Statistical Language Models and Information Retrieval: natural language processing *really* meets retrieval [*]

Djoerd Hiemstra and **Franciska de Jong**
University of Twente,
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands
{hiemstra, fdejong}@cs.utwente.nl

Abstract

Traditionally, natural language processing techniques for information retrieval have always been studied outside the framework of formal models of information retrieval. In this article, we introduce a new formal model of information retrieval based on the application of statistical language models. Simple natural language processing techniques that are often used for information retrieval – we give an introductory overview of these techniques in Section 2 – can be modeled by the new language modeling approach.

1 Introduction

Full-text information retrieval is all about natural language understanding. Users of for instance web search engines enter some vague ambiguous statement of what they are looking for, and it is the search engine's task to return a list of links to documents that are relevant to the user's request. Simply returning an unordered list of documents that contain the words the user entered is insufficient. On any very large document collection like the world wide web, many thousands of documents might contain the words the user has entered, but only a few of those documents will actually be relevant to the user. The typical user of a web search engine will only read the first ten or twenty documents of the thousands of documents retrieved anyway. It is therefore important that the system not only retrieves documents, but also ranks the retrieved documents in decreasing order of estimated degree of relevance to the user. Successful ranking algorithms use word statistics to 'understand' which documents are probably relevant to the user, and which documents are less relevant.

Information retrieval systems that *only* use word statistics treat words like "milk", "cow", "cows", "cattle", etc., as if they are totally unrelated. It seems obvious that much can be gained by using some simple linguistic knowledge during document indexing and/or query processing. Simple natural language processing tools might be used to detect relations between words, for instance that "milk cow" is a phrase, that "cows" is the plural of "cow", and that "cattle" is a possible synonym of "cows". It is however far from trivial to incorporate the simple type of linguistic knowledge into the information retrieval models that use word statistics.

The so-called Boolean model of information retrieval is an example of a model that does not use word statistics and does not rank the documents. A Boolean retrieval system is designed to retrieve an unordered list of documents that contain the precise combination of words included in the query. When two query terms are related by an AND connective, both terms must be present in the documents. When two query terms are related by the OR connective,

[*]This article was published in Glot International 5(8), 2001, pages 288-294, http://www.glotinternational.com

only one of the terms (or both) have to be present in the documents. Note that the OR connective is a natural choice if two words are used as synonyms, for instance as in `cows OR cattle`. Modern Boolean retrieval systems often offer additional operators to the standard operators AND, OR and NOT, like the ADJ operator which matches any document that contains two adjacent terms, and the NEAR operator which matches documents which contain the two terms within a certain window of terms. So, the query `milk ADJ cow` might be used to find documents containing the phrase "milk cow".

Many models of ranked retrieval have been proposed that try to tackle the Boolean model's inability rank documents (see e.g. Baeza-Yates and Ribeiro-Neto 1999 for an overview). However, none of these models adequately answer the question how to incorporate the simple type of linguistic knowledge, because they use word statistics in a rather ad-hoc way: the so-called *tf·idf* weights. These weights use a product of the term frequency *tf* and the inverse document frequency *idf*. The *tf* component is related to the number of times a term occurs in a document, whereas the *idf* component is inversely related to the number of documents in which the term occurs. Salton and Buckley (1988) report experiments with a total of 1,800 different variations of *tf·idf* weights, and many more variations have been suggested since.

The use of statistical language models for information retrieval was recently proposed by Hiemstra (1998), Miller et al. (1999) and by Ponte and Croft (1998). These models do not rely on *tf·idf* weighting. Instead, they use simple, easy to understand, probability measures of the form: "If the word occurs three times in a document that contains 100 words in total, then the probability of that word given the document is 0.03."; or "if the word occurs 2,000 times in a corpus of a million words, then the probability of that word in the English language (assuming that we are searching for English documents) is 0.002". A linear combination of these two probability measures results in a language model that behaves like the *tf·idf* weights, outperforming the best-performing *tf·idf* variations (Hiemstra 2000). This opens the way for a well-founded combination of statistics and linguistic knowledge for information retrieval: More expressive language models are easily constructed, for instance by using n-gram models, or by combining the simple language model with a statistical translation model.

This article is organized as follows. In Section 2, we give an overview of natural language processing techniques for information retrieval. In Section 3, we introduce the use of statistical language models for information retrieval. Finally, in Section 4, we conclude this paper by looking into the future of natural language search and information retrieval.

2 Natural language processing and retrieval: An overview

Natural language processing (NLP) techniques are often applied to enable users to enter a natural language request, without bothering them with formalisms such as the Boolean connectives. The NLP techniques presented in this section result in a representation of documents and user requests that is closer to the actual meaning of the text, ignoring as many of the irregularities of natural language as possible.

2.1 NLP techniques commonly applied

A typical approach to document indexing and query processing is the following. First a tokenization process takes place, then stop words are removed, and finally the remaining words are stemmed (what these terms mean will be explained below). Additionally, natural language processing techniques might identify phrases or split compounds. Figure 1 shows an example text that will be used to illustrate the typical approach to document indexing and query processing.

Tokenization

As a first step in processing a document or a query, it has to be determined what the processing tokens are. One of the most simple approaches to tokenization defines word symbols and inter-word symbols.

CHAPTER 1, PREAMBLE

1.1. Humanity stands at a defining moment in history. We are confronted with a perpetuation of disparities between and within nations, a worsening of poverty, hunger, ill health and illiteracy, and the continuing deterioration of the ecosystems on which we depend for our well-being.

Figure 1: Example text: opening lines of Agenda 21

In the example of Figure 2 all characters that are no letter and no digit are considered to be inter-word symbols. The inter-word symbols are ignored during this phase, and the remaining sequences of word symbols are the processing tokens. As a result it is not possible to search for punctuation marks like for instance hyphens and question marks.

```
chapter 1 preamble 1 1 humanity stands at a
defining moment in history we are confronted
with a perpetuation of disparities between and
within nations a worsening of poverty hunger
ill health and illiteracy and the continuing
deterioration of the ecosystems on which we
depend for our well being
```

Figure 2: The Agenda 21 text after tokenization

In the example, mark-up information is also ignored, but this information might be kept to search for e.g. title words. Additionally, heuristics might be used to identify sentences, or the fact that "1.1" should be kept as one processing token.

Stop word removal

Stop words are words with little meaning that are removed from the index and the query. Words might carry little meaning from a frequency (or information theoretic) point of view, or alternatively from a conceptual (or linguistic) point of view. Words that occur in many of the documents in the collection carry little meaning from a frequency point of view, because a search for documents that contain that word will retrieve many of the documents in the collection. By removing the very frequent words, the document rankings will not be affected that much. Stop word removal on the basis of frequency can be done easily by removing the 200-300 words with the highest frequencies in the document collection. As a result of stopping the very frequent words, indexes will be between 30 % and 50 % smaller.

If words carry little conceptual meaning, they might be removed whether their frequency in the collection is high or low. In fact, they should especially be removed if their frequency is low, because these words affect document rankings the most. Removing stop words for conceptual reasons can be done by using a stop list that enumerates all words with little meaning, typically function words like for instance "the", "it" and "a". These words also have a high frequency in English, but most publicly available stop lists are, at least partly, not constructed on the basis of word frequencies alone. For instance the stop list published by Van Rijsbergen (1979), contains infrequent words like "hereupon" and "whereafter", which occur respectively two and four times in the document collection that is provided by the Text Retrieval Conference (TREC).

```
chapter 1 preamble 1 1 humanity stands defining
moment history confronted perpetuation
disparities nations worsening poverty hunger
ill health illiteracy continuing deterioration
ecosystems depend well being
```

Figure 3: The Agenda 21 text after stop word removal

In Section 3, stop words are defined mathematically by assigning zero probability to one of the model's parameters. The mathematical definition does not conflict with the linguistically motivated definition of stop words.

Morphological normalization

Morphological normalization of words in documents and queries is used to find documents that contain morphological variants of the original query. Morphological normalization can be achieved either by using a stemmer or by using dictionary lookup.

A stemmer applies morphological 'rules of the thumb' to normalize words. Stemmers were already developed in the 1960's when the first retrieval systems were implemented. Well known stemmers are those by Lovins (1968) and Porter (1980), the last one being the most commonly accepted algorithm. As reported by Harman (1991) for English and Kraaij and Pohlmann (1996) for Dutch, the effect on retrieval performance is limited. Stemming tends to help as many queries as it hurts. Sometimes stemming algorithms may conflate two words with very different meanings to the same stem, for instance the words "skies" and "ski" might both be reduced to "ski". In such cases users might not understand why a certain document is retrieved and may begin to question the integrity of the system in general (Kowalski 1997). Still, stemmers are used in many research systems like Smart, Okapi and Twenty-One. Figure 4 gives the results of Porter's algorithm, which does not always result in linguistically correct stems.

```
chapter 1 preambl 1 1 human stand defin moment
histori confront perpetu dispar nation worsen
poverti hunger ill health illiteraci continu
deterior ecosystem depend well be
```

Figure 4: The Agenda 21 text after stemming

Linguistically correct output can be generated by dictionary lookup. Having a full-form dictionary is however not enough to build a lemmatizer. Some words will have multiple entries, possibly with different lemmas. For instance, the word "saw" may be a past tense verb, in which case its lemma is "see" and it may be a noun, in which case its lemma is equal to the full form. Similarly, the word "number" may be the comparative of "numb". For these cases, a lemmatizer has to determine the word's part-of-speech before the correct lemma can be chosen. Statistical algorithms trained on (partially) hand-tagged corpora may be used to effectively find the correct part-of-speech and therefore the correct lemma.

Instead of stemming or lemmatizing both the documents and the query, the system might as well generate morphological variants of all query terms and leave the documents as they are. In Section 3, stemming and lemmatizing will be approached by this point of view.

Phrase extraction and parsing

During document indexing and query processing, multiple words may be treated as one processing token. The meaning of phrases might be quite different from the meaning of the separate words. A user who enters the query "stock exchange" will probably not be satisfied with documents that discuss "exchange of live stock". There are three basic approaches to phrase extraction. Phrases might be simply predefined (Robertson and Walker 2000), extracted by statistical co-occurrence (Mitra et al. 1997) or extracted by syntactic processing (Strzalkowski 1995). Phrase extraction based on statistical co-occurrence may use very simple methods, e.g. the identification of all pairs of non stop words that occur contiguously in at least x documents. Syntactic processing might be used to extract noun phrases which are then normalized to head-modifier pairs. This will produce the same processing token for e.g. "information retrieval" and "retrieval of information", because in both "information" modifies the head "retrieval". Statistical and syntactic techniques for phrase extraction were compared by Mitra et al. (1997) for English and Kraaij and Pohlmann (1998) for Dutch. Both evaluations show that phrase extraction, like stemming, does not improve retrieval effectiveness significantly.

The methods just mentioned use both the phrase and the single words in the index, which might be problematic from a theoretical viewpoint. The phrase and its single words are obviously related, because the occurrence of the phrase implies the occurrence of its single words. The application of ranking algorithms that use term independence might therefore no longer by justified. This complication is not addressed by the publications mentioned above, but in fact, the obvious violation of the independence assumption might be one of the reasons for the disappointing results on retrieval performance. In Section 3 a bigram model will be introduced that explicitly models the dependence relation between words in phrases.

Compound splitting

During indexing or query formulation, some words might be treated as more than one processing token. A compound word is a single orthographic unit that consists of two or more single words, like for instance “airport” and “wildlife”. Compound words are especially an issue in languages that allow almost unrestricted compounding like Dutch and German. In Dutch, for instance the noun phrase “potable water supply” would be one compounded word: “drinkwatervoorziening”. A known problem with morphological parsers that split compounds is that they might accidentally split proper names and other words that are not listed in the dictionary, for instance “Bangkok” is not the composition of the Dutch words “bang” and “kok”. Kraaij and Pohlmann (1998) show that the splitting of compounds improves retrieval performance significantly for Dutch. Similar to phrases, both the compound and its components can be used during searching, but the use of a retrieval model that assumes the independence between terms might not be appropriate.

Synonym normalization

Much like stemming and lemmatization, synonymous words might also be conflated to one processing token during indexing and automatic query formulation. For instance in Okapi, closely related or synonymous terms like “CIA” and “Central Intelligence Agency” are conflated (Robertson and Walker 2000). In Inquery special processing tokens like #CITY and #COMPANY are added for respectively every mention of a U.S. city or company (Broglio et al. 1994).

2.2 Natural language search in commercial systems

Many commercial natural language search systems provide operators that replace the traditional Boolean set operators AND, OR and NOT by related operators that are more easy to understand by non-expert users. The use of these operators is however not mandatory, making it possible to enter a natural language request as shown in Figure 5. The operators, which can be found in for instance Dialog Target, Lexis-Nexis FreeStyle, or Altavista are summed up in the following paragraphs. In these systems, the actual tokens used for these operators might differ from the ones used in the examples.

```
how to promote sustainable
  consumption patterns
```

Figure 5: Simple natural language request: rank the documents containing one or more of the terms.

Exact match operator / mandatory terms

The mandatory term operator can be used to indicate that a term *must* be present in the selected documents. It is inspired by the AND operator in Boolean queries, but has slightly different semantics. Unlike the AND operator, which is a binary operator requiring two arguments, the mandatory term operator is a unary operator requiring one argument. The example presented in Figure 6 uses the plus symbol to flag mandatory terms, but other conventions are also used, e.g. using the asterisk-character, or using a separate user interface field.

Exclusion operator

The exclusion operator can be used to indicate that a term should *not* be present in the selected documents. Obviously it is inspired by the NOT operator in Boolean queries. This operator is not as common as the exact match operator, because the absence of a term is not as clear an indication of relevance as the presence of a term. In Figure 6 the minus symbol is used to flag terms that documents should not contain.

Phrases

Explicit marking of phrases is inspired by the 'Boolean' ADJ operator. The system uses the phrase to produce a better ranking. Identifying phrases is very useful in combination with the exact match operator to perform a high-precision search, looking for an exact phrase or an exact quotation. Most query languages use single or double quotation marks to mark phrases.

```
how to reduce the production of
  +"harmful materials" -uranium
```

Figure 6: Natural language request: rank the documents considering that documents should contain the phrase "harmful materials" but not the term "uranium"

Synonyms

Operators for synonyms are inspired by the Boolean OR. Explicit marking of synonyms is sometimes supported by putting synonyms between parenthesis. The system uses this information to produce a better ranking. Other conventions leave the main term outside the parenthesis as in `child (minor, infant)`.

Manual term weighting

Query term weights are, one way or the other, used in many ranking algorithms. Some systems give the user access to these weights so they can indicate themselves which terms are important and which terms are not important. Figure 7 gives an example of this use of term weights.

```
why (forbid prohibit ban) wasteful
  packaging[0.9] of products[0.1]
```

Figure 7: Natural language request: rank the documents considering that the terms "forbid", "prohibit" and "ban" are synonyms and considering that "packaging" is much more important than "products"

2.3 Discussion

The overview given in this section presents practical natural language processing techniques that have proven themselves useful for information retrieval, like stemming, stopping and identifying phrases or synonyms. Their use, however, is not very well justified by the formal models of information retrieval. Open questions related to their use are for instance: What is the formal justification of stopping infrequent words which have a high $tf \cdot idf$ weight? What is the formal relation between a stemmed index and an index of the full-form words? How can phrases be incorporated in a formal model that assumes term independence? How do we model mandatory query terms?

These questions can be answered by the new statistical language modeling approach presented in the next section.

3 Statistical language models

As said, we will introduce in this section the concept of statistical language modeling in some more detail. The introduction of statistical language models in the field of information retrieval can be seen as an attempt to provide the ad hoc and rather ill understood role of NLP in information retrieval with a theoretically sound foundation that can account for the effects of the kind of techniques described in Section 2.

3.1 A short history

Statistical language models have been around for quite a long time. They were first applied by Andrei Markov at the beginning of the 20th century to model letter sequences in works of Russian literature (Manning and Schütze 1999). Another famous application of language models are Claude Shannon's models of letter sequences and word sequences, which he used to illustrate the implications of coding and information theory (Shannon 1948). Later, statistical language models were developed as a general natural language processing tool. Language models were first successfully applied to automatic speech recognition at the end of the 1970's. The by now standard model of automatic speech recognition consists of two parts. The first part is the language model, that predicts the next word in continuous speech. The second part models the acoustic signal and is therefore called the acoustic model. The theory behind the speech recognition models is part of hidden Markov model theory (indeed, a 'hidden' version of Markov's models) that was developed by Leonard Baum and his colleagues at IBM in the late 1960's and early 1970's (Rabiner 1990; Jelinek 1997). In the 1990's, statistical language models were applied to many other areas of natural language processing, like for instance machine translation (Brown et al. 1990) and part-of-speech tagging (Cutting et al. 1992).

3.2 The basics

A statistical language model assigns a probability to sequences of words. It does not distinguish between wellformed and unwellformed sequences: any sequence of words is considered, but some sequences are much more probable than others. For information retrieval, a language model is defined for each separate document in the collection, modeling the typical language use of that particular document. A language model of this GLOT article should for instance assign a high probability to the phrase "natural language information retrieval", but a much lower probability to the phrase "you are invited to my birthday party". Using the language model, the system is able to decide that this article is a good candidate for retrieval if the user enters a request for documents about "information retrieval", but a bad candidate if the user searches for "birthday party".

We use the following notation: $P(D)$ is the probability of the the event "the document D is relevant", where D is a random variable that can be any document in the collection; $P(T)$ is the probability of the term T in general (natural) language, where T can be any word in the language; $P(T|D)$ is the probability of the term T in the relevant document's language. It is assumed that when users enter a query, some words in the query are important, while others are unimportant, that is, some words come from the general (English) vocabulary, while others come specifically from the vocabulary of the relevant document. Because it can not be known beforehand which are the important words and which are the unimportant ones, the model uses a mixture of the probability $P(T)$ of a term in general English, and the probability $P(T|D)$ of a term in the relevant document. The unknown parameter λ $(0 \leq \lambda \leq 1)$, which is the probability that a term is important, determines the mixture. For the basic model we assume independence between query terms in a sequence of n terms $T_1, T_2, \cdots, T_n$, resulting in the following definition of our basic model.

$$P(T_1,\cdots,T_n|D) \;=\; \prod_{i=1}^{n}((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D))$$

In an IR application documents are ranked in decreasing order of this probability. Experimental studies show that the basic model introduced above outperforms ranked retrieval models that use today's best-performing *tf* · *idf* weighting algorithms (Hiemstra 2001). It can also be shown, by proper definition of the probability measures and by applying order-preserving transformations, that the above model has an equivalent *tf*·*idf*-like weighting algorithm that produces the exact same results (Hiemstra 2000).

3.3 Modeling stop words and mandatory terms

Because it is unknown which terms are important and which terms are unimportant, the unknown parameter λ_i is set to some constant value. Experimental studies show that an optimum value for λ_i lies around 0.15 and 0.3 (Hiemstra 1998; Miller et al. 1999). However, simple natural language processing techniques might be used to distinguish between important and unimportant words in some more clever way. One of these approaches was introduced in section 2: the use of a stop list. The stop list contains all words that are generally unimportant. Each word T_i that is listed in the stop list might be assigned $\lambda_i = 0$. Identifying the important words might not be that simple, but we might for instance assume that nouns are generally more important than adjectives, or that the head of a phrase is more important than its modifiers.

It is easy to verify that for $\lambda_i = 0$, the term T_i, whether it occurs in the document D or not, contributes the same amount of probability to the final result of the computation. Therefore, if $\lambda_i = 0$, the term T_i does not affect the final ranking of the documents. Indeed, it might as well be removed from the user request, as is done with stop words. The model mathematically explains why terms might not contribute to the ranking of the documents, although they have non-zero $tf \cdot idf$ weights.

It is also easy to verify that for $\lambda_i = 1$, the probability of the final result will be 0 if, and only if, the term T_i does not occur in the document D. Therefore, only those documents are retrieved in which the term occurs: the term is mandatory in the retrieved list of documents.

To our knowledge, none of the other existing models of ranked retrieval can mathematically explain or justify the use of stop words and/or mandatory terms, but they play a role in many practical information retrieval systems. A framework for information retrieval based on language models can justify these common approaches, and therefore can be considered as contributing to our understanding of information retrieval in general.

3.4 Modeling simple phrases

For the basic model we assumed that words are independent because it is well-known that retrieval systems work fine if we completely ignore the (syntactic) structure of natural language. For applications other than retrieval, statistical language models often predict the next word given some history of words by using n-gram models. A bigram model (n-gram model with $n = 2$) uses bigram probabilities $P(T_i|T_{i-1})$: the probability of a word T_i given the previous word T_{i-1}. We expect the probability of $P(T_i = \texttt{exchange}|T_{i-1} = \texttt{stock}, D)$ to be much higher in documents about Wall Street than in documents about the "exchange of live stock", thereby giving the system the possibility to distinguish between the two. Simple two-word phrases, to be identified by a syntactic parser, can be modeled by incorporating bigram probabilities into the probability measure.

3.5 Modeling translation, synonyms, stemming, and more

The general problem that natural language processing tools like stemmers want to solve is the following. Often, users that enter a perfectly reasonable request, will get disappointing results because the vocabulary they use differs from the vocabulary of (some of the) relevant documents, which as a consequence will not be returned in the search result. The user might for instance enter plural nouns (requesting documents about "cows" or "cats"), whereas documents more often use singular nouns. Similarly, users might use words that are synonymous or closely related with the words that are used in the documents ("cows" vs. "cattle"). An extreme example of such a case is the situation where the user wants to do a so-called cross-language search: Using French queries on an English database.

To model these cases, we will combine a statistical translation model using probabilities $P(S_i|T_i)$ with the basic language model introduced above. The random variable S_i is used to denote the ith word in the user's request. It can be any word in the vocabulary of the user. The random variable T_i now can be any word in the vocabulary of the documents. The translation model and the basic language model can easily be combined in a way that is very similar to way the acoustic model and the language model in speech recognition are combined (Hiemstra and De Jong 1999).

In practice, the statistical translation model will be used as follows. A natural language processing tool will convert the request $S_1, S_2, \cdots, S_n$ into a possibly ambiguous representation of this request by listing several possible terms $T_i = t$ for each $S_i = s$. This might for instance be done by a machine translation tool that lists pairs (s, t), together with their translation probability. For each word s in this list there will be one or more possible translations t.

Let's assume the user enters the French request "déchets dangereux" to search an English document collection. Possible translations of "déchets" might be "waste", "litter" or "garbage", possible translations of "dangereux" might be "dangerous" or "hazardous". In practice, this type of translation takes place during the processing of the user request, resulting in a structured query like the one displayed below. The structured query (which is similar to the so-called word lattice in automatic speech recognition systems) is matched against each document in the collection.

$((\texttt{waste} \cup \texttt{litter} \cup \texttt{garbage}),\ (\texttt{dangerous} \cup \texttt{hazardous}))$

Instead of translation from e.g. French to English, a natural language processing tool might use any other conversion algorithm to generate for each word a set of variants: e.g. possible synonyms of a word, words that are phonetically similar, identify spelling mistakes and suggest corrections, etc., etc. These variants can be structured in the same way as the possible translations in the cross-language search example. All instances of natural language processing tools that yield alternative representations for words can thus be integrated in a retrieval system in a mathematically sound way using the language modeling approach.

The use of the traditional stemmer is a special case: The stemmer is used on-line (when the user enters the request), but also off-line during indexing of the documents. For the language modeling approach, however, it can be proven that on-line morphological generation produces the exact same results as the traditional use of a stemmer, if the generated morphological variants are the same words that are conflated by the stemmer.

4 Conclusion

The statistical language models for information retrieval provide a complete mathematical theory of information retrieval, covering topics like stop words, mandatory words, phrases, stemming and translation. The approach suggests new ways to apply natural language processing tools to information retrieval, and opens the way to new retrieval applications. Many of the information retrieval applications of the near future call for serious NLP, for instance applications that support cross-language information retrieval, question answering from unstructured texts, or automatic (multi-)document summarization.

A detailed description of the model and experimental results of several prototype systems on information retrieval text corpora are reported in the first author's Ph.D. thesis (Hiemstra 2001).

References

Baeza-Yates, R.A. and B. Ribeiro-Neto (1999). *Modern Information Retrieval.* Addison-Wesley, London.

Broglio, J., J.P. Callan, and W.B. Croft (1994). Inquery system overview. In D. Penrose (Ed.), *Proceedings of the TIPSTER Text Program (Phase I)*, Morgan Kaufmann, San Fransisco, pp. 40–48.

Brown, P.F., J.C. Cocke, S.A.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin (1990). A statistical approach to machine translation. *Computational Linguistics 16*(2), 79–85. Association for Computational Linguistics, Morristown, New Jersey.

Cutting, D., J. Kupiec, J. Pedersen, and P. Sibun (1992). A practical part-of-speech tagger. In M. Bates and O. Stock (Eds.) *Proceedings of Applied Natural Language Processing*, Association for Computational Linguistics, Morristown, New Jersey, pp. 133–140.

Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science 42*(1), 7–15. John Wiley & Sons, New York.

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. In C. Nikolaou and C. Stephanidis (Eds.) *Proceedings of the 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Springer-Verlag, Berlin Heidelberg, pp. 569–584.

Hiemstra, D. and F.M.G. de Jong (1999). Disambiguation strategies for cross-language information retrieval. In S. Abiteboul and A.M. Vercoustre, (Eds.) *Proceedings of the 3rd European*

Conference on Research and Advanced Technology for Digital Libraries (ECDL), Springer-Verlag, Berlin Heidelberg, pp. 274–293.

Hiemstra, D. (2000). A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries 3*(2), 131–139, Springer-Verlag, Berlin Heidelberg.

Hiemstra, D. (2001). *Using Language Models for Information Retrieval.* Ph.D. thesis, Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede. http://www.ub.utwente.nl/webdoc/docs/inf.shtml

Jelinek, F. (1997). *Statistical Methods for Speech Recognition.* MIT Press, Boston.

Kowalski, G. (1997). *Information Retrieval Systems: Theory and Implementation.* Kluwer Academic Publishers, Dordrecht.

Kraaij, W. and R. Pohlmann (1996). Viewing stemming as recall enhancement. In H.P. Frei, D. Harman, P. Schäuble, and R. Wilkinson (Eds.) *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 40–48, ACM Press, New York.

Kraaij, W. and R. Pohlmann (1998). Comparing the effect of syntactic vs. statistical phrase index strategies for dutch. In C. Nikolaou and C. Stephanidis (Eds.) *Proceedings of the 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Springer-Verlag, Berlin Heidelberg, pp. 605–617.

Lovins, J.B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics 11*(1-2), 22–31.

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

Miller, D.R.H., T. Leek, and R.M. Schwartz (1999). A hidden Markov model information retrieval system. In M. Hearst, F. Gey, and R. Tong (Eds.) *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 214–221, ACM Press, New York.

Mitra, M., C. Buckley, A. Singhal, and C. Cardie (1997). An analysis of statistical and syntactic phrases. In *Proceedings of the RIAO'97*, pp. 200–216. McGill University, Montreal.

Ponte, J.M. and W.B. Croft (1998). A language modeling approach to information retrieval. In W.B. Croft and A. Moffat and C.J. van Rijsbergen and R. Wilkinson and J. Zobel (Eds.) *Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval (SIGIR'98)*, pp. 275–281, ACM Press, New York.

Porter, M.F. (1980). An algorithm for suffix stripping. *Program 14*, 130–137.

Rabiner, L.R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. In A. Waibel and K.F. Lee (Eds.), *Readings in speech recognition*, pp. 267–296, Morgan Kaufmann, San Fransisco.

Rijsbergen, C.J. van (1979). *Information Retrieval, second edition.* Butterworths, London.

Robertson, S.E. and S. Walker (2000). Okapi / Keenbow at TREC-8. In *Proceedings of the eighth Text Retrieval Conference TREC-8*, pp. 151–162, NIST Special Publications 500-246, Gaithersburg.

Salton, G. and C. Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*(5), 513–523.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal 27*, 379–423, 623–656.

Strzalkowski, T. (1995). Natural language information retrieval. *Information Processing & Management 31*(3), 397–417.