course we select the candidate words which are considered for expansion. Consequently, the number of expansions a document (which may be a topic or an index document) gets depends on the number of words the document consists of in the first place. Since the index documents tend to have many terms (see chapter 4.1.1) they have a large lower boundary of extra terms. If the document is enhanced by one expansion per term one effectively doubles the size of the document.

For this reason we implemented a second method to retrieve and restrict expansion terms *per document* instead. This way, the document can be extended by only a fraction of its term quantity. Figure 5.2 shows an example for the two procedures.

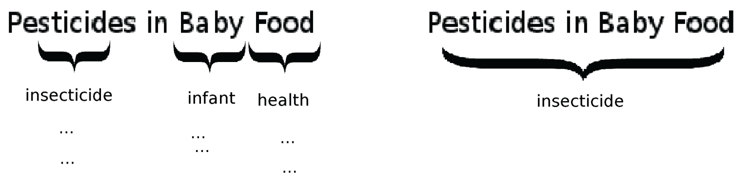


Figure 5.2: Displaying expansions per term and per document (here: query)

Since the DT can only return terms for a single word (that is, it is not *contextual*), narrowing terms per document has to operate upon that. We do this by retrieving the top ten expansion per source term and then apply a ranking between them before limiting their number. This ranking (or weighting) then again can be done by several methods. First, the sorting and cutting can be performed based on the expansion term weights set by the resource itself (relative to the source term). Furthermore, selecting source terms to expand by their frequency (within the documents corpus) is an alternative. The idea behind the latter is that more rare terms are more interesting than more general terms, e.g. in the phrase *solar energy* the second term is too general and would result in just as general additional terms. Third, one could collect the intersection of expansion terms (originated by different source words) and argue the following way. If the same terms appear multiple times as expansions, then they are good descriptors for the document.

Both expansion terms limiting methods (per term and per document) were explored in the experiments we present in this chapter.

5.3.2 TWSI Sense Substituter

The TWSI Sense Substituter can be obtained as a UIMA annotator[1]. Since the TWSI provides substitutions for words in a sentence-wide context, it is a *contextual* resource. Applying a contextual resource presumes sentence annotations. Second, unlike the DT, which provides expansions for nearly any term or part-of-speech, the sense substituter focuses on 1000 selected nouns. Hence, the expansion quantity will be smaller. But still there are the same two ways of adding and limiting expansions terms to a document using the TWSI resource. Although it is based on the sentence context, it may yield substitutions for more than one target word in the sentence. That is why we will stick to the *per term* limiting. Furthermore, since each document is a collection of sentences, we can apply the same *per document* restrictions as with the DT. In case of just using the title[2] of the topics for retrieval, there is only one sentence at most and both methods are the same. As sophisticated as this resource might be, for very short queries (e.g. *solar energy*) application is problematic (it does not work on most of them). The TWSI substitutions come with a score of how many people denoted that term, which functions as an internal ranking of the substitution terms per target word.

However, there is no obvious ranking of TWSI substitutions by document. As we have seen in chapter 1.3.2, only the score (number of votes) per target term is given. One could either use that as a global (i.e., per document) scoring, justified by the same thought as the per term scoring: If more people voted for that substitution, it is probably a more reliable one. Second, one could collect all the terms appearing in substitution lists of more than one term. Third, the substitution terms could be sorted by their original (target) terms frequency as well. The more specific target terms would result in more informative additional terms. But the worker's scores are absolute and have no meaning in comparison to substitutions from another target term. Since TWSI does not provide a score for the target term itself, they all have to be correlated in another way. One could relate a substitution's score to the sum of all substitution scores. Please consider the following example:

```
[...] <target="capital" [...] substitutions="[money,27][asset,14][fund,14][investment,14]
[cash,13][financial asset,10][finance,8][stock,8][wealth,6][property,5]"> [...]
```

In this example, the target term features ten substitutions, with a total score sum of 109 (all substitution terms containing whitespaces are ignored, and so are their scores). Hence, the term *money* will be given a weight of $\frac{27}{109} = 0{,}2477$, the terms *asset* and *investment* one of $\frac{14}{109} = 0{,}1284$ and so on.

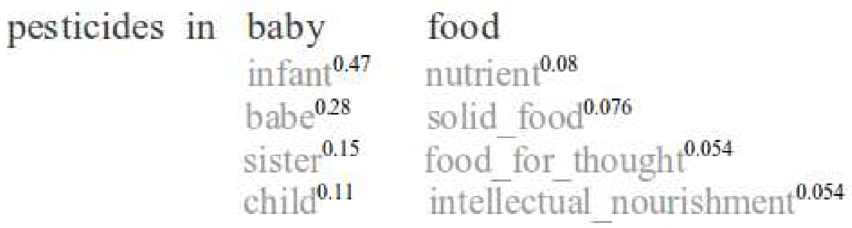
5.3.3 Delex System

For the Delex system, a model has to be trained on a corpus first (either LexSub or TWSI). Making use of machine learning algorithms as a foundation for decision making and annotation creation, the Delex system uses ClearTK [Ogren et al., 2008] as it is a suitable natural language processing component. ClearTK is built on top of UIMA. Then, the system can be applied to new data. But first of all, the Delex system was available as a separate, complex system on its own, incorporating machine learning methods and several resources. It had to be tailored to suit our purposes and made available as a single ready-to-use package (ideally as an UIMA annotator) and integrated into the existing lexical expansion pipeline.

Applying the Delex system is time-intensive, since it uses WordNet, the Web1t corpus (using up to 5-grams for each candidate word) and the DT at the same time to generate its features for candidate ranking. So selecting the target terms to get paraphrases for is the major challenge using this resource.

[1] http://www.ukp.tu-darmstadt.de/software/twsi-sense-substituter/
[2] See chapter 4.1.2 on page 30 for a recap of the topics structure.

But first, the system can only provide substitutions for terms having synonyms in WordNet (serving as candidates to be ranked). To reduce computation time, it is useful to assess a word's availability in WordNet first. Therefore, we wrote an annotator to do just that: Mark the tokens of the text that are available in WordNet and have at least one more lexeme (different from the target word) gathered by its related synsets. The availability in WordNet is a tight restriction, which may eliminate the possibility of expanding informative words. Figure 5.3 illustrates an example query title (RobustWSD, train set, C041) with all Delex expansions available[3] (after stopwords).



pesticides	in	baby	food
		infant$^{0.47}$	nutrient$^{0.08}$
		babe$^{0.28}$	solid_food$^{0.076}$
		sister$^{0.15}$	food_for_thought$^{0.054}$
		child$^{0.11}$	intellectual_nourishment$^{0.054}$

Figure 5.3: Displaying weighted Delex expansions for a query

In this particular query title, *pesticides* might well be the most informative term. However, it does not feature any entries in WordNet (other than itself), Delex neither receives nor ranks any substitution candidates for that term. But first of all, let us examine the coverage of terms ready to expand in the data set. In table 5.6, the WordNet coverage is given for the RobustWSD topics set (in percent of all tokens after stopwords).

Table 5.6: WordNet coverage RobustWSD topics

	Train	Test
T	69.90	70.38
TD	78.32	80.52
TDN	80.58	81.65

Looked up in WordNet is the lemma form of the token with its mapped WordNet entity part-of-speech (n, v, adj, adv). Taken into account are only the terms having at least one synset collected (same lexeme as target is skipped). So overall there is quite a good coverage on the tokens to have candidates in WordNet. Interestingly, the coverage increases as the query gets more verbose. This is due to the title being very succinct and containing more specific terms. As seen in the example above, expanding terms with Delex in any case may not be appropriate. That is, in some cases (queries) it may be better not to expand.

5.4 Query side

Query side expansion (short *query expansion*) means to expand queries at runtime. At the query side there is the possibility of setting different weights to individual terms and thus experiments work as follows. Due to the possibility of setting individual weights for individual expansion terms, query expansion is much more flexible than index-side ones (in which all additional terms have a common field weight). Here is an example of how a weighted query expansion representation looks like (expansions retrieved from the DT):

pesticides^1.00 insecticide^0.24 baby^1.00 infant^0.32 food^1.00 meat^0.22

[3] Coincidentally, both the terms have four candidates in WordNet.

All other (original) query terms not explicitly weighted get a custom weight of 1.0 by Terrier if manual weights are applied. However, the flexibility of that individual weighting is why this method is in need of a term-specific weighting method. First, as pointed out in section 5.3.1, there are several restriction methods of adding terms to the query.

Since the query usually is short, it is hard to fully apply contextual benefits. In this section, all the query expansion approaches are applied at non-expanded index documents. That is, only query-side expansion is examined here.

5.4.1 Distributional Thesaurus

Since the DT is an all-words expansion system, it can even deal with queries of short length (as it may contain word categories not covered by other resources). As depicted in figure 5.2 on page 41, there are (at least) two methods of adding and restricting expansion terms to a query, namely *per term* and *per query*. So for example, one could add five terms per query term (a three term query would end up with 15 expansions this way) or limit that to ten terms per entire query. But in doing so, a sorting method for expansions is needed. For this, we take the following two approaches, which will be abbreviated by *byweight* and *leastfrequent* and described as follows.

byweight
: The DT provides similarity scores to rank its expansions terms among themselves. Since it comes with self score for the target term, that can be used as a reference point for normalization. Then, expansion terms of different target words can be compared against each other.

leastfrequent
: Another way of approaching the selection of terms to expand is to look at the query terms first. Filtering terms by their *interestingness* before expansion might lead to more interesting expansions in return. One way of determine interestingness is the term's frequency. The lower its frequency, the more interesting it is.

We will start with the *byweight* method, since it is the obvious and "'built-in'" one. The *per term* and *per query* approach will be examined, each differentiated by part-of-speech. The POS we distinguish are ALL, NN, VB and JJ.

per term
: Per query term, up to 5 expansion terms will be added in separate experiments. That is, the ranks $topNstops = \{1, 3, 5\}$ will be assessed.

per query
: Since the query is very short, up to 20 expansions per query will be enough, so that $topNstops = \{1, 3, 5, 10, 20\}$ is set as check for expansions per query (representing an entire document).

As we have seen in the query example above, all expansion terms have a weight of $w \in [0, 1]$ and all original terms get a weight of $w = 1.0$. Now, setting those weights to the expanded query terms is crucial, since it is the only way of using a weighting query-side. Those weight values come from the Distributional Thesaurus itself, since it contains similarity scores for its generated terms.[4]. To map them into a range of $[0, 1]$ we divide a term's score by its target words score. In the following experiments, expanded queries are applied on the (non-expanded) document collection. The queries are prepared by part-of-speech and by the number of expansion terms, further distinguished *per term* and *per query*. Tables 5.7 and 5.8 show the results categorized by POS and number of expansions terms.

[4] Please refer to Chapter 1.3.1 on page 13 on how they are calculated.

Table 5.7: RobustWSD: DT query expansion MAP results (per term)

	top1	top3	top5	top10
ALL	0.2847	0.2708	0.2622	0.2623
JJ	0.2869	0.2806	0.2782	0.2782
VB	0.2880	0.2878	0.2892	0.2892
NN	0.2865	0.2857	0.2860	0.2860

Table 5.8: RobustWSD: DT query expansion MAP results (per doc)

	top1	top3	top5	top10
ALL	0.2859	0.2868	0.2789	0.2677
JJ	0.2869	0.2854	0.2839	0.2784
VB	0.2880	0.2879	0.2879	0.2882
NN	0.2870	0.2865	0.2855	0.2804

Table 5.9: Tipster: DT query expansion MAP results (per term)

	top1	top3	top5	top10
ALL	0.1787	0.1655	0.1581	0.1581
JJ	0.1827	0.1803	0.1793	0.1793
VB	0.1869	0.1863	0.1867	0.1867
NN	0.1854	0.1781	0.1738	0.1738

Table 5.10: Tipster: DT query expansion MAP results (per doc)

	top1	top3	top5	top10
ALL	0.1826	0.1808	0.1797	0.1775
JJ	0.1840	0.1844	0.1844	0.1844
VB	0.1873	0.1872	0.1868	0.1866
NN	0.1867	0.1862	0.1849	0.1841

First, all DT query expansion experiments hurt the performance as compared to the baseline of 0.3146 in MAP. The more terms, the worse, especially for expansions per term. There is one exception with the verbs, though. But according to Table 4.2 on page 31, they are one of the smallest group of parts-of-speech examined.

Table 5.9 and table 5.10 display the results of the same experiments applied on the Tipster data set. With a baseline value of 0.1876 in terms of MAP, all DT query expansion experiments with that data set hurt the performance likewise.

5.4.2 TWSI

In contrast to the Distributional Thesaurus, the experiments with the TWSI substitutions do not need to be prepared for by different part-of-speech (since it solely contains substitutions for nouns). Anyhow, we can apply the same two expansion modes for the query, i.e. adding and limiting them per term and per document. In Tables 5.11 and 5.12 you will find the results for those experiments on the RobustWSD data set. Compared to the baseline of 0.3146, using one expanded noun per query resulted in a slight performance gain.

In Table 5.13 and 5.14 the results for the same experiments on the Tipster data is shown. Compared to the baseline value of 0.1876, the tiny improvement from the equivalent RobustWSD experiment could not be confirmed. Even more, the Tipster results have its highest value at three expansions per query.

5.4.3 Delex

As pointed out in the general description of applying the Delex substitutions (see section 5.3.3), selecting terms to expand in the first place is crucial. This is not only important because of performance issues due to all the resources used. Also, expanding less interesting terms may harm retrieval performance. That is

Table 5.11: RobustWSD: TWSI query expansion MAP results (per term)

	top1	top3	top5	top10
NN	0.3116	0.3108	0.3118	0.3118

Table 5.12: RobustWSD: TWSI query expansion MAP results (per doc)

	top1	top3	top5	top10	top20
NN	0.3153	0.3124	0.3090	0.3136	0.3098

Table 5.13: Tipster: TWSI query expansion MAP results (per term)

	top1	top3	top5	top10
NN	0.1841	0.1842	0.1839	0.1839

Table 5.14: Tipster: TWSI query expansion MAP results (per doc)

	top1	top3	top5	top10	top20
NN	0.1848	0.1855	0.1839	0.1849	0.1840

why we implemented two term selecting methods for the query tokens to be marked for expansion. Both are based on the idea that less frequent terms are more expressive and thus more interesting to expand. The first one is based on the number of lexemes found in the original and related synsets of the target terms. For example, the narrative of the query example shown above (see figure 5.3 on page 43) is *find reports on pesticides in baby food*. In this sentence, *find* adds up to 32 lexemes and *report* to 11, whereas *baby* and *food* both result in four. Since *pesticide* is not even found in more than its original synset, that number might serve as an indicator to select tokens.

However, as mentioned earlier, selecting candidates can only be done among terms that WordNet can provide candidates for. And it is possible that all those terms are not interesting to receive expansions for. That is why our second approach is based on the lemma frequency of all terms in the query with the option of not expanding any terms in the query at all. The lemma frequency is taken from the Google n-gram corpus and annotated at every token in the query (after stopwords). The lowest n frequent lemmas are then selected as candidates for expansion. However, if in turn those lemmas have no candidates in WordNet (as with the *pesticides in baby food* query, at which *pesticide* has the lowest frequency count), no terms are expanded in this method. The following description is a summary of the two methods introduced.

lowest WordNet candidates
: Query terms are sorted by their number of WordNet candidates and the lowest ones are selected.

lowest frequency count
: First, all the tokens get an underlying lemma frequency annotation, afterwards they are sorted before expansion. If these terms don't have candidates in WordNet, no term at all will be selected for expansion.

Using the described second method in a first experiment, only 39,33% (59 out of 150) of the train topics of RobustWSD get expansions at all. But even though only a few expansion terms are added to the original set of query terms, the mean average precision slightly worsened. Table 5.15 gives a detailed overview compared to the baseline.

Table 5.15: RobustWSD: Delex and baseline results (train set, Lemma, TD)

	recall	map	r-prec	bpref	p@5	p@10	p@15	p@20
baseline	0.8748	0.3146	0.2995	0.3747	0.3147	0.2487	0.2209	0.1950
Delex top1	0.8611	0.3114	0.2974	0.3699	0.3093	0.2487	0.2196	0.1947

The top 1 expansion was added per term (if the two conditions overlap: has WordNet candidates and is the lowest frequent lemma in the sentence). Table 5.16 shows the results for the same experiments on the Tipster data set. The continuous decrease of the RobustWSD run cannot be confirmed. In contrast, while there is a small raise in MAP and *p@10*, the overall effect is negligible.

5.5 Index side

Expanding documents at index time means pre-processing the document before indexing to include additional terms to index along with the original text. Because document expansion is done once at index

Table 5.16: Tipster: Delex and baseline results (Lemma, TD)

	recall	map	r-prec	bpref	p@5	p@10	p@15	p@20
Tipster baseline	0.4899	0.1876	0.2405	0.2073	0.4700	0.4200	0.3840	0.3520
Delex top1	0.4929	0.1877	0.2398	0.2077	0.4680	0.4220	0.3827	0.3515

Table 5.17: DT expansions per term

tag name	~#tokens / doc
TOP1_PERTERM_JJ	29.32
TOP1_PERTERM_VB	45.82
TOP1_PERTERM_NN	105.73
TOP1_PERTERM_ALL	257.80
TOP3_PERTERM_JJ	87.97
TOP3_PERTERM_VB	137.46
TOP3_PERTERM_NN	317.18
TOP3_PERTERM_ALL	773.36
TOP5_PERTERM_JJ	146.62
TOP5_PERTERM_VB	229.10
TOP5_PERTERM_NN	528.63
TOP5_PERTERM_ALL	1288.89
TOP10_PERTERM_JJ	293.21
TOP10_PERTERM_VB	458.19
TOP10_PERTERM_NN	1057.23
TOP10_PERTERM_ALL	2577.56

Table 5.18: DT expansions per document

tag name	~#tokens / doc
TOP10_PERDOC_NN	1.74
TOP10_PERDOC_JJ	0.85
TOP10_PERDOC_VB	0.31
TOP10_PERDOC_ALL	9.99
TOP50_PERDOC_ALL	49.99
TOP50_PERDOC_VB	3.49
TOP50_PERDOC_NN	10.04
TOP50_PERDOC_JJ	3.25
TOP100_PERDOC_ALL	99.99
TOP100_PERDOC_VB	10.73
TOP100_PERDOC_NN	23.53
TOP100_PERDOC_JJ	6.95
TOP500_PERDOC_VB	94.45
TOP500_PERDOC_NN	167.19
TOP500_PERDOC_JJ	49.14
TOP500_PERDOC_ALL	483.34

time, it has a runtime overhead advantage for the user of an IR system, compared to query expansion methods (which are performed at query time). Basically, there are two ways of exercising lexical expansion at index side. First, one could simply add all the extended terms to the one default (token) field. This way, there is no chance of weighting the extra words (or even distinguishing them from the original document terms). That is where the use of index fields come into play. As explained in chapter 3.2.1, there are several ways of using multiple index fields in Terrier. An intuitive idea is to list all the original documents words in one field (as done with the default one) and all terms gathered by lexical expansion resources into another (separated) one. This way, there is total control on whether and how to apply expansion terms. In all the indexing experiments, the retrieval model *BM25F* of Terrier is used, which is a variation of the BM25 model for fields. The Terrier team implemented *BM25F* as described by [Zaragoza et al., 2004].

5.5.1 Distributional Thesaurus

At first, we will look at the distribution of the expansion terms in the enhanced index. Tables 5.17 and 5.18 give an overview of how many expansion terms can be found in the enhanced index by category (tag names are abbreviated).

Following the distinctions drawn in section 5.3.1, we will first start by examining the use of expansions per term. Tables 5.19, 5.20, 5.21 and 5.22 show the experimentation results for the topN (N = 1, 3, 5, 10) results for the selected part-of-speech (POS) and given several expansion field weights (the lemma fields weight remains 1.0). The results that improve both the zero field weight value as well as the general baseline of 0.3146 are highlighted.

Table 5.19: RobustWSD: DT index per term MAP results (top1)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3178	0.3191	0.3197	0.3194	0.3172	0.3178	0.3164	0.3148	0.3129
JJ	0.3135	0.3136	0.3132	0.3134	0.3134	0.3137	0.3132	0.3129	0.3126
VB	0.3171	0.3173	0.3176	0.3166	0.3168	0.3165	0.3160	0.3159	0.3155
NN	0.3189	0.3180	0.3185	0.3162	0.3159	0.3159	0.3142	0.3131	0.3125

Table 5.20: RobustWSD: DT index per term MAP results (top3)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3106	0.3138	0.3145	0.3125	0.3078	0.3047	0.3023	0.2979	0.2949
JJ	0.3116	0.3113	0.3109	0.3102	0.3100	0.3088	0.3084	0.3079	0.3078
VB	0.3147	0.3156	0.3155	0.3157	0.3157	0.3153	0.3152	0.3152	0.3133
NN	0.3135	0.3171	0.3159	0.3167	0.3171	0.3156	0.3133	0.3119	0.3129

Table 5.21: RobustWSD: DT index per term MAP results (top5)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3076	0.3128	0.3091	0.3008	0.2965	0.2938	0.2887	0.2833	0.2792
JJ	0.3125	0.3115	0.3088	0.3044	0.3040	0.3033	0.3009	0.3004	0.2990
VB	0.3165	0.3183	0.3184	0.3185	0.3183	0.3181	0.3170	0.3158	0.3137
NN	0.3157	0.3148	0.3132	0.3126	0.3108	0.3079	0.3050	0.3006	0.2986

Table 5.22: RobustWSD: DT index per term MAP results (top10)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.2990	0.2961	0.2873	0.2732	0.2605	0.2532	0.2464	0.2397	0.2301
JJ	0.3126	0.3094	0.3051	0.3036	0.3014	0.3009	0.2975	0.2967	0.2958
VB	0.3114	0.3138	0.3137	0.3140	0.3139	0.3133	0.3129	0.3130	0.3127
NN	0.3072	0.3068	0.3035	0.2993	0.2967	0.2934	0.2905	0.2875	0.2859

As can be seen, setting the expansion fields weight to zero does not result in a value equal to the one obtained with the lemma field only in the baseline. This may be due to the different retrieval model that is applied with using index fields (*BM25F*). What is more, not even all the results are exactly equal having set the expansions field weight to zero. Theoretically, just the original terms of the document are considered dismissing the expansions field entirely.

Despite of that, there are small improvements in MAP using top1 to top5 expansions per term. Using even the top10 expansions per term, just VB experiments show a small increase. In general, setting the expansion fields weight to 0.2 performs best consistently. Additionally, using all part-of-speech (*ALL*) results in the biggest impact in the same way. That is due to the higher set of terms. Using a particular part-of-speech does not exhibit any advantages in the same manner. Only the verbs show a resembling improvement, despite of their words base small size. The results for the same experiments on the Tipster data set follow in Table 5.23, 5.24, 5.25 and 5.26. Again, the results that improve both the zero field weight value as well as the general baseline of 0.1876 are highlighted. For up to five expansions per term, small improvements in average precision could be observed.

Table 5.23: Tipster: DT index per term MAP results (top1)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1869	0.1915	0.1934	0.1928	0.1920	0.1900	0.1881	0.1861	0.1842
JJ	0.1874	0.1874	0.1874	0.1871	0.1868	0.1865	0.1861	0.1858	0.1853
VB	0.1893	0.1893	0.1892	0.1891	0.1888	0.1886	0.1883	0.1881	0.1876
NN	0.1858	0.1888	0.1906	0.1912	0.1903	0.1902	0.1892	0.1886	0.1882

Table 5.24: Tipster: DT index per term MAP results (top3)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1987	0.2045	0.2033	0.1992	0.1938	0.1890	0.1852	0.1810	0.1777
JJ	0.1875	0.1877	0.1874	0.1867	0.1852	0.1843	0.1836	0.1825	0.1815
VB	0.1976	0.1982	0.1984	0.1984	0.1985	0.1986	0.1984	0.1984	0.1979
NN	0.1834	0.1877	0.1888	0.1871	0.1854	0.1833	0.1818	0.1800	0.1787

Table 5.25: Tipster: DT index per term MAP results (top5)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1968	0.2017	0.1957	0.1888	0.1829	0.1777	0.1720	0.1677	0.1636
JJ	0.1872	0.1869	0.1858	0.1846	0.1829	0.1817	0.1807	0.1799	0.1793
VB	0.1994	0.2005	0.2008	0.2004	0.2001	0.1996	0.1992	0.1986	0.1981
NN	0.1790	0.1839	0.1836	0.1808	0.1785	0.1764	0.1740	0.1714	0.1693

Table 5.26: Tipster: DT index per term MAP results (top10)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1926	0.1986	0.1880	0.1761	0.1654	0.1570	0.1497	0.1434	0.1372
JJ	0.1871	0.1860	0.1845	0.1828	0.1815	0.1798	0.1783	0.1768	0.1756
VB	0.2008	0.2018	0.2020	0.2016	0.2008	0.1999	0.1989	0.1980	0.1970
NN	0.1720	0.1776	0.1755	0.1724	0.1696	0.1669	0.1643	0.1623	0.1594

Now, in Tables 5.27, 5.28, 5.29 and 5.30 the results of the same experiments with expansions added per document are displayed. There is practically no effect with less than 100 expansion terms per document. The reason for that is the documents size of about 500 words itself (refer to Table 4.1 on page 30 for a detailed overview). There is an insignificant positive effect on verbs and nouns, applying several hundred expansions per documents. Both effects can be confirmed with the second data set, as shown in tables 5.31, 5.32, 5.33 and 5.34.

5.5.2 TWSI

In this section the results are shown for applying the TWSI to the index documents. Before that, let us examine the quantity of the terms added by expansion to put the expansions impact into perspective. Table 5.35 shows the average number of tokens for each tag name in the entire collection. In average, about every fourth token in a document is expanded by the TWSI sense substituter.

Table 5.27: RobustWSD: DT index per doc MAP results (top10)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3172	0.3172	0.3173	0.3171	0.3172	0.3172	0.3170	0.3172	0.3171
JJ	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146
VB	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146	0.3146
NN	0.3146	0.3145	0.3146	0.3145	0.3145	0.3144	0.3145	0.3146	0.3146

Table 5.28: RobustWSD: DT index per doc MAP results (top50)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3174	0.3175	0.3157	0.3135	0.3134	0.3114	0.3114	0.3111	0.3107
JJ	0.3145	0.3145	0.3144	0.3144	0.3144	0.3144	0.3144	0.3144	0.3144
VB	0.3148	0.3147	0.3147	0.3146	0.3144	0.3144	0.3144	0.3144	0.3143
NN	0.3151	0.3151	0.3150	0.3150	0.3149	0.3132	0.3131	0.3130	0.3133

Table 5.29: RobustWSD: DT index per doc MAP results (top100)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3159	0.3150	0.3146	0.3120	0.3094	0.3065	0.3053	0.3039	0.3033
JJ	0.3138	0.3137	0.3137	0.3136	0.3136	0.3134	0.3133	0.3132	0.3132
VB	0.3150	0.3150	0.3150	0.3149	0.3148	0.3148	0.3151	0.3151	0.3150
NN	0.3146	0.3147	0.3139	0.3138	0.3120	0.3116	0.3110	0.3108	0.3100

Table 5.30: RobustWSD: DT index per doc MAP results (top500)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.3088	0.3080	0.3049	0.3001	0.2954	0.2918	0.2856	0.2788	0.2752
JJ	0.3105	0.3102	0.3098	0.3093	0.3064	0.3061	0.3056	0.3050	0.3044
VB	0.3163	0.3170	0.3166	0.3161	0.3159	0.3160	0.3159	0.3156	0.3160
NN	0.3152	0.3140	0.3132	0.3101	0.3085	0.3063	0.3048	0.3030	0.3015

Table 5.31: Tipster: DT index per doc MAP results (top10)

w POS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1863	0.1865	0.1865	0.1865	0.1864	0.1863	0.1861	0.1859	0.1859
JJ	0.1862	0.1862	0.1862	0.1862	0.1861	0.1862	0.1862	0.1862	0.1862
VB	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863
NN	0.1864	0.1865	0.1864	0.1864	0.1864	0.1864	0.1864	0.1864	0.1864

We apply the same field weights to the TWSI expanded terms. Table 5.36 shows the results for the topN (N = 1, 3, 5, 10) ranked expansions terms applied *per term* (i.e., per target term).

Since the documents are expanded by only a few terms (at least for *top1*), the results change very little. However, performance decreases as expected when the weighting of those additional terms increase. Adding the top10 substitutions per target term results in an expansions set being even bigger than the

Table 5.32: Tipster: DT index per doc MAP results (top50)

POS \ w	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1868	0.1873	0.1873	0.1871	0.1867	0.1862	0.1855	0.1847	0.1836
JJ	0.1862	0.1863	0.1863	0.1862	0.1862	0.1862	0.1862	0.1862	0.1861
VB	0.1869	0.1869	0.1869	0.1869	0.1869	0.1869	0.1869	0.1869	0.1869
NN	0.1863	0.1864	0.1864	0.1864	0.1861	0.1858	0.1855	0.1851	0.1849

Table 5.33: Tipster: DT index per doc MAP results (top100)

POS \ w	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1887	0.1892	0.1890	0.1884	0.1874	0.1865	0.1854	0.1840	0.1821
JJ	0.1863	0.1864	0.1863	0.1863	0.1862	0.1861	0.1860	0.1859	0.1858
VB	0.1877	0.1878	0.1878	0.1879	0.1879	0.1879	0.1878	0.1878	0.1878
NN	0.1872	0.1873	0.1872	0.1869	0.1863	0.1859	0.1853	0.1847	0.1843

Table 5.34: Tipster: DT index per doc MAP results (top500)

POS \ w	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
ALL	0.1979	0.1991	0.1967	0.1931	0.1871	0.1814	0.1754	0.1703	0.1654
JJ	0.1874	0.1875	0.1873	0.1869	0.1864	0.1858	0.1851	0.1844	0.1836
VB	0.1939	0.1939	0.1937	0.1937	0.1932	0.1927	0.1922	0.1915	0.1911
NN	0.1872	0.1888	0.1879	0.1867	0.1849	0.1824	0.1801	0.1781	0.1762

original terms set. That is why expansions hurt the performance in any field weight setting and decreases even faster when raising the field weight value. Compared to the in-line baseline (the weight of the expansion field set to zero) however, there are (except for top10) tiny increases for the weights of 0.1 to 0.3. This corresponds to index side DT experiments (expansions per term) and encourages taking an extra field weight (if any) of about 0.2.

Table 5.37 shows the results for the TWSI *per doc* expansions on the RobustWSD collection. Tables 5.38 and 5.39 show the results for the same experiments on the Tipster collection.

5.6 Data analysis

At this point, we want to conduct an analysis of the result displayed in the previous sections. The changes in the results at index expansion are still overall tiny, but they exhibit some positive movements. To explore these effects further, we have to take a deeper look at the ranks of two system runs. To do so, the ranked results of the baseline run are explored and will be compared to the run of the DT top3 per term index expansion results as an example expansion run (refer to table 5.20 on page 48). As the best performing setting, the expansion field's weight of 0.2 is chosen.

In statistics and visual analytics, different types of data are distinguished, e.g. *nominal variables* for unordered sets like movie titles, *quantitative variables* for numerical data and *ordinal data* for ordered sets. Since ranked document lists from information retrieval systems count as ordinal data, we can apply some rank correlation metrics to compare those. We start with the Spearman's rank correlation coefficient [Spearman, 1904], since it is appropriate for discrete variables as well. Spearman's correlation coefficient is a statistical measure of the strength of a *monotonic* relationship between paired data. In a sample it is denoted by r_s and is by design constrained as shown in Equation 5.1.

Table 5.35: Expansions quantities by tag name (TWSI)

tag name	~#tokens / doc
TOKEN	257.68
LEMMA	257.45
STEM	257.68
TWSI_TOP1_PERTERM	86.21
TWSI_TOP3_PERTERM	252.77
TWSI_TOP5_PERTERM	404.81
TWSI_TOP10_PERTERM	632.95
TWSI_10_PERDOC	9.99
TWSI_50_PERDOC	49.65
TWSI_100_PERDOC	97.13
TWSI_500_PERDOC	372.55

Table 5.36: RobustWSD: TWSI index per term MAP results

w top	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
top1	0.3150	0.3135	0.3137	0.3132	0.3118	0.3126	0.3126	0.3121	0.3117
top3	0.3200	0.3218	0.3212	0.3164	0.3176	0.3169	0.3162	0.3144	0.3142
top5	0.3200	0.3222	0.3211	0.3199	0.3167	0.3163	0.3171	0.3172	0.3184
top10	0.3201	0.3205	0.3217	0.3181	0.3140	0.3128	0.3137	0.3132	0.3127

Table 5.37: RobustWSD: TWSI index per doc MAP results

w top	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
top10	0.3140	0.3143	0.3141	0.3151	0.3154	0.3154	0.3157	0.3156	0.3158
top50	0.3143	0.3144	0.3147	0.3142	0.3145	0.3150	0.3150	0.3148	0.3145
top100	0.3157	0.3165	0.3160	0.3160	0.3157	0.3159	0.3160	0.3158	0.3153
top500	0.3195	0.3215	0.3190	0.3144	0.3121	0.3114	0.3104	0.3091	0.3081

Table 5.38: Tipster: TWSI index per term MAP results

w top	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
top1	0.1864	0.1878	0.1888	0.1892	0.1894	0.1893	0.1893	0.1892	0.1891
top3	0.1832	0.1854	0.1862	0.1863	0.1866	0.1861	0.1854	0.1848	0.1841
top5	0.1840	0.1878	0.1898	0.1904	0.1900	0.1897	0.1894	0.1889	0.1877
top10	0.1848	0.1887	0.1900	0.1905	0.1907	0.1904	0.1895	0.1886	0.1879

$$-1 \leq r_s \leq 1 \tag{5.1}$$

To use the Spearman rank correlation coefficient, the data is converted into ranks and the difference between the ranks of the two runs is calculated. Let $rank(x_i)$ be the rank of the first run, and $rank(y_i)$ that of the second run, then d_i is the difference between them as given in 5.2.

$$d_i = rank(x_i) - rank(y_i) \tag{5.2}$$

Table 5.39: Tipster: TWSI index per doc MAP results

w / top	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
top10	0.1866	0.1867	0.1868	0.1867	0.1865	0.1866	0.1866	0.1867	0.1867
top50	0.1864	0.1880	0.1888	0.1893	0.1896	0.1899	0.1899	0.1900	0.1902
top100	0.1875	0.1896	0.1909	0.1923	0.1927	0.1931	0.1933	0.1932	0.1933
top500	0.1868	0.1895	0.1907	0.1907	0.1905	0.1903	0.1894	0.1888	0.1877

The Spearman rank correlation r_s is then computed as shown in 5.3.

$$r_s = 1 - \frac{6 \sum_i {d_i}^2}{n \times (n^2 - 1)} \tag{5.3}$$

Note that the formulation given in 5.3 is a simplification of the Spearman rank correlation coefficient and only holds for the condition of all ranks being unique (i.e., there are no ties). This holds in practice for IR rankings.

For each query, the ordered documents collection from the Terrier output needs to be extracted and can then be given to the correlation function. Results are averaged over all queries. The result for the settings explained above (Baseline vs. DT index top 3 per term) is $r_s = 0.49663$. A value of 1.0 would be caused from two rankings being exactly equal and a value of 0.0 could be caused by an empty intersection of the two ranks document collection (i.e., no monotonic correlation). The correlation value our comparison results in is considered to be a moderate one. In the setting of IR rankings however, it means that there is a lot of change in ranks. That is, despite only tiny positive change in average precision, most of the rankings in the lists change. From this it follows that the expansions provided in that setting have a high impact on the retrieval.

To do the same with the Tipster data set, we chose to compare the baseline against the DT run with top3 expansions per term (of any part-of-speech, denoted as *ALL*) and a field weight of 0.1, since that configuration yielded the best relative improvement (refer to Table 5.24 on page 49) and hence seems worth investigating. Comparing the ranks as explained above produces a Spearman rank correlation coefficient of $r_s = 0.53276$, that is about the same as with the RobustWSD experiments. Likewise, the expansions have a high impact, even though a little bit less as the value is closer to 1.0. In order to get a better understanding of the changes and the differences between the two settings, we want to take a comparative look at the coverage of judgements in all the four rankings, namely the ones marked as relevant and the ones marked as non-relevant to the given query. Table 5.40 shows the coverage of judgments for both the baseline runs and the expansions experiments on both data sets, distinguished by the first three to all thousand ranks.

Table 5.40: Coverage of assessments in rank correlation experiments

	top3	top5	top10	top50	top100	top500	top1000
RobustWSD: Baseline	0.67	0.4	0.4	0.38	0.25	0.13	0.09
RobustWSD: Expansions	0.67	0.6	0.4	0.12	0.14	0.07	0.05
Tipster: Baseline	1.0	1.0	1.0	1.0	1.0	0.59	0.35
Tipster: Expansions	1.0	1.0	1.0	1.0	0.97	0.53	0.32

The clear discrepancy between the extent of assessments is noticeable, as is its decrease with dropping ranks. Second, the results the expansions runs return nearly always contain a higher proportion of non-judged documents (i.e., neither relevant nor irrelevant ones). Since those additional documents are of unknown relevance to the topic, no qualitative statement can be made about that change. The difference in coverage between the two data sets is tremendous, especially beginning from rank 100. While in the Tipster disks about every second document has an assessment up to the 500th rank, it is one out of ten in the RobustWSD collection. The differences within the collection results are very similar, though. From this follows that the extent of judgments a collection possesses is not a crucial feature for lexical expansions to retrieve more relevant documents on average. To get a more visual understanding of the changes between the two runs, Figures 5.4 and 5.5 show a query-by-query analysis of the changes between the two runs of both the comparisons discussed (x-axis: topic, y-axis: change in MAP).

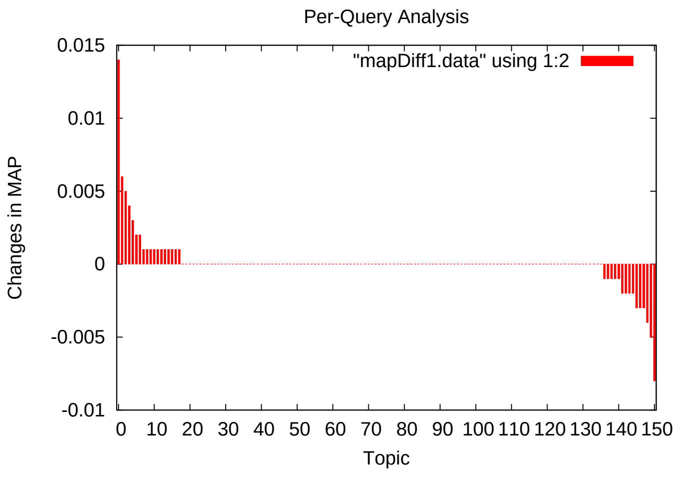


Figure 5.4: By-Query Analysis (RobustWSD)

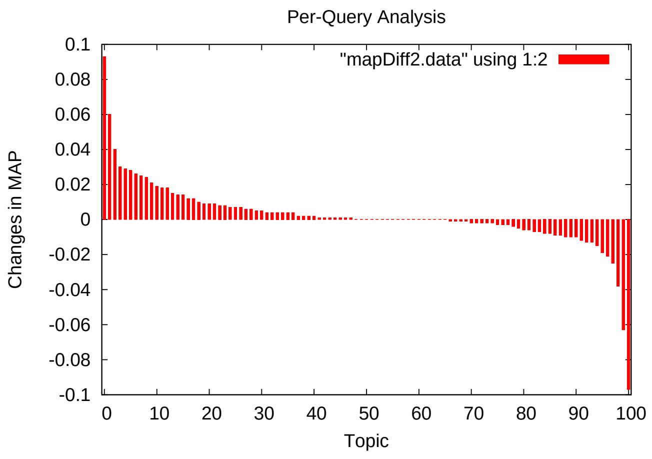


Figure 5.5: By-Query Analysis (Tipster)

The changes in MAP are depicted in descending order, starting with the query having the highest improvement. In general, about the same amount of topics show an increasing as a decreasing performance, compensating its overall benefits. This suggests a normal distribution, not making it possible to give a clear advise on the expansions used. In any case, the direct comparison of the two figures confirms the previously noted effect that the extent of assessments does not reveal new findings by using the second data set but rather reinforces those gained by the first one.

6 Discussion and future work

In this chapter, we will discuss the overall results shown so far throughout the paper. Furthermore, we will point to some ideas for future investigation in theoretical as well as practical respects.

6.1 Results discussion

Overall, this work aimed at measurably improving Information Retrieval by lexical expansion. Another popular approach is to analyze query logs to re-rank search results [Zhuang and Cucerzan, 2006], to suggest related queries [Baeza-Yates et al., 2004], to perform query expansion [Cui et al., 2003] or to get a better understanding of the user's query reformulation [Huang and Efthimiadis, 2009]. This, however, requires extensive search log data collected over a period of time. In a smaller laboratory setting we tried to improve IR on the lexical basis by enriching the user's information need and the documents representation. That lexical expansion was achieved by using three lexical resources that provide substitutions for words in context. Before going into any interpretation, we first want to summarize the experimental results found by this work. Some of the experiments could exhibit small improvements over the baseline in terms of average precision over all the queries. Table 6.1 gives an overview over the improvements and which could be confirmed by the second data set.

Table 6.1: Experimental result improvements

	RobustWSD	Tipster disks
DT query	X	X
DT index	✓	✓
TWSI query	✓	X
TWSI index	✓	✓
Delex query	X	X

The query expansion experiments with the DT had no positive effect on the first, and neither on the second. By contrast, its positive effect on index-side experiments could be confirmed by the Tipster experiments. In general, the effects are very small. There are some hints, however. Looking at all the results of all experiments, the learning effect can be divided into the following categories.

Query vs. index expansion
: The most remarkable result is the difference between query and index side expansion. Hardly any positive results could be found and confirmed conducting query expansion experiments. Since the query is very short and sensitive to changes, the more terms had been added, the worse the precision got. While on the other hand all index side ones continuously featured improvements. It is advised to further investigate this path in the future, i.e., to focus on index expansion research.

Part-of-speech
: Whether and how to use parts-of-speech (POS) depends on the resource. The DT is able to provide expansions for any POS contained in text. While the TWSI substitutions exclusively work for nouns, the Delex system depends on the four WordNet parts-of-speech (nouns, verbs, adjectives and adverbs). In most of the DT experiments divided by POS the ones using verbs and nouns stand out. If and only if nouns or verbs returned positive results in one setting (represented by one table in Chapter 5), then *ALL* (any POS) gained in terms of MAP as well, but never solely in that category. From this it follows that the *ALL* improvements might result from the effects of the other

POS categories. The ones using verbs or nouns continuously featured the greatest improvements. Further work should focus on those two lexical classes.

Resource
: Since the previous inspection suggests to expand nouns and verbs, TWSI seems to be an adequate fit. This is confirmed by all index side experiments using TWSI. Each setting could display many positive trends on both data sets. With the TWSI resource being limited to that particular lexical category, it can be utilized without any danger. The DT on the other hand is quite capable of covering the aspect of verbs. Since it even incorporates different tenses of verbs, it actually allows a more fine-grained study of that direction. That is why we recommend using the DT for future lexical expansion experiments as well.

Per term vs. per doc
: We employed the distinction between producing expansions *per term* and *per document/query* because of providing a clear and flexible limit in terms of quantity. In the end it was worth it because the results across all the experiments are quite indecisive about that distinction. Within the DT experiments, expansions per term resulted in more improvements in terms of quantity as well as quality. With the TWSI experiments however, it nearly is the other way around. The TWSI substitutions could even spud in yielding improvements with less expansions per document (50 or 100), while the DT experiments hardly produced any changes, and if so, used 100 or 500 expanded terms. Looking at the extent of changes in MAP however, adding and limiting expansions per document turned out to perform better continuously. Hence, it is suggested to follow that classification and to analyze different approaches of selecting terms per document.

Number of expansions
: Results on both the data sets are indecisive about the number of expansions per term, but there is a mean of about 3 to 5 additional words per term. The same discrepancy holds with limitations by document across both the collections. That is why we suggest to abandon the approach of strict quantitative limitations, but rather pursue an approach of adding the enhanced terms by a given similarity threshold of the resource itself. By that, only the associated terms of the strongest semantic relationship (e.g. synonyms) should be taken into account.

Field weights
: Applying expansions at index side experiments required using fields and setting field weights. Most of the useful results turned out to use a field weight between 0.1 and 0.5. For future investigations, field weights higher than that can be disregarded.

The experiments throughout this thesis incorporated several conditions (mentioned above). Overall, the analysis pointed out several aspects on which conditions to restrict, abandon or augment in the future. It is to be examined if the results improved once the signals are narrowed down and amplified. So far, however, the results are not significant. We applied a Student's t-test to the second comparison (Tipster baseline vs. DT index top3 expansions per term), since it is one of our best performing settings. The test results in a p-value of 0.52, that is, it is not nearly significant. That is why we recommend not to explore this approach of improving IR by lexical expansion, because it comes with considerable overhead.

6.2 Future work

In addition to the findings of the previous section, we want to divide the proposals for future work into the following categories.

Resources
: An avenue worth investigating might be the combined application of lexical expansion resources.

First, one could agree on just using the expansions provided by all resources to a particular target term, i.e. use the expansion term intersection. This way, a higher confidence about the expansions is gained. Second, the resources can be combined in such way that e.g. the DT is used as a fallback, if the most interesting lemma has no candidates in another resource.

Term selection
: Before expanding however, we advise to put time and thought into selecting terms for expansion. As pointed out in Chapter 5.4, there are target terms that are more *interesting* than others. We advise to apply measures of significance to the terms. At last, similarly to using the term intersection of different resource, the intersection of all target words may be discovered and given a higher weight.

Weighting
: In this thesis, only little focus was put on expansion term weighting. The query languages of modern search engine software allows for very flexible weighting schemes to be applied. Following the findings of this thesis about query expansion however, there was no hint that this is a promising avenue. At index side, using the different tags (headline, text) of the index documents can be considered to be used as different index fields with different weightings for them. Placing emphasis on the index documents, title can be considered as a small summary of its referring text and given a higher weight when matching query terms.

Pre-processing
: Another issue is the notion of so called multi-word expressions (mwe), which has been disregarded in all our experiments. Interesting terms for motivating that aspect are not only named entities like *El Nino*, but also expressions like *baby food*, *multi-billionaire* and *car industry*, which are one word terms in other languages like German. The idea of multi-word expressions is that they are small context units. Lexical expansion should be applied to those terms as a whole. Identifying those and making use of them might prevent too general expansions to be included.

Query investigation
: As Figures 5.4 and 5.5 on page 54 indicate, half of the queries are showing a slight increase in MAP performance. It remains to investigate, if general features can be learned from these queries that do. If the topics that do show an improvement could be identified and selected by a rule before expansion, only those should be expanded. It is possible that the expanded topics hurting performance exhibit characteristics by which their expansion could be prevented.

General
: Another approach of utilizing lexical expansion is to remove terms from the original query/document, if (by some methods discussed above) they don't appear in any expansions of the other ones, or reduce their term weights. Additionally, lexical expansion resources can be used for query reformulation to reflect the user not knowing the document collection and to adapt to its word distribution. The search engine could also passively suggest similar words or word nets for the user to choose from.

6.3 Perspective

When a user enters a search string into the input field of a search engine on the web, the system returns information it has pre-processed. In this work we tried to extend this information need (expressed by a query) of the user and the index documents at the lexical level to bridge the lexical mismatch problem between these two text representations. By doing so, working at the index side turned out to be a little bit more promising than query side. But overall the system returned about the same amount of relevant documents as it returned non-relevant ones. This indicates that the enhanced terms added by the resources are ambiguous and general as well, exhibiting the same language problems as the target terms do. The approaches did bridge the lexical gap to some extent, but they did also open further word relations that the user might not intend. What this achieved is a plus in quantity, but not necessarily in quality. So, the approach of improving the search process by adding additional terms as conducted in this thesis did not result in any significant changes. This indicates that the bottleneck in retrieval systems is not at the level of language, or that this approach is maxed out already. Applying lexical expansions or working at the language level in general comes with a considerable extent of work and testing to put into. More promising approaches have to be investigated instead, e.g. re-rankings by query logs or some kind of visual search.

List of Figures

List of Tables

Bibliography

[Agirre et al., 2009] Agirre, E., Nunzio, G. M. D., Mandl, T., and Otegi, A. (2009). CLEF 2009 Ad Hoc Track Overview: Robust-WSD Task.

[Amati, 2003] Amati, G. (2003). *Probability models for Information Retrieval based on Divergence from Randomness*. PhD thesis, University of Glasgow.

[Baeza-Yates et al., 2004] Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In *Proceedings of the 2004 international conference on Current Trends in Database Technology*, EDBT'04, pages 588–596, Berlin, Heidelberg. Springer-Verlag.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Bär et al., 2012] Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 435–440, Montreal, Canada.

[Biemann, 2012a] Biemann, C. (2012a). *Structure Discovery in Natural Language*. Springer, Berlin.

[Biemann, 2012b] Biemann, C. (2012b). Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *LREC*, pages 4038–4042.

[Biemann, 2013] Biemann, C. (2013). Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation: Special Issue on Collaboratively Constructed Language Resources*, 47(1):97–122.

[Biemann and Riedl, 2013] Biemann, C. and Riedl, M. (2013). Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1).

[Bordag, 2007] Bordag, S. (2007). *Elements of knowledge-free and unsupervised lexical acquisition*. PhD thesis, Universität Leipzig.

[Broder, 2002] Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, 36(2):3–10.

[Buckley and Voorhees, 2000] Buckley, C. and Voorhees, E. M. (2000). Evaluating evaluation measure stability. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 33–40, New York, NY, USA. ACM.

[Buckley and Voorhees, 2004] Buckley, C. and Voorhees, E. M. (2004). Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 25–32, New York, NY, USA. ACM.

[Büttcher et al., 2007] Büttcher, S., Clarke, C. L. A., Yeung, P. C. K., and Soboroff, I. (2007). Reliable information retrieval evaluation with incomplete and biased judgements. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 63–70, New York, NY, USA. ACM.

[Carpineto and Romano, 2012] Carpineto, C. and Romano, G. (2012). A Survey of Automatic Query Expansion in Information Retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50.

[Carr, 1999] Carr, P. (1999). *English Phonetics and Phonology: An Introduction*. Wiley.

[Cleverdon, 1997] Cleverdon, C. (1997). Readings in information retrieval. chapter The Cranfield tests on index language devices, pages 47–59. Morgan Kaufmann Publishers Inc., San Francisco.

[Crouch, 1990] Crouch, C. J. (1990). An approach to the automatic construction of global thesauri. *Inf. Process. Manage.*, 26(5):629–640.

[Cui et al., 2003] Cui, H., Wen, J.-R., Nie, J.-Y., and Ma, W.-Y. (2003). Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

[Delavenay, 1960] Delavenay, É. (1960). *An Introduction to Machine Translation*. Thames and Hudson.

[Ferrucci et al., 2006] Ferrucci, D., Lally, A., Gruhl1, D., Epstein, E., Schor, M., Murdock, J. W., Frenkiel, A., and Brown, E. W. (2006). Towards an Interoperability Standard for Text and Multi-Modal Analytics. Technical report.

[Finlayson, 2013] Finlayson, M. A. (2013). *MIT Java Wordnet Interface (JWI) User's Guide*. MIT.

[Firth, 1957] Firth, J. R. (1957). A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32.

[Furnas et al., 1983] Furnas, G., Landauer, T., Gomez, L., and Dumais, T. (1983). Statistical semantics: Analysis of the Potential Performance of Keyword Information Systems. *Bell System Technical Journal*, 62(6):1753–1806.

[Furnas et al., 1987] Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971.

[Gottron, 2010] Gottron, T. (Sommersemester 2010). Information Retrieval. Vorlesungsskript.

[Gurevych et al., 2007] Gurevych, I., Mühlhäuser, M., Müller, C., Steimle, J., Weimer, M., and Zesch, T. (2007). Darmstadt Knowledge Processing Repository Based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology*, Tübingen, Germany.

[Harris, 1954] Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

[He and Ounis, 2006] He, B. and Ounis, I. (2006). Query performance prediction. *Inf. Syst.*, 31(7):585–594.

[Huang and Efthimiadis, 2009] Huang, J. and Efthimiadis, E. N. (2009). Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 77–86, New York, NY, USA. ACM.

[Jurafsky and Martin, 2009] Jurafsky, D. and Martin, J. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall.

[Kelly and Stone, 1975] Kelly, E. F. and Stone, P. J. (1975). *Computer Recognition of English Word Senses*. North-Holland, Amsterdam.

[Kishida, 2005] Kishida, K. (2005). *Property of Average Precision and Its Generalization: An Examination of Evaluation Indicator for Information Retrieval Experiments*. NII technical report: Kokuritsu-Jōhōgaku-Kenkyūsho. National Institute of Informatics.

[Krovetz and Croft, 1992] Krovetz, R. and Croft, W. B. (1992). Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.*, 10(2):115–141.

[Lin, 1998] Lin, D. (1998). Automatic Retrieval and Clustering of Similar Words. In *COLING-ACL*, pages 768–774.

[Lin and Dyer, 2010] Lin, J. and Dyer, C. (2010). *Data-intensive Text Processing with MapReduce*. Synthesis lectures on human language technologies. Morgan & Claypool.

[Luhn, 1958] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

[Manning and Schütze, 1999] Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, 1 edition.

[Marcus et al., 1993] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330.

[Marneffe et al., 2006] Marneffe, M. D., Maccartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *In LREC 2006*.

[McCarthy and Navigli, 2009] McCarthy, D. and Navigli, R. (2009). The English lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

[Metzler and Croft, 2005] Metzler, D. and Croft, W. B. (2005). A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 472–479, New York, NY, USA. ACM.

[Metzler and Croft, 2007] Metzler, D. and Croft, W. B. (2007). Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 311–318, New York, NY, USA. ACM.

[Miller, 1995] Miller, G. A. (1995). WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41.

[Miller et al., 2012] Miller, T., Biemann, C., Zesch, T., and Gurevych, I. (2012). Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1781–1796.

[Na and Ng, 2011] Na, S.-H. and Ng, H. T. (2011). Enriching document representation via translation for improved monolingual information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 853–862, New York, NY, USA. ACM.

[Nadkarni et al., 2011] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

[Navigli, 2009] Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69.

[Ogren and Bethard, 2009] Ogren, P. V. and Bethard, S. J. (2009). Building test suites for UIMA components. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '09, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ogren et al., 2008] Ogren, P. V., Wetzler, P. G., and Bethard, S. (2008). ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*.

[Ounis et al., 2006] Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., and Lioma, C. (2006). Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*.

[Pérez-Agüera and Zaragoza, 2008] Pérez-Agüera, J. R. and Zaragoza, H. (2008). UCM-Y!R at CLEF 2008 Robust and WSD tasks. In *CLEF 2008 Workshop, 17-19 September*, Lecture Notes in Computer Science. Springer, Aarhus, Denmark.

[Robertson et al., 1996] Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. (1996). Okapi at TREC-3. pages 109–126.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18:613–620.

[Sanderson, 1994] Sanderson, M. (1994). Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 142–151, New York, NY, USA. Springer-Verlag New York, Inc.

[Sanderson, 2000] Sanderson, M. (2000). Retrieving with Good Sense. *Inf. Retr.*, 2(1):49–69.

[Schmid, 1994] Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, United Kingdom.

[Spearman, 1904] Spearman, C. (1904). The proof and measurement of association between two things. *The American journal of psychology*, 15:72–101.

[St. Charles, 2012] St. Charles, W. (2012). A Comparison of Lexical Expansion Methodologies to Improve Medical Question and Answering Systems. Master's thesis, University of Colorado at Boulder.

[Szarvas et al., 2013] Szarvas, G., Biemann, C., and Gurevych, I. (2013). Supervised All-Words Lexical Substitution using Delexicalized Features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*. To appear.

[Taghavi et al., 2012] Taghavi, M., Patel, A., Schmidt, N., Wills, C., and Tew, Y. (2012). An analysis of web proxy logs with query distribution pattern approach for search engines. *Comput. Stand. Interfaces*, 34(1):162–170.

[UIMA Development Community, 2006] UIMA Development Community (2006). *UIMA Overview & SDK Setup*. Apache Software Foundation.

[Voorhees and Harman, 2000] Voorhees, E. M. and Harman, D. (2000). Overview of the Eighth Text REtrieval Conference (TREC-8). pages 1–24.

[Weaver, 1955] Weaver, W. (1955). Translation. In Locke, W. N. and Boothe, A. D., editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA. Reprinted from a memorandum written by Weaver in 1949.

[Wolf et al., 2010] Wolf, E., Bernhard, D., and Gurevych, I. (2010). Combining probabilistic and translation-based models for information retrieval based on word sense annotations. In et al. (Eds.), C. P., editor, *CLEF 2009 Workshop, Part I*, volume 6241 of *Lecture Notes in Computer Science*, pages 120–127. Springer, Berlin / Heidelberg.

[Yarowsky, 1993] Yarowsky, D. (1993). One Sense per Collection. In *ARPA Human Language technology. Workshop*.

[Zaragoza et al., 2004] Zaragoza, H., Craswell, N., Taylor, M., Saria, S., and Robertson, S. (2004). Microsoft Cambridge at TREC 13: Web and HARD tracks. In *Text REtrieval Conference (TREC-13)*.

[Zhai and Lafferty, 2001] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 334–342, New York, NY, USA. ACM.

[Zhong and Ng, 2012] Zhong, Z. and Ng, H. T. (2012). Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 273–282, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zhuang and Cucerzan, 2006] Zhuang, Z. and Cucerzan, S. (2006). Re-ranking search results using query logs. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 860–861, New York, NY, USA. ACM.