Relating the new language models of information retrieval to the traditional retrieval models *

Djoerd Hiemstra and Arjen P. de Vries
University of Twente, CTIT
P.O. Box 217, 7500 AE Enschede
The Netherlands
{hiemstra,arjen}@ctit.utwente.nl

Abstract

During the last two years, exciting new approaches to information retrieval were introduced by a number of different research groups that use statistical language models for retrieval. This paper relates the retrieval algorithms suggested by these approaches to widely accepted retrieval algorithms developed within three traditional models of information retrieval: the Boolean model, the vector space model and the probabilistic model. The paper shows the existence of efficient retrieval algorithms that only use the matching terms in their computation. Under these conditions, the language models of information retrieval are surprisingly similar to both *tf.idf* term weighting as developed for the vector space model and relevance weighting as developed in the traditional probabilistic model. The paper suggests a new method for relevance weighting and a new method to rank documents giving Boolean queries. Experimental results on the TREC collection indicate that the language modelling approach outperforms the three traditional approaches.

1 Introduction

Information retrieval was one of the first areas of natural language processing in which statistics were successfully applied. Two models of ranked retrieval developed in the late 60s and early 70s are still in use today: Salton's vector space model [28] and Robertson and Sparck-Jones' probabilistic model [23]. Despite of their success, the real breakthrough of statistical models in natural language processing did not come from the information retrieval community, but from the speech recognition community in the 70s and 80s. Many of the statistical techniques that were first successfully applied in speech, like Shannon's noisy channel model, n-gram models and hidden Markov models are used today in all sorts of applications, like for instance part-of-speech tagging, optical character recognition, statistical translation and stochastic context free grammars [4].

Only very recently, it has been shown that statistical language models that were first successfully applied for speech can be used to model information retrieval as well. The past two years show a remarkably large number of publications in which statistical language models are used to compute the ranking of documents given a query. Ponte and Croft [20] were the first to suggest the use of language models in information retrieval. They used estimation based on risk functions to overcome the problem of small sample sizes. Hiemstra and Kraaij [8, 11] were the first to introduce ranking based on a linear interpolation of global and local probabilities that is also used in publications of other groups. Miller et al. [15, 16] use hidden Markov models for ranking, including the use of bi-grams to model two word phrases and a method for performing blind feedback. Berger and Lafferty [1, 2] developed a model that includes statistical translation to model synonyms and related words. A similar model was developed for cross-language information

*Published as CTIT technical report TR-CTIT-00-09, May 2000, http://www.ctit.utwente.nl

retrieval by Hiemstra and de Jong [10] and Kraaij et al. [12]. Ng [18] introduced a model that uses the ratio of the conditional probability of the query given the document and the prior probability of the query, including a method for query expansion. Song and Croft [31] used a model which includes bi-grams and introduced Good Turing re-estimation to smooth the document models.

This paper adds to this discussion by showing that the language models of information retrieval are in fact very similar to the traditional retrieval models. The paper shows the existence of efficient retrieval algorithms that only use the matching terms in their computation. Under these conditions, the language models of information retrieval can be interpreted as belonging to the family of *tf.idf* term weighting algorithms as developed for the traditional vector space model. Also under these conditions, the language models can be interpreted as using the odds of the probability of relevance as used in the traditional probabilistic model of information retrieval, thereby introducing a new method for relevance weighting of search terms. Finally, this paper introduces a new method to rank documents given Boolean queries.

2 Three traditional retrieval models

This section gives a brief description of the three retrieval models we think were the most influential in modern information retrieval research: the vector space model, the probabilistic model and the Boolean model. The section mentions some recent history and gives an indication of where the models stand today.

2.1 The vector space model

The traditional vector space model can best be characterised by its attempt to rank documents by the similarity between the query and each document [28]. Calculations in the vector space model are based on geometry, that is, each term has its own dimension in a multidimensional space; queries and documents are points or vectors in this space and the similarity is usually measured by the cosine measure that computes the angle between two vectors. If documents and queries are weighted vectors $\vec{d}$ and $\vec{q}$ with element weights between 0 and 1 then the cosine measure is defined as:

$$similarity(\vec{d}, \vec{q}) = \frac{\sum_{k=1}^{l} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{l} (d_k)^2 \cdot \sum_{k=1}^{l} (q_k)^2}}$$

A second major achievement of the researchers that developed the vector space model is the introduction of the family of *tf.idf* term weights. These weights have a term frequency (tf) factor measuring the frequency of occurrence of the terms in the document or query texts and an inverse document frequency (idf) factor measuring the inverse of the number of documents that contain a query or document term. If the cosine measure is used, the vector lengths are normalised. The three components tf, idf and length normalisation can be calculated by various formulas and are often reported upon by a three letter combination [26]. One of the recent weighting algorithms Lnu.ltu [30] uses a combination of the document length and the average document length instead of the cosine measure for length normalisation. This algorithm outperforms the cosine versions on the TREC collections, but lacks the metaphor of measuring the angle between two vectors.

2.2 The probabilistic model

The most important characteristic of the probabilistic model is its attempt to rank documents by their probability of relevance given a query [23]. Documents and queries are represented by *binary* vectors $\vec{d}$ and $\vec{q}$, each vector element indicating whether a document attribute or term occurs in the document or query, or not. Instead of probabilities, the probabilistic model uses odds $O(R)$, where $O(R) = P(R)/1 - P(R)$, R means 'document is relevant' and $\bar{R}$ means 'document is not relevant'. The model makes the so-called binary independence assumption, assuming that given

relevance, the attributes are statistically independent.

$$O(R|\vec{d},\vec{q}) \;=\; O(R|\vec{q}) \cdot \prod_{k=1}^{l} \frac{P(d_k|R,\vec{q})}{P(d_k|\bar{R},\vec{q})}$$

This formula can be rewritten into a formula that includes values for term presence only, resulting in:

$$O(R|\vec{d},\vec{q}) \;\propto \sum_{\substack{k \in \text{match-}\\ \text{ing terms}}} \log\left(\frac{P(d_k|R,\vec{q})(1-P(d_k|\bar{R},\vec{q}))}{P(d_k|\bar{R},\vec{q})(1-P(d_k|R,\vec{q}))}\right)$$

If relevance information is available, the model should improve retrieval performance. If no relevance information is available, the model behaves like the vector space model using *idf* weights only (ignoring *tf* and length normalisation), that is, much poorer than most *tf.idf* weights [25]. During TREC-2 and TREC-3, the developers of the probabilistic model tried a considerable number of new term weighting algorithms, which led to the BM25 (BM = best match) term weighting algorithm, an algorithm that includes the three components *tf*, *idf* and length normalisation. Robertson and Walker [24] motivated the best match algorithms by the probabilistic model and by some simple approximations to the 2-Poisson model, but indicated that their result was as much guided by experimentation as by theory.

2.3 The (extended) Boolean model

Introduced already in the 50s, the traditional Boolean model is the oldest of the three [28]. The Boolean model allows for the use of operators of Boolean algebra, AND, OR and NOT, for query formulation, but has one major disadvantage: a Boolean system is not able to rank the returned list of documents. Numerous extensions of the Boolean model were suggested that do provide a ranked list of documents. An extensive comparison both in terms of theoretical properties and retrieval effectiveness of extended Boolean models was conducted by Lee [13]. Lee measured retrieval effectiveness of Boolean queries on one of the TREC subcollections. The best performing extended Boolean models were the P-norm model developed by Salton et al. [27] which like the vector space model is based on geometry and a fuzzy-set model suggested by Paice [19].

Based on recent publications on extended Boolean models, the P-norm model is probably the most popular of the two well-performing models. Greiff, Croft and Turtle [6] copied the behaviour of the P-norm model in their inference network architecture and Losada and Barreiro [14] propose a belief revision operator that is equivalent to a P-norm case.

2.4 Discussion

The vector space model and the probabilistic model stand for different approaches to information retrieval. The former is based on the similarity between query and document, the latter is based on the probability of relevance, using the distribution of terms over relevant and non-relevant documents. Both approaches have proven to be very valuable, guiding research in information retrieval. However, advocates of both models seem to interpret their models quite loosely, preferring methods that perform well experimentally over methods with a strong theoretical motivation.

The language modelling approach to information retrieval presented in this paper shares some interesting characteristics with the three traditional models presented in this section. The paper presents a strong theoretical motivation of the language modelling approach and shows that the approach outperforms the weighting algorithms developed within the traditional models.

3 Statistical language models of retrieval

When using statistical language models for information retrieval, one builds a simple language model for each document in the collection. Given a query, documents are ranked by the probability

that the language model of each document generated the query. Because simple unigram models (n-gram models with $n = 1$) will be used for retrieval, it may be instructive to describe the process of generating the query from the model as if it were a physical process.

3.1 An informal description: the urn model metaphor

The metaphor of "urn models" [17] might give more insight. Instead of drawing balls at random with replacement from an urn, we will consider the process of drawing words at random with replacement from a document. Suppose someone selects one document in the document collection; draws at random, one at a time, with replacement, ten words from this document and hands those ten words (the query terms) over to the system. The system now can make an educated guess as from which document the words came from, by calculating for each document the probability that the ten words were sampled from it and by ranking the documents accordingly. The intuition behind it is that users have a reasonable idea of which terms are likely to occur in documents of interest and will choose query terms accordingly [20]. In practice, some query terms do not occur in any of the relevant documents. This can be modelled by a slightly more complicated urn model. In this case the person that draws at random the ten words, first decides for each draw if he will draw randomly from a relevant document or randomly from the entire collection. The yes/no decision of drawing from a relevant document or not, will also be assigned a probability. This probability will be called the relevance weight of a term, because it defines the distribution of the term over relevant and non-relevant documents. For ad-hoc retrieval all non stop words in the query will be assigned the same relevance weight. The user's feedback might be used to re-estimate the relevance weight for each query term.

The model can be used for Boolean queries by treating the sampling process as an AND-query and allowing that each draw is specified by a disjunction of more than one term. For example, the probability of first drawing the term *information* **and** then drawing either the term *retrieval* **or** the term *filtering* from a document can be calculated by the model introduced in this paper without any additional modelling assumptions.

Furthermore, the model can be extended with additional statistical processes to model differences between the vocabulary of the query and the vocabulary of the documents. Statistical translation can be added to the process of sampling terms from a document by assuming that the translation of a term does not depend on the document it was sampled from. Cross-language retrieval using e.g. English queries on a French document collection uses the sampling metaphor as follows: first an French word is sampled from the document, and then this word is translated to English with some probability that can be estimated from a parallel corpus.

3.2 Definition of the corresponding probability measures

Based on the ideas mentioned above, probability measures can be defined to rank the documents given a query. The probability that a query $T_1, T_2, \cdots, T_n$ of length n is sampled from a document with document identifier D is defined by equation 1.

$$P(T_1, T_2, \cdots, T_n|D) = \prod_{i=1}^{n}((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D)) \tag{1}$$

In the formula, $P(T)$ is the probability of drawing a term randomly from the collection, $P(T|D)$ is the probability of drawing a term randomly from a document and λ_i is the relevance weight of the term. If a query term is assigned a relevance weight of $\lambda_i = 1$, then the term is treated as in exact matching: the system will assign zero probability to documents in which the term does *not* occur. If a query term is assigned a relevance weight of 0, then the term is treated like a stop word: the term does not have any influence on the final ranking. In section 4 it is shown that this probability measure can be rewritten to a *tf.idf* term weighting algorithm. Miller, Leek and Schwartz [15, 16] showed that equation 1 can be interpreted as a two-state hidden Markov model

in which λ and $(1-\lambda)$ define the state transition probabilities and $P(T)$ and $P(T|D)$ define the emission probabilities.

The evaluation of Boolean queries is straightforward. Following again the urn model metaphor, for each draw different terms are mutually exclusive. That is, if one term is drawn from a document, the probability of drawing e.g. both the term *information* and the term *retrieval* is 0. Following the axioms of probability theory (see e.g. Mood [17]) the probability of a disjunction of terms in one draw is the sum of the probabilities of drawing the single terms. This line of reasoning was first introduced to model possible translations or alignments from document terms to query terms [1, 2, 10, 12]. Disjunction of m possible translations T_{ij} $(1 \leq j \leq m)$ of the source language query term on position i is defined as follows.

$$P(T_{i1} \cup T_{i2} \cup \cdots \cup T_{im}|D) = \sum_{j=1}^{m}((1-\lambda_i)P(T_{ij}) + \lambda_i P(T_{ij}|D)) \tag{2}$$

Following this line of reasoning, AND queries are interpreted just like the unstructured queries defined by equation 1. Or, to put it differently, unstructured queries are implicitly assumed to be AND queries. If a relevance weight of $\lambda_i = 1$ is assigned to each query term, then the system will behave like the traditional Boolean model of information retrieval, assigning zero probability to documents that do not exactly match the Boolean query.

In [10, 12] statistical translation is added to these probability measures by assuming that the translation of a term does not depend on the document it occurs in. If $N_1, N_2, \cdots, N_n$ is an English query of length n and the English term on position i has m_i possible French translations T_{ij} $(1 \leq j \leq m_i)$, then the ranking of French documents given the English query would be done according to equation 3

$$P(N_1, N_2, \cdots, N_n|D) = \prod_{i=1}^{n}\sum_{j=1}^{m_i} P(N_i|T_{ij})((1-\lambda_i)P(T_{ij}) + \lambda_i P(T_{ij}|D)) \tag{3}$$

Berger and Lafferty's statistical translation model of information retrieval [1, 2] differs from equation 3 only by using global information for each N_i instead of using global information on each T_{ij}.

3.3 Parameter estimation

As said in section 2, it is good practice in information retrieval to use the term frequency and document frequency as the main components of term weighting algorithms. The term frequency $tf(t,d)$ is defined by the number of times the term t occurs in the document d. The document frequency $df(t)$ is defined by the number of documents in which the term t occurs. Estimation of $P(T)$ and $P(T|D)$ in equation 1 and 2 might therefore be done as follows [8, 11].

$$P(T_i = t_i|D = d) = \frac{tf(t_i,d)}{\sum_t tf(t,d)} \tag{4}$$

$$P(T_i = t_i) = \frac{df(t_i)}{\sum_t df(t)} \tag{5}$$

From the viewpoint of using language models for retrieval and from the viewpoint of the urn model metaphor, equation 5 would not be the obvious method for the estimation of $P(T)$. One might even argue that equation 5 violates the axioms of probability theory, because $P(T_{i1} \cup T_{i2}) \neq P(T_{i1}) + P(T_{i2})$ if T_{i1} and T_{i2} co-occur in some documents. Therefore equation 5b would be the preferred method for the estimation of $P(T)$, where the collection frequency $cf(t)$ is defined by the number of times the term t occurs in the entire collection.

$$P(T_i = t_i) = \frac{cf(t_i)}{\sum_t cf(t)} = \frac{\sum_d tf(t_i,d)}{\sum_d \sum_t tf(t,d)} \tag{5b}$$

The latter method was used in [1, 16, 18, 20]. For historic reasons, that is, because this paper tries to relate the language modelling approach to the traditional approaches, the remainder of this paper will use former method. By using equation 5, the language modelling approach to information retrieval gives a strong theoretical backup for using *tf.idf* term weighting algorithms: a backup that is not provided by the traditional retrieval models.

The prior probability $P(D=d)$ that a document d is relevant, might assumed to be uniformly distributed, in case which the formulas above suffice. Alternatively, it might be assumed that the prior probability of a document being relevant is proportional to the length of the document as in equation 6.

$$P(D = d) = \frac{\sum_t tf(t,d)}{\sum_t \sum_d tf(t,d)} \tag{6}$$

It can be included in the final ranking algorithm by adding the logarithm of equation 6 to the document scores as a final step (see section 4, table 2).

3.4 Relevance weighting

If no information on relevant documents is available, relevance weights λ_i should be constant for each non-stop word in the query. The optimum value of this constant might change for different applications. High relevance weights result in rankings that obey the conditions of co-ordination level ranking [8], that is, documents containing n query terms are always ranked above documents containing $n-1$ query terms. High relevance weights are a good choice for applications that aim at high precision or applications in which very short queries are used, like web search engines. Low relevance weights are a good choice for applications that aim at high precision at all recall points or in applications in which relatively long queries are used, as for instance in the TREC evaluations. For the TREC experiments described in section 5.2 and 5.4 a value of $\lambda_i = 0.15$ was used.

Documents that are judged as relevant by the user can be used to re-estimate the relevance weights for each i separately. Notice that we cannot simply use the proportions of relevant and non-relevant documents that contain a query term to directly estimate the new relevance weight as is done in the probabilistic model. When searching for the best relevance weights, the term frequencies of terms in the relevant documents have to be taken into account. A possible approach to relevance weighting is the EM-algorithm (expectation maximisation algorithm [5]) of figure 1.

$$\text{E-step:} \quad m_i = \sum_{j=1}^{r} \frac{\lambda_i^{(p)} \cdot P(T_i|D_j)}{(1-\lambda_i^{(p)})P(T_i) + \lambda_i^{(p)} P(T_i|D_j)}$$

$$\text{M-step:} \quad \lambda_i^{(p+1)} = \frac{m_i}{r}$$

Figure 1: relevance weighting of query terms: EM-algorithm

The algorithm iteratively maximises the probability of the query $T_1, T_2, \cdots, T_n$ given r relevant documents $D_1, D_2, \cdots, D_r$. Before the iteration process starts, the relevance weights are initialised to their default values $\lambda_i^{(0)}$, where i is the position in the query. Each iteration p estimates a new relevance weight $\lambda_i^{(p+1)}$ by first doing the E-step and then the M-step until the value of the relevance weight does not change significantly anymore.

4 From probability measure to term weighting algorithm

Similar to the probabilistic model, probability measures for ranking documents can be rewritten into a format that is easy to implement. A presence weighting scheme [23] (as opposed to a

presence/absence weighting scheme) assigns a zero weight to terms that are not present in a document. Presence weighting schemes can be implemented using the vector product formula. This section presents the resulting algorithms.

4.1 Relation to *tf.idf* and relevance weighting

First, let's have a look again at the basic probability measure as introduced by equation 1:

$$P(T_1, T_2, \cdots, T_n|D) = \prod_{i=1}^{n}((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D))$$

Dividing the formula by $\prod_{i=1}^{n}((1-\lambda_i)P(T_i))$ will not affect the ranking because λ_i and $P(T_i)$ have the same value for each document. Doing so results in a document ranking function that is somewhat similar to Ng's likelihood ratio formula [18].

$$P(T_1, T_2, \cdots, T_n|D) \propto \prod_{i=1}^{n}(1 + \frac{\lambda_i P(T_i|D)}{(1-\lambda_i)P(T_i)})$$

Any monotonic transformation of the document ranking function will produce the same ranking of the documents. Instead of using the product of weights the formula can be implemented by using the sum of logarithmic weights. Doing so and replacing $P(T_i|D)$ and $P(T_i)$ by the estimates defined in equation 4 and 5 results in:

$$P(T_1 = t_1, T_2 = t_2, \cdots, T_n = t_n|D = d) \propto \sum_{i=1}^{n} \log(1 + \frac{\lambda_i tf(t_i, d) \sum_t df(t)}{(1-\lambda_i) df(t_i) \sum_t tf(t, d)})$$

The resulting formula can be interpreted as a *tf.idf* term weighting algorithm with document length normalisation and the formula can be interpreted as using the odds of probability of relevance (like $O(R|d_i, \vec{q})$ in the traditional probabilistic model) as follows:

$\frac{tf(t_i, d)}{df(t_i)}$	is the *tf.idf* weight of the term t_i in the document d
$\frac{1}{\sum_t tf(t, d)}$	is the inverse length of document d
$\frac{\lambda_i}{1 - \lambda_i}$	is the odds of probability of relevance given the query term on position i
$\sum_t df(t)$	is constant for any document d and term t

The query weights of the vector product formula can be used to account for multiple occurrences of the same term in the query. The resulting vector product version of the ranking formula is displayed in figure 2 [9].

vector product formula: $\text{score}(d, q) = \sum_{k \in \text{matching terms}} w_{qk} \cdot w_{dk}$

query term weight: $w_{qk} = tf(k, q)$

document term weight: $w_{dk} = \log(1 + \frac{tf(k, d) \sum_t df(t)}{df(k) \sum_t tf(t, d)} \cdot \frac{\lambda_k}{1-\lambda_k})$

Figure 2: *tf.idf* term weighting algorithm

4.2 Discussion

The purpose of this paper is not to show that the language modelling approach to information retrieval is so flexible that it can be used to model or implement many other approaches to information retrieval. For this reason, it differs considerably from other publications that also compare retrieval models within one frame work [32, 35]. Although we claim that the language modelling approach may result in *tf.idf* term weighting, the *tf* component and the *idf* component both fall within the logarithm, making it a $tf + idf$ algorithm rather than a $tf.idf$ algorithm. Also, as shown in section 3.3, we would prefer the use of collection frequencies over the use of document frequencies, making it a *tf.icf* algorithm. One may have similar objections against the comparison of the language modelling approach with the probabilistic model. The occurrence in figure 2 of probability odds that are directly related to the distribution of terms over relevant and non-relevant documents, could be related to $O(R|d_i, \vec{q})$, but this measure is at no point in the probabilistic model directly used or estimated, as the model makes the binary independence assumption shown in section 2.2.

Despite of the differences, we think that the similarity between the language model of information retrieval and the traditional models is important, because it gives insight in why *tf.idf* term weighting works and why the combination with relevance weighting as done in the BM25 algorithm works.

4.3 Boolean queries

In section 3.2 it was suggested that the calculation of the probability of possible translations [1, 2, 10, 12] can be used to model Boolean queries as well. Disjunction of m possible translations T_{ij} $(1 \leq j \leq m)$ of the source language query term on position i is defined by equation 2, 4 and 5 as follows.

$$P(T_i{=}t_{i1} \cup T_i{=}t_{i2} \cup \cdots \cup T_i{=}t_{im}|D{=}d) = \sum_{j=1}^{m}((1{-}\lambda_i)\frac{df(t_{ij})}{\sum_t df(t)} + \lambda_i\frac{tf(t_{ij},d)}{\sum_t tf(t,d)})$$

As addition is associative and commutative, we do not have to calculate each probability separately before adding them. Instead, respectively the document frequencies and the term frequencies of the disjuncts can be added beforehand, resulting in a formula that is very similar to the original linear interpolation of global and local probabilities of equation 1. The added frequencies can be used to replace $df(k)$ and $tf(k,d)$ in the weighting formula of figure 2.

$$P(T_i{=}t_{i1} \cup T_i{=}t_{i2} \cup \cdots \cup T_i{=}t_{im}|D{=}d) = (1{-}\lambda_i)\frac{\sum_{j=1}^{m} df(t_{ij})}{\sum_t df(t)} + \lambda_i\frac{\sum_{j=1}^{m} tf(t_{ij},d)}{\sum_t tf(t,d)}$$

A similar algorithm was introduced earlier by Harman [7] for on-line stemming. Harman did not present her algorithm as an extension of Boolean searching, but instead called it 'grouping'. If collection frequencies instead of document frequencies are used, then this method has the nice characteristic that on-line morphological generation treating the morphological variants as disjuncts, will produce exactly the same results as off-line stemming of the documents. For instance the query *funny tables* will be evaluated like the Boolean query on the left-hand side of equation 7 if the index is not stemmed and like the query on the right-hand side if the index is stemmed with the Porter stemmer [21]. The extended Boolean models mentioned in section 2 do not have this characteristic, that is, they will not produce the same results for both queries.

$$P(\text{funny} \cup \text{funnies}, \text{table} \cup \text{tables} \cup \text{tabled}) = P(\texttt{funni}, \texttt{tabl}) \tag{7}$$

Until now it was assumed that Boolean queries are always in conjunctive normal form. This might be true for queries that are generated by a translation module or by a morphological component, but not generally for manually formulated Boolean queries. Both in the language modelling approach and in the extended Boolean models mentioned in section 2, the distributive

laws that hold for conventional Boolean expressions are not valid. For instance, (A AND B) OR C and (A OR C) AND (B OR C) are equivalent in the traditional Boolean model, but this is not generally the case for the language models and the extended Boolean models. For the language model it is even questionable if the former query is a valid query at all; (A AND B) refers to drawing two terms from a document which contradicts with OR C which refers to drawing one term from a document. For the experiments presented in section 5, manually formulated Boolean queries will therefore be converted automatically to their conjunctive normal form. More complex Boolean expressions that do not contradict on the number of draws, like for instance the disjunction of two two-word phrases as in (*funny* AND *tables*) OR (*amusing* AND *chairs*) were also converted to their conjunctive normal form. These queries deserve additional attention in future evaluations.

5 Experimental results

In this section the results are presented of three experiments. All experiments were done with the TREC collection. Each experiment serves to illustrate that the language model of retrieval performs well in situations that call for, respectively, (1) similarity between query and documents; (2) probability of relevance estimation from relevant documents; (3) the ability to process Boolean structured queries.

5.1 The Mirror DBMS

Experiments were done using the Mirror DBMS, a prototype database management system especially designed for multimedia and web retrieval [33]. The Mirror DBMS combines content management and data management in a single system. The main advantage of such integration is the facility to combine information retrieval with traditional data retrieval. Furthermore, information retrieval researchers can experiment more easily with new retrieval models, using and combining various sources of information. This is an important benefit for advanced information retrieval research; web retrieval, speech retrieval, and cross-language retrieval. Each require the use of several representations of content, which is hard to handle in the traditional file-based approach, and becomes too slow in traditional database systems.

Documents and queries were preprocessed as follows. Tokenisation was done by assuming that all non-letters, including hyphens, are word boundaries. Words appearing in the Smart stop list[1] were removed, including some that are specific to the TREC domain, like 'document' and 'relevant'. The remaining words were stemmed using the Porter stemmer [21]. All reported experiments used the same index. For the language model runs, the non-uniform prior probability of relevance of equation 6 was used to correct for different document lengths [34].

5.2 Results on the ad hoc task

The first experiment is a TREC-style automatic ad hoc experiment using TREC topics 401-450. It serves to illustrate that the language modelling approach performs well on a task where no relevance information is available and the system has to rely on the similarity between the query and the documents. The experiment compares the average precision of five different term weighting algorithms. The first is what Salton and Buckley [26] call *original tf.idf* with cosine normalisation. The second is the probabilistic model as introduced by Robertson and Sparck-Jones [23]. Lnu.ltu and BM25 are described in [30] and [25] respectively. The Lnu.ltu slope parameter was set to 0.2 . The BM25 tuning parameters were set to $k_1 = 2$, $b = 0.75$ and $k_3 = \infty$.

The experiment shows that both the original probabilistic model and the original vector space model underperform on this task. The BM25 and Lnu.ltu algorithms perform better, but still considerably worse compared to the language model.

[1] ftp://ftp.cs.cornell.edu/pub/smart/

run	precision at recall:			average
	0.2	0.5	0.8	precision
tfc.tfc	0.211	0.100	0.026	0.126
probabilistic	0.247	0.185	0.079	0.165
Lnu.ltu	0.365	0.227	0.065	0.229
BM25	0.392	0.242	0.073	0.243
LM	0.428	0.265	0.130	0.277

Table 1: results of ad hoc queries

5.3 Results of relevance weighting

The second experiment takes the relevant documents of each topic (401-450) to estimate relevance weights, which are used retrospectively to determine optimal performance on the collection. The same experiment was done by Robertson and Sparck-Jones [23] on the Cranfield collection using the traditional probabilistic model. The purpose of this experiment is two-fold. Firstly, the experiment shows how the language model's relevance weighting method performs compared to relevance weighting of the traditional probabilistic model and the BM25 formula. Secondly, by comparing the performance with the experiments presented in the previous section, the experiments show how much can be gained if the system has perfect knowledge about the distribution of terms over relevant and non-relevant documents.

run	precision at recall:			average
	0.2	0.5	0.8	precision
probabilistic	0.293	0.208	0.120	0.198
BM25	0.416	0.251	0.085	0.263
LM	0.471	0.283	0.147	0.311

Table 2: results of retrospective relevance weighting

The experiment shows that the language model's increase in performance is as good as the increase in performance of the traditional probabilistic model and even better than the performance increase of the BM25 algorithm. Closer inspection of the runs shows that the three methods actually decrease the average precision of respectively 4, 18 and 10 out of 50 queries. This is rather alarming, because a good relevance feedback method should never decrease performance if the weights are used retrospectively. For the language model, we do have a clue why the relevance weighting algorithm seems to be suboptimal. As said in section 3.4, the algorithm optimises the probability that the query is generated from the relevant documents. An optimal algorithm, however, would optimise the probability of relevance given the query. More research into relevance feedback algorithms for the language modelling approach to information retrieval is therefore needed.

5.4 Results of Boolean retrieval

The third experiment uses manually formulated Boolean queries. For this experiment we used the Boolean queries that were formulated by Schiettecatte [29] for TREC topics 301-350. Wild cards and multi-term expressions were replaced by Boolean equivalents, using the OR-connector for wild cards and the AND-connector for multi-term expressions. This experiment tries to answer two questions. First of all this experiment shows how the language model performs compared to the P-norm model. Like the experiments reported by Salton et al. [27] binary query weights and *tf.idf* document weights were used for the P-norm experiments. Experiments were done both using tfc

weights and Ltu weights for documents. Secondly, it measures the additional benefit of extended Boolean models over versions of the model that do not use the Boolean operators. The P-norm model can be reduced to the vector model by assigning a value of 1 to both p parameters. Using a higher value for p, say $p = 2$, should therefore show improved results. The language model does not have a similar knob. Therefore one experiment "LM vector" was done after throwing away the Boolean operators, just leaving the terms. Again, using the Boolean operators as in "LM Boolean" should show improved results compared to not using them.

run	precision at recall:			average
	0.2	0.5	0.8	precision
P-norm tfc p=1	0.162	0.040	0.014	0.084
P-norm tfc p=2	0.193	0.057	0.011	0.102
P-norm Ltu p=1	0.310	0.131	0.031	0.156
P-norm Ltu p=2	0.316	0.157	0.061	0.182
LM vector	0.401	0.206	0.076	0.224
LM Boolean	0.414	0.232	0.090	0.244

Table 3: results of Boolean queries

The experiment shows that not much can be gained by the special treatment of Boolean operators. Special treatment of Boolean operators seems to have the same absolute impact on the P-norm model as on the language model: about 0.02 gain in mean average precision. Term weighting seems to have a bigger impact on retrieval performance. Like for the automatic ad-hoc experiments, the language model outperforms Ltu weights which in turn outperform the traditional tfc weights.

5.5 Document vs. collection frequencies

For the experiments described in this paper we used equation 5 instead of the alternative equation 5b. This section gives the results of the experiments presented above if equation 5b is used.

run	precision at recall:			average
	0.2	0.5	0.8	precision
LM adhoc	0.420	0.259	0.126	0.273
LM rel. weights	0.471	0.285	0.144	0.309
LM vector	0.389	0.201	0.084	0.221
LM Boolean	0.408	0.237	0.093	0.240

Table 4: results of using equation 5b

Comparison of the results given in table 4 with the language model results given in table 1, 2 and 3, indicates that the use of collection frequencies (equation 5b) instead of document frequencies (equation 5) has the tendency to perform a little bit worse (about 1 % on all four runs), but not nearly as bad as generally assumed (see e.g. [3]).

6 Conclusion

In this paper the new language models of information retrieval were compared with three traditional models of information retrieval: the vector space model, the probabilistic model and the

Boolean model. The paper showed that the language modelling approach shares some interesting characteristics with the vector space model and the probabilistic model: it respectively gives an elegant justification for using *tf.idf* weights, and it has a powerful new method for relevance weighting of query terms. The paper also showed that the language modelling approach suggests a new method for extended Boolean retrieval. Experiments on the TREC collection indicate that the language modelling approach outperforms the three traditional methods.

As a side effect of the introduction of language models for retrieval, this paper introduced new ways of thinking about two popular information retrieval tools: the use of stop words and the use of a stemmer. It is hard to talk about stopping and stemming in terms of the traditional models. The fact that often stop words are removed and the remaining words in documents and queries are stemmed is not really suggested or motivated by the traditional models. In the language modelling approach, terms can be assigned a zero relevance weight. These terms will not affect the final ranking, which suggests that they might as well be removed as is done with stop words in the traditional models. As for stemming, the language modelling approach might be used to evaluate Boolean queries that are generated from the user's request by grouping all morphological variations of each word using the Boolean OR operator. As shown in section 4.3, the Boolean OR can be implemented by simply adding respectively the term frequencies and the collection frequencies of the disjuncts. Therefore, on-line morphological generation will produce exactly the same results as off-line stemming, giving the use of a stemmer a theoretical sound interpretation in the language modelling approach.

Although the language modelling approach clearly outperforms the traditional probabilistic model, the latter has some nice theoretical properties. It models the relevance R explicitly and tries to relate relevance weighting to the probability ranking principle [22]. Future research has to point out if it is possible to relate the relevance weighting algorithm of the language model directly to the probability ranking principle as well and hopefully give a formal proof that the algorithm can be expected to improve performance.

Acknowledgements

The work reported in this paper is funded in part by the Dutch Telematics Institute project Druid. We like to thank François Schiettecatte of FS Consulting for making the Boolean queries available to TREC participants.

References

[1] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 222–229, 1999.

[2] A. Berger and J. Lafferty. The Weaver system for document retrieval. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publications, to appear.

[3] K.W. Church and W.A. Gale. Inverse document frequency: a measure of deviations from Poisson. In Armstrong et al. (eds.), *NLP using very large corpora*, Kluwer Academic Publishers, 1999.

[4] K.W. Church and R.L. Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, 1993.

[5] A.P. Dempster, N.M. Laird and D.B. Rubin. Maximum likelihood from incomplete data via the em-algorithm plus discussions on the paper. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[6] W.R. Greiff, W.B. Croft and H.R. Turtle. Computationally tractable probabilistic modelling of boolean operators. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 119–128, 1997.

[7] D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, 1991.

[8] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 569–584, 1998.

[9] D. Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3, 2000, to appear.

[10] D. Hiemstra and F.M.G. de Jong. Disambiguation strategies for cross-language information retrieval. In *Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 274–293, 1999.

[11] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 227–238. NIST Special Publication 500-242, 1999.

[12] W. Kraaij, R. Pohlmann and D. Hiemstra. Twenty-One at TREC-8: using language technologie for information retrieval. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publications, to appear.

[13] J.H. Lee. Analyzing the effectiveness of extended boolean models in information retrieval. Technical Report TR95-1501, Cornell University, 1995. http://cs-tr.cs.cornell.edu/.

[14] D.E. Losada and A. Barreiro. Using a belief revision operator for document ranking in extended boolean models. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 66–73, 1999.

[15] D.R.H. Miller, T. Leek and R.M. Schwartz. BBN at TREC-7: using hidden markov models for information retrieval. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pages 133–142. NIST Special Publication 500-242, 1999.

[16] D.R.H. Miller, T. Leek and R.M. Schwartz. A hidden Markov model information retrieval system. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 214–221, 1999.

[17] A.M. Mood and F.A. Graybill, editors. *Introduction to the Theory of Statistics, Second edition*. McGraw-Hill, 1963.

[18] K. Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publications, to appear.

[19] C.P. Paice. Soft evaluation of boolean search queries in information retrieval systems. *Information Technology: Research and Development*, 3(1):33–42, 1984

[20] J.M. Ponte and W.B. Croft. A language modelling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

[21] M.F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[22] S.E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, 1977.

[23] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[24] S.E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 232–241, 1994.

[25] S.E. Robertson, S. Walker and M. Beaulieu. Okapi at TREC-7: automatic ad hoc, filtering, vlc and interactive. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, pages 253–264. NIST Special Publication 500-242, 1999.

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

[27] G. Salton, E.A. Fox and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[28] G. Salton and M.J. McGill, editors. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[29] F. Schiettecatte. Document retrieval using the mps information server (a report on the trec-6 experiment). In *Proceedings of the 6th Text Retrieval Conference, TREC-6*, pages 477–488. NIST Special Publication 500-240, 1998.

[30] A. Singhal, C. Buckley and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 21–29, 1996.

[31] F. Song and W.B. Croft. A general language model for information retrieval. In *Proceedings of Eighth International Conference on Information and Knowledge Management (CIKM'99)*, 1999.

[32] H.R. Turtle and W.B. Croft. A comparison of text retrieval models. *The Computer Journal*, 35(1):279–289, 1992.

[33] A.P. de Vries. *Content and Multimedia Database Management Systems*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 1999.

[34] A.P. de Vries and D. Hiemstra. The Mirror DBMS at TREC. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publications, to appear.

[35] S.K.M. Wong and Y.Y. Yao. On modelling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, 1995.