Using Metafeatures to Increase the Effectiveness of Latent Semantic Models in Web Search

Alexey Borisov[†, ‡]
alborisov@yandex-team.ru

Pavel Serdyukov[†]
pavser@yandex-team.ru

Maarten de Rijke[‡]
derijke@uva.nl

[†] Yandex, Moscow, Russia
[‡] University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

In web search, latent semantic models have been proposed to bridge the lexical gap between queries and documents that is due to the fact that searchers and content creators often use different vocabularies and language styles to express the same concept. Modern search engines simply use the outputs of latent semantic models as features for a so-called global ranker. We argue that this is not optimal, because a single value output by a latent semantic model may be insufficient to describe all aspects of the model's prediction, and thus some information captured by the model is not used effectively by the search engine.

To increase the effectiveness of latent semantic models in web search, we propose to create *metafeatures*—feature vectors that describe the structure of the model's prediction for a given query-document pair—and pass them to the global ranker along with the models' scores. We provide simple guidelines to represent the latent semantic model's prediction with more than a single number, and illustrate these guidelines using several latent semantic models.

We test the impact of the proposed metafeatures on a web document ranking task using four latent semantic models. Our experiments show that (1) through the use of metafeatures, the performance of each individual latent semantic model can be improved by 10.2% and 4.2% in NDCG scores at truncation levels 1 and 10; and (2) through the use of metafeatures, the performance of a combination of latent semantic models can be improved by 7.6% and 3.8% in NDCG scores at truncation levels 1 and 10, respectively.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*

Keywords

Metafeatures; Latent Semantic Models; Web Search

1. INTRODUCTION

For the majority of cases in which search engine users complain that they cannot find information, while the information does exist in the system, the reasons are due to a mismatch between terms in queries and documents [24]. Term mismatch happens because searchers and content creators often use different vocabularies and language styles to refer to the same concepts [14]. To bridge this lexical gap between queries and documents, *latent semantic models* have been proposed [24].

Today, the use of latent semantic models by search engines is restricted to simply passing their outputs as features to a so-called global ranker, along with outputs of other models used for ranking. We argue that this is not optimal, because a single value output by a latent semantic model may be insufficient to describe all aspects of the latent semantic model's prediction. Let us illustrate this using the Latent Semantic Indexing model [9].

The score produced by the Latent Semantic Indexing model is a sum of scores per latent space dimension. This is where potentially useful information for a global ranker may go missing as the model may assign similar scores to documents with radically different sets of per dimension scores. See Figure 1 for an example of this phenomenon: the sums of the per dimension scores are the same, but the lack of information about the distributions of their values, illustrated by this example, may hinder the global ranker when it attempts to reliably rank the two documents.

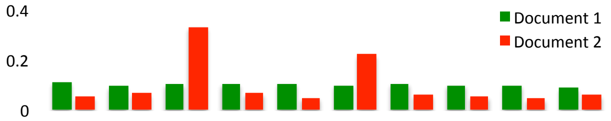


Figure 1: The plot shows the per dimension scores computed for two documents by the Latent Semantic Indexing model. The sums of the scores are the same, but the distributions of their values differ in interesting ways. (Best viewed in color.)

In previous work on latent semantic models in web search, comparisons of models are performed on a web document ranking task using the models' scoring functions [1, 4, 8, 15, 16, 20, 29, 33–36]. While the scores produced by latent semantic models have demonstrated a strong correlation with document relevance, they are just the "tip of the iceberg" in capturing the relation between a query and document.

We argue that considering a latent semantic model's score only is not enough to determine its effectiveness in search, and all potentially useful information captured by the model should be considered. To increase the effectiveness of latent semantic models in search engines, we propose to expose the structure of their predictions to the global ranker. This is done through the use of *metafeatures*—feature vectors that we construct for a latent semantic model to describe its prediction and the way it arrived at this prediction. The latent semantic models' metafeatures are passed to the global ranker along with the models' scores.

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
DOI: http://dx.doi.org/10.1145/2872427.2882987.

The problem of creating metafeatures requires a solid understanding of the model's internal workings. However, as we will see, for many latent semantic models, their scoring functions give a good starting point. In particular, we provide two simple and broadly applicable guidelines for creating metafeatures based on the scoring functions used by latent semantic models, and demonstrate the effectiveness of these guidelines by inferring metafeatures from different latent semantic models in a systematic manner.

We test our ideas for complementing a latent semantic model's scores with metafeatures on a web document ranking task, and demonstrate that metafeatures provide a means to improve the performance of both (1) individual latent semantic models and (2) combinations of multiple latent semantic models.

The proposed approach for creating and using metafeatures is substantially more than just feature engineering (where a significant part of the efforts of search engine employees goes). The process of *feature engineering* described by Domingos [12] consists of three steps: (1) to choose a particular machine learning algorithm and a target evaluation criterion (i.e., the difference in NDCG scores between the global ranker trained with and without candidate features); (2) to generate a large number of candidate features; and (3) to select the best features according to the chosen criterion. However, this approach is difficult to apply in practice, because there are no clear guidelines for selecting the candidate features (and, thus, the effectiveness of features often depends more on an individual's intuition rather than on a solid methodology). To alleviate this problem, we propose a *methodology* that guides the design choice of new features. Our proposal is both novel and ensures a broader impact of our technical contribution beyond the particular task setting (web search) that we consider.

We discuss related work in §2. Our method for creating metafeatures for latent semantic models and passing them along to a global ranker is detailed in §3. We detail our experimental setup in §4 and present our results and analysis in §5. We conclude in §6.

2. RELATED WORK

We discuss three types of related work: latent semantic models, ranking models based on latent semantic models, and combinations of latent semantic models.

Latent semantic models. Latent Semantic Indexing (LSI) [9] uses singular value decomposition (SVD) of a document-term matrix to map queries and documents to low-dimensional concept vectors. The relevance of a document to a query is assumed to be proportional to the cosine similarity between their concept vectors. A major limitation of LSI that prevents it from being used in very large scale applications, is the computational cost of SVD. To overcome this limitation, a Regularized Latent Semantic Indexing (RLSI) [33] with an efficient implementation in MapReduce has been proposed.

Probabilistic Latent Semantic Indexing [18] views documents as mixtures of topics and ranks the documents by the probability of the query given the document distribution over topics. Latent Dirichlet Allocation (LDA) [6, 34] extends PLSI and assumes that topic distributions have a Dirichlet prior.

With increasingly large volumes of user logs, supervised latent semantic models trained on (clicked/not clicked) query-document pairs [1, 4, 8, 15, 16, 20, 29, 35, 36] have begun to outperform many unsupervised latent semantic models trained on documents only. Supervised Semantic Indexing (SSI) [1] and Regularized Mapping to Latent Spaces (RMLS) [35] learn weights for each query-document term pair using low rank matrix decomposition. The Bilingual Topic Model (BLTM) [16] is an extension of LDA, where queries and relevant documents are assumed to share the same distribution over topics, while they might use different topic word distributions for queries and documents.

Another approach to document ranking is based on statistical machine translation [4, 15]. The Word-based Translation Model (WTM) [4] ranks documents by the probability of the query given the document unigram translation model. The Phrase-based Translation Model (PTM) [15] extends WTM with phrases.

Recently, latent semantic models based on neural networks [8, 16, 20, 28, 29, 36] have gained popularity. The Discriminative Projection Model (DPM) [16, 36] maps queries and document to concept vectors with the siamese network architecture [8]. Salakhutdinov and Hinton [28] propose a deep generative model that maps documents to memory addresses in such a way that semantically similar documents are located at nearby addresses. The Deep Structured Semantic Model (DSSM) [20] uses a deep feed-forward neural network to map queries and documents to the concept vectors. DSSM works at the level of character-trigrams, which allows generalization to unseen word forms. The performance of DSSM degrades as the text length increases, as its "bag of character-trigrams" representation leads to a combinatorial blow-up. The Convolutional Latent Semantic Model (CLSM) [29] mitigates this disadvantage by applying a DSSM-like model to word n-grams and combining their vectors at a later stage.

In our experiments we use LDA, WTM, DPM and DSSM as typical representatives of different families of models: topic models (PLSI, LDA, BLTM), translation models (WTM, PTM), conventional models based on matching in latent space (LSI, RLSI, SSI, RMLS, DPM) and recent models working at the level of character trigrams (DSSM, CLSM).

Ranking with latent semantic models. The training objective for LSI and RLSI is to minimize the reconstruction error of the document "bag of words" representation; PLSI and LDA (BLTM) minimize the perplexity of the document corpus (clicked query-document pairs); SSI, DPM minimize the ranking loss between clicked and unclicked documents. Common scoring functions used by the latent semantic models are (1) the cosine similarity between the query and document concept vectors (LSI, RLSI, SSI, RMLS, DPM, DSSM, CLSM) and (2) the query likelihood function (PLSI, LDA, BLTM, WTM, PTM). These scoring functions are simple and intuitive, but we argue that they are not expressive enough to tune latent semantic models for relevance prediction and that they do not use all potentially useful information from the model.

Combinations of latent semantic models. Most work on latent semantic models in search does not address the problem of combining latent semantic models and only provides a comparison of latent semantic models with each other [1, 2, 4, 8, 15, 16, 20, 29, 33, 34, 36]. Some work also uses linear interpolation with traditional retrieval models based on lexical matching (VSM, TFIDF, BM25) for comparison [2, 35, 36]. Wu et al. [35] test the performance of their proposed RMLS model by comparing the performance of a global ranker with baseline ranking models used as features against the global ranker with the baseline ranking models and RMLS. We provide an experimental analysis of the contribution of metafeatures to the combination of latent semantic models.

What we add on top of the work mentioned above is the following. First, we show that the common ways of using latent semantic models in web search (i.e., (1) to rank web documents by scores of the cosine similarity and query likelihood functions, and (2) to pass these scores as features to global ranker) are suboptimal. Second, we propose an approach that extracts more information from latent semantic models and leverages this information to improve

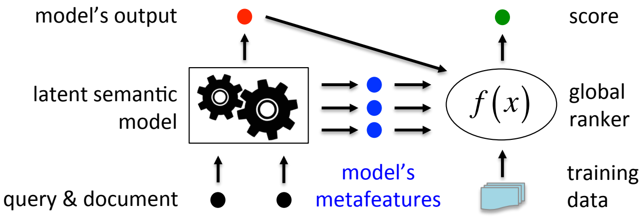


Figure 2: Adding metafeatures to the output of latent semantic models and passing their combination as features to a global ranker.

ranking performance of both individual latent semantic models and their combinations.

3. METHOD

To increase the effectiveness of latent semantic models in (web) search engines, we propose to complement their outputs with *metafeatures*, feature vectors that describe the structure of the predictions of latent semantic models for a given query-document pair, and pass them to a global ranker together with the latent semantic models' scores (see Figure 2).

We start with a disclaimer. There is no general algorithm for generating metafeatures for an arbitrary latent semantic model. However, as we will see, for many latent semantic models, their scoring functions provide a good starting point. In §3.1, we offer two guidelines for creating metafeatures for a latent semantic model based on its scoring function. We operationalize these guidelines in §3.2 using a diverse set of latent semantic models.

3.1 Guidelines for creating metafeatures

We use Q to denote a query composed of terms $q_1, q_2, \ldots, q_{|Q|}$ and D to denote a document composed of terms $w_1, w_2, \ldots, w_{|D|}$.

We write $\mathrm{MF}_n(\mathrm{M}, P)$ to denote the n-th group of metafeatures for a latent semantic model or a class of such models, M; P denotes the parameters of the group of metafeatures. We use a compact form $\mathrm{MF}_{n_1,\ldots,n_k}(\mathrm{M}, P)$ to denote the metafeatures $\mathrm{MF}_{n_1}(\mathrm{M}, P)$, $\ldots$, $\mathrm{MF}_{n_k}(\mathrm{M}, P)$.

Guideline 1 ("Divide and Conquer")

The values making up the score produced by a latent semantic model capture more potentially useful information than the latent semantic model's score by itself. They could, for instance, help to distinguish the two documents shown in Figure 1.

Below, we provide guidelines (a)–(c) for different types of scoring functions. We say that a scoring function $F(Q, D)$ can be expressed as a composition of functions $\mathcal{F} = \{f_i(Q, D)\}$, if for all query-document pairs (Q, D), it is possible to compute the value of $F(Q, D)$ by knowing the values of a subset of $\mathcal{F}$. The values of some $f_i(Q, D)$ might be not defined for a given query-document pair (Q, D), but it should still be possible to compute the value of $F(Q, D)$ using the values of other $f_i(Q, D) \in \mathcal{F}$. The *occurrence probability* of $f_i(\cdot)$ is the probability that $f_i(\cdot)$ is defined for a query-document pair (Q, D) randomly drawn from the (unknown) distribution of query-document pairs.

(a) For a scoring function $F(Q, D)$ that can be expressed as a composition of a fixed number (N) of functions $f_i(Q, D)$, define metafeatures $\mathrm{MF}_1(F)$ by composing them of the values of the functions $f_i(Q, D)$ for a query-document pair (Q, D):

$$\mathrm{MF}_1(F) = (f_1(Q, D), \ldots, f_N(Q, D)).$$

(b) For a scoring function $F(Q, D)$ that can be expressed as a composition of a very large or unlimited number of functions $f_i(Q, D)$, define metafeatures $\mathrm{MF}_1(F, N)$ of size N by composing them of the values of N functions $f_i(Q, D)$ that have the highest occurrence probabilities; if the value of function $f_i(Q, D)$ is not defined for a given query-document pair (Q, D), set the metafeatures' component that corresponds to the function $f_i(Q, D)$ with a NaN (Not a Number) value:

$$\mathrm{MF}_1(F, N) = \left(\widehat{f}_1(Q, D), \ldots, \widehat{f}_N(Q, D)\right),$$

where $\widehat{f}_i(Q, D)$ is $f_i(Q, D)$ if $f_i(Q, D)$ is defined, and NaN otherwise.

(c) For a scoring function $F(Q, D)$ that can be expressed as a composition of a very large or unlimited number of functions $f_i(Q, D)$ that have relatively high occurrence probabilities, define metafeatures $\mathrm{MF}_1(F, k, p_1, \ldots, p_N)$ by composing them of the descriptive statistics of the distribution of values of the functions $f_i(Q, D)$ for a given query-document pair (Q, D), e.g., the expected value $\mathbb{E}[f_i(Q, D)]$, variance $\mathrm{Var}[f_i(Q, D)]$, k min / max values of $f_i(Q, D)$ and percentiles p_i of $\{f_i(Q, D)\}$. The default value of k used in our study is 3; the default percentiles p_i are 0.01, 0.03, 0.05, 0.125, 0.25, 0.5, 0.75, 0.875, 0.95, 0.97, 0.99.

Note: For a scoring function $F(Q, D)$ that can be expressed as a composition of functions $f_i(Q, D)$ that come from two or more different groups $\mathcal{G} = \{g_k : k = 1, \ldots, |\mathcal{G}|\}$, define $|\mathcal{G}|$ scoring functions $F_k(Q, D)$ by setting the values of $f_i \notin g_k$ to constants, and construct metafeatures for $F_k(Q, D)$ using the guidelines (a)–(c).

Guideline 2 ("Find the Strongest and Weakest Links")

For some latent semantic models, there is only a limited number of ways to express a scoring function $F(Q, D)$ as a composition of functions $f_i(Q, D)$. E.g., the scoring functions of WTM and DSSM cannot be expressed as a composition of per document term functions $f_1(Q, D), \ldots, f_{|D|}(Q, D)$, where the functions $f_i(Q, D)$ describe all relevant information about the i-th terms in the document D for computing the scoring function $F(Q, D)$. But if such functions $f_1(Q, D), \ldots, f_{|D|}(Q, D)$ existed, their values could be regarded as contributions of the document terms to the scoring function $F(Q, D)$, and in this way, might help the global ranker to distinguish the documents that "answer" all query terms from those that match only some of them.

We suggest to create "surrogate functions" that perform the role of $f_i(Q, D)$. In particular, we propose to compute gradients of the scoring function $F(Q, D)$ with respect to the occurrences of document terms. This is a reasonable choice, because the components of these gradients measure how the score would change if we removed a small fraction of a term. Below, we capture this intuition more formally.

For a symmetric scoring function $F(Q, D)$ of query terms Q and document terms D (i.e., a function that takes the same value for any permutation of query terms and document terms), first define a function $F(\mathbf{q}, \mathbf{d})$ of vectors $\mathbf{q}$ and $\mathbf{d}$ of vocabulary size $|V|$, whose i-th components are the number of occurrences of the i-th term in query Q and document D, respectively. Then define a matrix M_Q of size $|Q| \times |V|$, whose i-th row is a one-hot vector with the component corresponding to the i-th query term q_i set to 1; and a matrix M_D of size $|D| \times |V|$, whose j-th row is a one-hot vector with the component corresponding to the j-th document term d_j set to 1. Finally, define the metafeatures $\mathrm{MF}_1(F, N_1)$ and $\mathrm{MF}_2(F, N_2)$ as gradients of $F(\mathbf{q}, \mathbf{d})$ with respect to $\mathbf{q}$ and $\mathbf{d}$, $\nabla_Q F(\mathbf{q}, \mathbf{d})$ and $\nabla_D F(\mathbf{q}, \mathbf{d})$, multiplied by the matrices $M_Q(N_1)$

and $M_Q(N_2)$ composed of the first N_1 and N_2 rows of the matrices M_Q and M_D:

$$\begin{aligned} \mathrm{MF}_1(F, N_1) &= M_Q(N_1)\nabla_Q F(\mathbf{q}, \mathbf{d}), \\ \mathrm{MF}_2(F, N_2) &= M_Q(N_2)\nabla_D F(\mathbf{q}, \mathbf{d}). \end{aligned}$$

Sometimes, it is more convenient to apply a monotonic function $g(x)$ to the scoring function $F(\cdot)$, and use the resulting complex function $g(F(\cdot))$ instead of $F(\cdot)$.

As we show in the next section, the presented guidelines help to extract a lot of potentially useful metafeatures from existing state-of-the-art latent semantic models.

3.2 Application of our guidelines

In this section we demonstrate how to apply the guidelines presented in Section 3.1 to different families of latent semantic models. Note that we do not aim to extract all possible metafeatures that could be extracted according to Guidelines 1 and 2, as the main purpose of our study is not an exhaustive search for all of them (which could potentially be a very large number), but a demonstration of the benefits of the methodology of metafeature extraction in general. Thus, we focus on the extraction of the most promising and interpretable metafeatures in this section.

3.2.1 Latent semantic models based on the language modeling approach

We describe the metafeatures that we infer for topic models [6, 16, 18] and the Word-based Translation Model [4] that employ the language modeling approach. We start with the common metafeatures and then take a closer look at each model's scoring function.

Latent semantic models based on the language modeling approach score a query-document pair (Q, D) by the probability of the query Q given the document model M_D. Many latent semantic models make the "bag of words" assumption, which allows one to decompose the probability of the query into the product of the query term probabilities. Thus, the scoring function used by these models is the product of the query term probabilities given the document model, $P(q_i \mid M_D)$:

$$P(Q \mid M_D) = \prod_{i=1}^{|Q|} P(q_i \mid M_D). \tag{1}$$

Following Guideline 1 (b), we use the factors under the product sign of (1) to define metafeatures $\mathrm{MF}_1(\mathrm{QL}, N)$ that capture probabilities of the first N query terms given the document model:

$$\mathrm{MF}_1(\mathrm{QL}, N) = (P(q_1 \mid M_D), \ldots, P(q_N \mid M_D)). \tag{2}$$

These metafeatures may help the global ranker to distinguish between two documents that get very similar scores by the query likelihood scoring function, but for very different reasons. For example, consider two documents D_1 and D_2, such that:

- given the document model for D_1, all query terms q_1, …, $q_{|Q|}$ have roughly the same probabilities; and
- given the document model for D_2, the first query term q_1 has a very low probability, and the other query terms $q_2, \ldots, q_{|Q|}$ have higher probabilities than given the document model for D_1.

Intuitively, these are two different cases. We want the global ranker to know about this through the use of metafeatures. The components of $\mathrm{MF}_1(QL, N)$ can be seen as query term contributions, and are in fact obtained using Guideline 2 for $g(x) = \log x$.

Some work [4, 15, 16] reports that $P(q_i \mid M_D)$ may be too coarse to be used for retrieval and suggests to use linear interpolation with the document unigram language model $P_{\mathrm{LM}}(q_i|D)$ [4], the collection unigram language model $P_{\mathrm{LM}}(q_i \mid C)$ [15] or both [16]:

$$\widetilde{P}(Q \mid M_D) = \prod_{i=1}^{|Q|} \left(\alpha_1 P_{\mathrm{LM}}(q_i \mid C) + \alpha_2 P_{\mathrm{LM}}(q_i \mid D) + \alpha_3 P(q_i \mid M_D)\right),$$

where $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Similarly, following Guideline 1 (b), we define metafeatures $\mathrm{MF}_{2,3}(\mathrm{QL}, N)$ that capture the probabilities of the first N query terms given the document language model and the collection language model:

$$\begin{aligned} \mathrm{MF}_2(\mathrm{QL}, N) &= (P_{\mathrm{LM}}(q_1 \mid C), \ldots, P_{\mathrm{LM}}(q_N \mid C)), \\ \mathrm{MF}_3(\mathrm{QL}, N) &= (P_{\mathrm{LM}}(q_1 \mid D), \ldots, P_{\mathrm{LM}}(q_N \mid D)). \end{aligned}$$

These metafeatures provide extra information about the individual query terms that might help the global ranker to make better use of $\mathrm{MF}_3(\mathrm{QL}, N)$. E.g., low query term probabilities given the latent semantic model $P(q_i \mid M_D)$ are less severe for query terms q_i for which $P_{\mathrm{LM}}(q_i \mid C)$ is high than for query terms q_i for which $P(q_i \mid M_D)$ is low. A high query term probability given by the document model, $P_{\mathrm{LM}}(q_i \mid D)$, indicates that a high probability of a query term given the latent semantic model, $P(q_i \mid M_D)$, is due to a lexical match. In this case, term-based models' predictions might be more reliable for the global ranker.

Topic models. Topic models (e.g., PLSI, LDA, BLTM) view documents as mixtures of topics, i.e., $M_D = \{P(t_j \mid D) : 1 \leq j \leq |T|\}$, where $P(t_j \mid D)$ denotes the probability of topic t_j given the document D, and $|T|$ denotes the number of topics. The probability of a query term q_i given document model M_D, $P(q_i \mid M_D)$, is defined as the sum of the query term probabilities given the topic, $P(q_i \mid t_j)$, weighted by topic probabilities, $P(t_j \mid D)$:

$$P(q_i \mid M_D) = \sum_{j=1}^{|T|} P(q_i \mid t_j) P(t_j \mid D).$$

The document score is a function of (1) topic probabilities given the document, $P(t_j \mid D)$ and (2) query term probabilities given the topic, $P(q_i \mid t_i)$. Guideline 1 suggests that we define metafeatures $\mathrm{MF}_{1,2}(\mathrm{TM})$ that describe these values:

$$\begin{aligned} \mathrm{MF}_1(\mathrm{TM}) &= \left(P(t_1 \mid D), \ldots, P(t_{|T|} \mid D)\right), \\ \mathrm{MF}_2(\mathrm{TM}, N) &= \begin{pmatrix} P(q_1 \mid t_1), & \ldots, & P(q_1 \mid t_{|T|}), \\ \ldots & \ldots, & \ldots \\ P(q_N \mid t_1), & \ldots, & P(q_N \mid t_{|T|}) \end{pmatrix}. \end{aligned}$$

However, the size of $\mathrm{MF}_2(\mathrm{TM}, N)$ might be too large to train the global ranker without causing overfitting. In essence, $\mathrm{MF}_1(\mathrm{TM})$ and $\mathrm{MF}_2(\mathrm{TM}, N)$ are meant to inform the global ranker about how well the document topic distribution helps to maximize the query probability. This can be achieved by comparing the document topic distribution with the query topic distribution—the query topic distribution is, by definition, the topic distribution that maximizes the query probability. This idea was originally proposed in [23], where the authors used Kullback-Leibler divergence between the query and document topic distributions as a scoring function.

$$KL(Q, D) = \sum_{i=1}^{|T|} P(t_i \mid Q) \log \frac{P(t_i \mid Q)}{P(t_i \mid D)} \tag{3}$$

Based on preliminary experiments, the Kullback-Leibler divergence scoring function (3) performed much poorer than the Query Likelihood scoring function (1), but it happened to be useful for creating metafeatures. Following Guideline 1 (a), we define metafeatures

$\mathrm{MF}_1(\mathrm{TM})$ (defined earlier) and $\mathrm{MF}_3(\mathrm{TM})$ that capture the query and document topic distributions:

$$\mathrm{MF}_3(\mathrm{TM}) \quad = \quad \left(P(t_1 \mid Q), \ldots, P(t_{|T|} \mid Q)\right).$$

The Word-based Translation Model. The Word-based Translation Model casts document ranking as a statistical machine translation problem, in which query Q is assumed to be a translation of document D. The document model M_D is defined as a probability distribution of terms given a document. The probability of term w given document model M_D is defined as the average translation probability of document terms $w_1, \ldots, w_{|D|}$ into term w. So the probability of query term q_i given document model M_D is:

$$P(q_i \mid M_D) = \frac{1}{|D|} \sum_{j=1}^{|D|} P(q_i \mid w_j).$$

The document score is a function of the translation probabilities of a document term into a query term. Following Guideline 1 (b) we define metafeatures $\mathrm{MF}_1(\mathrm{WTM}, N_1, N_2)$ that capture probabilities of the query terms given the document terms:

$$\mathrm{MF}_1(\mathrm{WTM}, N_1, N_2) = \begin{pmatrix} P(q_1 \mid w_1), & \ldots, & P(q_1 \mid w_{N_2}), \\ \ldots & \ldots, & \ldots \\ P(q_{N_1} \mid w_1), & \ldots, & P(q_{N_q} \mid w_{N_2}) \end{pmatrix}.$$

As the number of $P(q_i, w_j)$ that occur with a not low frequency is large, we also define metafeatures $\mathrm{MF}_2(\mathrm{WTM})$ from the set of translation probabilities of document terms into query terms $\{P(q_i, w_j)\}_{1 \le i \le |Q|, 1 \le j \le |D|}$, following Guideline 1 (c).

Guideline 2 suggests that passing on information about the query and document term contributions to the document score might be useful for the global ranker. The metafeatures $\mathrm{MF}_1(\mathrm{QL}, N)$ that we have defined earlier (see (2)) describe the query term contributions. To define metafeatures that describe the document term contributions, we first apply a monotonic function $g(x) = \log x$ to the scoring function $P(Q \mid M_D)$ that gives a new scoring function

$$F(Q, D) = g(P(Q \mid M_D)) = \sum_{i=1}^{|Q|} \log \frac{1}{|D|} \sum_{j=1}^{|D|} P(q_i \mid w_j).$$

Then we define a function $F(\mathbf{q}, \mathbf{d})$ of $\mathbf{q}$ and $\mathbf{d}$:

$$F(\mathbf{q}, \mathbf{d}) = \mathbf{q}^T (\log A\mathbf{d} - \log B\mathbf{d}),$$

where A and B are matrices of size $|V| \times |V|$ with $A_{i,j} = P(t_i, t_j)$, and $B_{i,j} = 1$; $\log(\mathbf{x})$ denotes the element-wise logarithm of vector $\mathbf{x}$. Finally, we define the metafeatures that capture quantitative contributions of document terms as follows:

$$\begin{aligned} \mathrm{MF}_3(F, N_2) &= M_Q(N_2) \nabla_D F(\mathbf{q}, \mathbf{d}) \\ &= M_Q(N_2)(A^T \circ \mathrm{inv}(A\mathbf{d}) - B^T \circ \mathrm{inv}(B\mathbf{d}))\mathbf{q}, \end{aligned}$$

where $Z = X \circ \mathbf{y}$ denotes the element-wise multiplication of matrix X by vector $\mathbf{y}$: $Z[i,j] = X[i,j]\mathbf{y}[i]$; and $\mathbf{z} = \mathrm{inv}(\mathbf{x})$ denotes the element-wise multiplicative inverse of vector $\mathbf{x}$: $\mathbf{z}[i] = 1/\mathbf{x}[i]$, if $\mathbf{x}[i] \neq 0$ and 0 otherwise. Briefly, $\mathrm{MF}_3(\mathrm{WTM}, N_2)$ are:

$$\left(\sum_{i=1}^{|Q|} \frac{P(q_i \mid w_1)}{\sum_{j=1}^{|D|} P(q_i \mid w_j)} - \frac{|Q|}{|D|}, \ldots, \sum_{i=1}^{|Q|} \frac{P(q_i \mid w_{N_2})}{\sum_{j=1}^{|D|} P(q_i \mid w_j)} - \frac{|Q|}{|D|} \right).$$

The components of $\mathrm{MF}_3(\mathrm{WTM}, N_2)$ characterize the contributions of document terms w_i with respect to the other terms in the document D: $\alpha_k = \sum_{i=1}^{|Q|} \frac{P(q_i|w_k)}{\sum_{j=1}^{|D|} P(q_i|w_j)}$ is the actual quantitative contribution of the document term w_i; $\beta = \frac{|Q|}{|D|} = \frac{1}{|D|} \sum_{k=1}^{|D|} \alpha_k$ is the average contribution of the document terms $w_1, \ldots, w_{|D|}$.

3.2.2 *Latent semantic models based on the latent space matching approach*

Next, we describe metafeatures that we infer for latent semantic model that employ the latent space matching approach. We start with the metafeatures shared by all models of this class and then take a closer look at the Deep Structured Semantic Model [20].

Latent semantic models based on the latent space matching approach learn vector representations for queries and documents, such that the distance between a query vector v_Q and a document vector v_D reflects the degree of relevance of the document D to the query Q. The standard choice for a distance function between the query and document vectors is the cosine similarity measure [25]:

$$\cos(v_Q, v_D) = \frac{v_Q \cdot v_D}{\|v_Q\| \|v_D\|} = \sum_{i=1}^{N} \frac{v_Q^{(i)} v_D^{(i)}}{\|v_Q\| \|v_D\|}, \qquad (4)$$

where N denotes the dimensionality of the vector space and $v^{(i)}$ denotes the i-th component of vector v.

Following Guideline 1 (a), we use the terms under the summation sign in (4) to define metafeatures $\mathrm{MF}_1(\mathrm{COS})$ that capture the per dimension matching scores:

$$\mathrm{MF}_1(\mathrm{COS}) = \left(\frac{v_Q^{(1)} v_D^{(1)}}{\|v_Q\| \|v_D\|}, \ldots, \frac{v_Q^{(N)} v_D^{(N)}}{\|v_Q\| \|v_D\|} \right).$$

The rationale is analogous to the one underlying $\mathrm{MF}_1(\mathrm{QL}, N)$: we want to be able to distinguish the two cases shown in Figure 1. One document has roughly the same scores in all dimensions, the other one has much higher scores in two dimensions and lower scores in other dimensions.

Viewed differently, the document score is a function of both the query vector v_Q and the document vector v_D. Hence, following Guideline 1 (a), we define metafeatures $\mathrm{MF}_{2,3}(\mathrm{COS})$ that capture the components of the query vector v_Q and document vector v_D:

$$\begin{aligned} \mathrm{MF}_2(\mathrm{COS}) &= \left(v_Q^{(1)}, \ldots, v_Q^{(N)}\right), \\ \mathrm{MF}_3(\mathrm{COS}) &= \left(v_D^{(1)}, \ldots, v_D^{(N)}\right). \end{aligned}$$

The Deep Structured Semantic Model. The Deep Structured Semantic Model (DSSM) uses a technique called *word hashing*, which represents queries and documents with letter-trigram vectors. For example, *San Francisco* is encoded with the following letter-trigrams #sa, san, an#, #fr, fra, ran, anc, nci, cis, isc, sco, co#, where # is a boundary symbol. These letter-trigram vectors are passed through the three fully connected layers of a feed-forward neural network to obtain vector representations v_Q and v_D, as illustrated in Figure 3.

Guideline 2 suggests that we construct metafeatures that capture the quantitative contributions of query and document terms to the document score. But since DSSM works at the letter-trigram level, we consider letter-trigram contributions.[1]

We use the following notation: $|T|_c$ is the length of text T in characters; $|L|$ is the total number of letter-ngrams; $\mathbf{q_c}$ and $\mathbf{d_c}$ are vectors of size $|L|$, whose elements on the i-th position are the numbers of occurrences of the i-th letter-ngram in query Q and document D, respectively; $v_Q(\mathbf{q_c})$ and $v_D(\mathbf{d_c})$ are functions of $\mathbf{q_c}$ and $\mathbf{d_c}$ that return the semantic representations of query Q and docu-

[1] In the case of the DSSM model, term contributions are the sums of the contributions of the letter-trigrams that constitute the term.

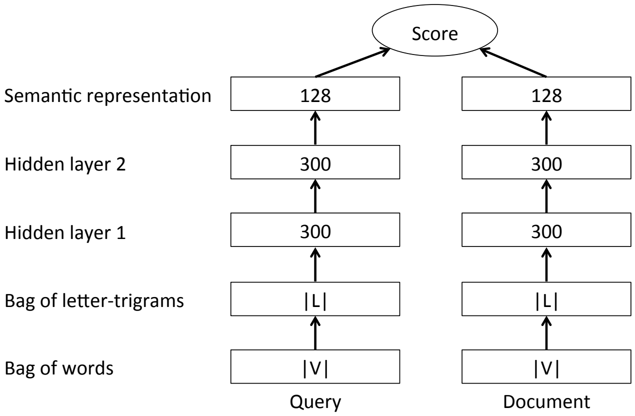


Figure 3: The Deep Structured Semantic Model (DSSM) [20].

ment D (these functions are learned by DSSM); M_Q is a matrix of size $|Q|_c \times |V|$, whose i-th row is a one-hot vector with the component corresponding to the i-th query letter-trigram set to 1; M_D is a matrix of size $|D|_c \times |V|$, whose i-th row is a one-hot vector with the component corresponding to the i-th document letter-trigram set to 1. We write $M_Q(N_1)$ and $M_D(N_2)$ for the matrices composed of the first N_1 and N_2 rows of the matrices M_Q and M_D.

We define metafeatures that account for the contributions of the query and document letter-trigrams as follows:

$$\begin{aligned}\mathrm{MF}_1(\mathrm{DSSM}, N_1) &= M_Q(N_1)\nabla_{\mathbf{q_c}} \cos\left(v_Q(\mathbf{q_c}), v_D(\mathbf{d_c})\right),\\ \mathrm{MF}_2(\mathrm{DSSM}, N_2) &= M_Q(N_2)\nabla_{\mathbf{d_c}} \cos\left(v_Q(\mathbf{q_c}), v_D(\mathbf{d_c})\right).\end{aligned}$$

Using the chain rule $\nabla_{\mathbf{x}} F(\mathbf{y}(\mathbf{x})) = J^T_{\mathbf{x}}\left(\mathbf{y}(\mathbf{x})\right)\nabla_{\mathbf{y}} F(\mathbf{y})$, where $J_{\mathbf{x}}\left(\mathbf{y}(\mathbf{x})\right)$ denotes the Jacobian matrix of a vector function $\mathbf{y}(\mathbf{x})$ of a vector $\mathbf{x}$ w.r.t. the vector $\mathbf{x}$, and the equality $\nabla_{\mathbf{x}} \cos(\mathbf{x}, \mathbf{y}) = \nabla_{\mathbf{x}} \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|} = \frac{\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|} - \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|^3\|\mathbf{y}\|}$, we rewrite $\mathrm{MF}_{1,2}(\mathrm{DSSM}, N_1)$ as follows:

$$\begin{aligned}\mathrm{MF}_1(\mathrm{DSSM}, N_1) &= M_Q(N_1)J^T_{\mathbf{q_c}}(v_Q(\mathbf{q_c}))\nabla_{v_Q}\cos\left(v_Q, v_D\right)\\ &= M_Q(N_1)J^T_{\mathbf{q_c}}(v_Q(\mathbf{q_c}))\left(\frac{v_Q}{\|v_Q\|\|v_D\|} - \frac{v_D^T v_Q}{\|v_Q\|^3\,\|v_D\|}\right),\\ \mathrm{MF}_2(\mathrm{DSSM}, N_2) &= M_Q(N_2)J^T_{\mathbf{d_c}}(v_D(\mathbf{d_c}))\nabla_{v_D}\cos\left(v_Q, v_D\right)\\ &= M_Q(N_2)J^T_{\mathbf{q_c}}(v_D(\mathbf{d_c}))\left(\frac{v_D}{\|v_Q\|\|v_D\|} - \frac{v_D^T v_Q}{\|v_Q\|\,\|v_D\|^3}\right).\end{aligned}$$

The elements of $J_{\mathbf{q_c}}(v_Q(\mathbf{q_c}))$ and $J_{\mathbf{d_c}}(v_D(\mathbf{d_c}))$ are the first derivatives of the functions in the output layer of the neural network w.r.t. the input vectors $\mathbf{q_c}$ and $\mathbf{d_c}$ (see e.g., [3] for the exact formulas).[2]

The metafeatures $\mathrm{MF}_1(\mathrm{DSSM}, N_1)$ and $\mathrm{MF}_2(\mathrm{DSSM}, N_2)$ do not capture information about word boundaries. This might be a disadvantage because the importance of a letter-trigram, which we define as its expected contribution, varies with its position. E.g., letter-trigrams at the end of a term are typically less informative than those in the middle (for many European languages) [19].

To capture the boundaries between terms in the query and terms in the documents we define metafeatures $\mathrm{MF}_{3,4,5,6}(\mathrm{DSSM}, N)$:

$$\begin{aligned}\mathrm{MF}_3(\mathrm{DSSM}, N_1) &= \left(\mathit{start}(Q, 1), \ldots, \mathit{start}(Q, N_1)\right),\\ \mathrm{MF}_4(\mathrm{DSSM}, N_1) &= \left(\mathit{end}(Q, 1), \ldots, \mathit{end}(Q, N_1)\right),\\ \mathrm{MF}_5(\mathrm{DSSM}, N_2) &= \left(\mathit{start}(D, 1), \ldots, \mathit{start}(D, N_2)\right),\\ \mathrm{MF}_6(\mathrm{DSSM}, N_2) &= \left(\mathit{end}(D, 1), \ldots, \mathit{end}(D, N_2)\right),\end{aligned}$$

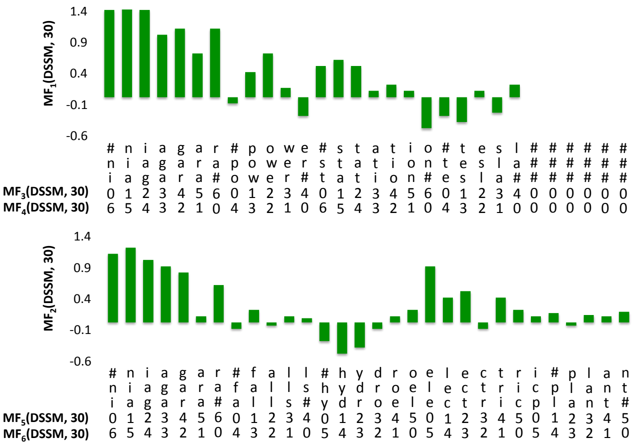


Figure 4: Metafeatures $\mathrm{MF}_{1,2,3,4,5,6}(\mathrm{DSSM}, 30)$ **for query** *"niagara power station tesla"* **and document** *"niagara falls hydroelectric plant"*.

where $\mathit{start}(T, p)$ and $\mathit{end}(T, p)$ denote the offsets of the p-th letter-trigram in the text T from the first and the last trigrams of the term it falls. An illustration is given in Figure 4.

Note that for more sophisticated latent semantic models, the metafeatures can be more sophisticated. E.g., following Guideline 2 for DSSM and WTM we arrive at the most non-trivial metafeatures among the ones that we have proposed, such as $\mathrm{MF}_1(\mathrm{DSSM}, \mathrm{N}_1)$, $\mathrm{MF}_2(\mathrm{DSSM}, \mathrm{N}_2)$ and $\mathrm{MF}_3(\mathrm{WTM}, \mathrm{N}_2)$. However, as we demonstrate in our experiments below, all metafeatures, regardless of their complexity, are useful for improving retrieval quality.

4. EXPERIMENTAL SETUP

4.1 Research questions

We seek to answer the following research questions.

RQ1 Do metafeatures associated with latent semantic models help improve the performance of individual latent semantic models on a web document ranking task? In particular:

(a) Which of the metafeatures defined in Section 3.2 help to improve performance of the underlying latent semantic models?

(b) Does employing a combination of different metafeatures inferred from a single latent semantic model yield better performance than individual metafeatures?

(c) How does the performance of the global ranker, trained with metafeatures, vary with the size the training data set?

RQ2 Do metafeatures associated with latent semantic models provide a means to improve the performance of a combination of latent semantic models on the web document ranking task.

4.2 Data sets and evaluation methodology

In modern search engines, a document is described by multiple fields, including body text, title, URL and anchor texts [30]. In our experiments, we focus on the title field. We do this for the following reasons. First, recent work on supervised latent semantic models in web search focuses on the title field [15, 16, 20, 29, 35].[3] Second, the title field gives better [29] or, at least, as effective [30] retrieval results as the body text field (using BM25). Third, some models achieve state-of-the-art performance using the title field, but do not

[2] Since DSSM uses weight sharing, $J_{\mathbf{q_c}}(v_Q(\mathbf{q_c})) = J_{\mathbf{d_c}}(v_D(\mathbf{d_c}))$.

[3] Although our work targets supervised latent semantic models, we include a comparison with LDA, as a well-known baseline.

apply to other fields (e.g., it is not feasible to train DSSM [20] on the body text field, because of the extremely large number of non-zero elements in the word hashing layer).

To train latent semantic models, we collected a training data set that comprises 130,024,971 search sessions sampled from a commercial web search engine over a six months period.[4] It contains a total of 51,117,758 unique user queries and 178,289,551 retrieved documents. Figures 5 and 6 show the query and document distributions of the number of words and characters in the training data set. We use these statistics to set the parameters of metafeatures in our experiments (Section 4.4).

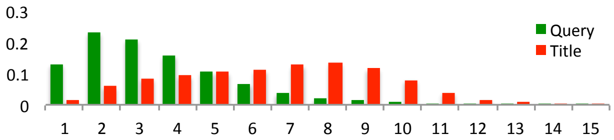


Figure 5: The query and title distributions of the number of words in the training data set. (Best viewed in color.)

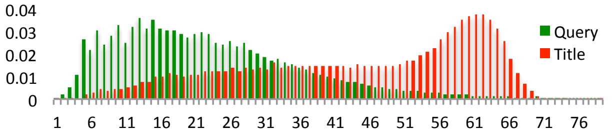


Figure 6: The query and title distributions of the number of characters in the training data set. (Best viewed in color.)

To evaluate the performance of the models, we collected an evaluation data set that contains 111,203 queries sampled from the query-log files of a commercial web search engine.[5] On average, each query is associated with 27 documents (URLs). Each query-document pair has a human-generated relevance label on a scale from 0 to 4, with $0 =$ bad, $1 =$ fair, $2 =$ good, $3 =$ excellent, $4 =$ perfect. Performance is measured using mean Normalized Discounted Cumulative Gain (NDCG) [21] at truncation levels 1, 3, 5 and 10, using k-fold cross-validation. We use $(k-1)$ folds to train the global ranker and 1 fold to evaluate the performance; the results over k folds are averaged. We perform significance testing using a paired t-test; differences are considered statistically significant for p-values lower than 0.001.

4.3 Model settings and baseline performance

We use four latent semantic models for which we infer metafeatures. The baseline performance achieved by the models after optimization is listed in Table 1. Below we describe how the models are trained. For comparison, we also list a state-of-the art ranking model based on lexical matching, BM25.

BM25 (row 1 in Table 1) is a state-of-the-art document ranking model based on lexical matching. The parameters k_1 and b are optimized using Powell's method [27].

LDA (rows 2–3 in Table 1) is our MapReduce implementation of the topic model proposed in [6]. It is trained with 200 iterations of Gibbs sampling using $\alpha = 50/|T|$ and $\beta = 0.01$ (the default values used, e.g., in [34]). The number of topics $|T|$ is set to 100 (used, e.g., in [16]).[6] We consider ranking models based on the unsmoothed LDA model and the LDA model smoothed using a title unigram language model and a background unigram language model. The ranking model based on the smoothed version of LDA model (row 3) outperforms the ranking model based on the unsmoothed version of LDA model (row 2) by a large margin. In the rest of our experiments we use the smoothed version.

WTM (rows 4–5 in Table 1) is our implementation of the Word-based Translation Model [4]. It is trained on the query-title pairs. We consider ranking models based on the unsmoothed WTM and the WTM smoothed using a title unigram language model and a background unigram language model. We find that the ranking model based on the smoothed version of WTM (row 5) outperforms the ranking model based on the unsmoothed version of WTM (row 4) by a large margin. In the rest of our experiments we use the smoothed version of WTM.

DPM (row 6 in Table 1) is our implementation of the Discriminative Projection Model proposed in [16, 36]. It is trained on clicked query-title pairs. The number of dimensions is set to 300 (the performance does not improve much for larger numbers [36], while the training time is linear in the number of dimensions).

DSSM (row 7 in Table 1) is our implementation of the Deep Structured Semantic Model proposed in [20]. It is trained by maximizing the conditional likelihood of the clicked query-title pairs. We use the configuration proposed in [20] (the numbers of neurons in the hidden layers are $\{300, 300, 128\}$).

ALL (row 8–9 in Table 1) is a combination of LDA (w/ smoothing), WTM (w/ smoothing), DPM and DSSM by a linear model (row 8) and Gradient Boosted Regression Trees (GBRT [13], row 9).

All ranking models based on latent semantic models outperform the BM25 ranking model by a large margin, on all metrics. DSSM is the best performing individual latent semantic model, again on all metrics. The two combination models outperform all individual models and GBRT outperforms the linear model. In the rest of the experiments, we use GBRT as our global ranker.

Table 1: Performance of the baseline ranking models.

		NDCG			
#	Ranking model	@1	@3	@5	@10
1	BM25	0.520	0.583	0.631	0.709
2	LDA (w/o smoothing)	0.538	0.608	0.657	0.734
3	LDA (w/ smoothing)	0.552	0.619	0.667	0.741
4	WTM (w/o smoothing)	0.545	0.611	0.659	0.736
5	WTM (w/ smoothing)	0.560	0.626	0.673	0.746
6	DPM	0.537	0.602	0.648	0.724
7	DSSM	0.568	0.640	0.687	0.759
8	ALL (linear model)	0.580	0.649	0.694	0.764
9	ALL (GBRT)	0.604	0.664	0.706	0.772

4.4 Experiments

We design two experiments to answer our research questions.

Experiment 1. To answer RQ1, we compare the performance of each individual latent semantic model's score against the output of the global ranker trained with the latent semantic model's score as well as the metafeatures inferred from the latent semantic model used as features.

[4] There is no publicly available data set for training supervised latent semantic models [1, 4, 15, 16, 20, 29, 35, 36].

[5] There is no sufficiently large publicly available dataset with relevance labels and disclosed query and document terms (the TREC data set is too small to train the global ranker; the well-known Yahoo L2R and MSLR data sets do not contain query terms and document terms).

[6] Shen et al. [29] used LDA with 100 and 500 topics, but the observed changes were not marked as statistically significant.

In Section 3.2 we have introduced some metafeatures together (e.g., $\mathrm{MF}_2(\mathrm{COS})$ and $\mathrm{MF}_3(\mathrm{COS})$) and provided the intuitions why those metafeatures might be helpful for the global ranker. We evaluate the impact of the metafeatures within the groups where they were introduced (Table 2).

For the metafeatures $\mathrm{MF}_{1,2,3}(\mathrm{QL}, N)$, $\mathrm{MF}_1(\mathrm{WTM}, N_1, N_2)$ and $\mathrm{MF}_2(\mathrm{WTM}, N)$ we set $N = N_1 = N_2 = 15$, because less than 0.3% of the queries and 0.2% of the titles in the training set contain more than 15 words (Figure 5). For the metafeatures $\mathrm{MF}_{1,2,3,4,5,6}(\mathrm{DSSM}, N)$ we set $N = 80$ because less than 0.4% of the queries and 0.2% of the titles in the training set contain more than 80 characters (Figure 6).

Table 2: Experiment 1 (runs).

Metafeatures	Latent semantic models
$\mathrm{MF}_{1,2,3}(\mathrm{QL}, 15)$	LDA, WTM
$\mathrm{MF}_{1,3}(\mathrm{TM})$	LDA
$\mathrm{MF}_1(\mathrm{WTM}, 15, 15)$	WTM
$\mathrm{MF}_2(\mathrm{WTM})$	WTM
$\mathrm{MF}_1(\mathrm{QL}, 15), \mathrm{MF}_3(\mathrm{WTM}, 15)$	WTM
$\mathrm{MF}_1(\mathrm{COS})$	DPM, DSSM
$\mathrm{MF}_{2,3}(\mathrm{COS})$	DPM, DSSM
$\mathrm{MF}_{1,2,3,4,5,6}(\mathrm{DSSM}, 80)$	DSSM

Experiment 2. To answer RQ2, we compare the performance of the global ranker trained with the individual latent semantic models' scores only (for LDA, WTM, DPM, DSSM) against the global ranker trained with those individual latent semantic models' scores plus the metafeatures inferred from those models.

5. RESULTS

We present the outcomes of the two experiments described in Section 4.4, and provide answers to our research questions.

5.1 Experiment 1

The results of Experiment 1 are given in Table 3 and Figure 7. Table 3 compares the performance of each individual latent semantic model and the global ranker trained using the latent semantic model's score together with the metafeatures listed in Table 2 as features. Figure 7 plots the performance (in terms of NDCG@10) for the latent semantic models with the metafeatures listed in Table 2 vs. the number of queries used for training.

A first general observation from Table 3 is that for every latent semantic model, the addition of all metafeatures defined for that model yields the highest performance. We also see that combinations of metafeatures nearly always lead to performance increases. Let us now turn to each research question individually.

RQ1 (a). We find that all metafeatures defined in Section 3.2 yield statistically significant improvements in NDCG scores at truncation levels 1, 3, 5 and 10 over the underlying rankings produced by the latent semantic models' scores, for all latent semantic models. The relative improvements in NDCG scores at truncation levels 1, 3, 5 and 10 are above 10.2%, 7.8%, 6.4% and 4.2%, respectively (Table 4). Hence, our metafeatures have a clear early precision enhancing effect.

RQ1 (b). We find that the best results for each latent semantic model are obtained by the combination of all latent semantic model metafeatures ("ALL MF"). However, combinations of metafeatures do not always lead to performance improvements. In particular, for DPM, the addition of $\mathrm{MF}_1(\mathrm{COS})$ does not lead to a statistically significant improvement over the metafeatures $\mathrm{MF}_{2,3}(\mathrm{COS})$. And for DSSM $\mathrm{MF}_{1,2,3}(\mathrm{COS})$ does not outperform $\mathrm{MF}_{2,3}(\mathrm{COS})$. This can be explained by the fact that the components of $\mathrm{MF}_1(\mathrm{COS})$ are the products of the components of $\mathrm{MF}_2(\mathrm{COS})$ and $\mathrm{MF}_3(\mathrm{COS})$: $\mathrm{MF}_1(\mathrm{COS})$ does not bring any new information to the global ranker trained with $\mathrm{MF}_{2,3}(\mathrm{COS})$, but increases the number of features used by the global ranker, which results in a stronger overfitting of the global ranker.

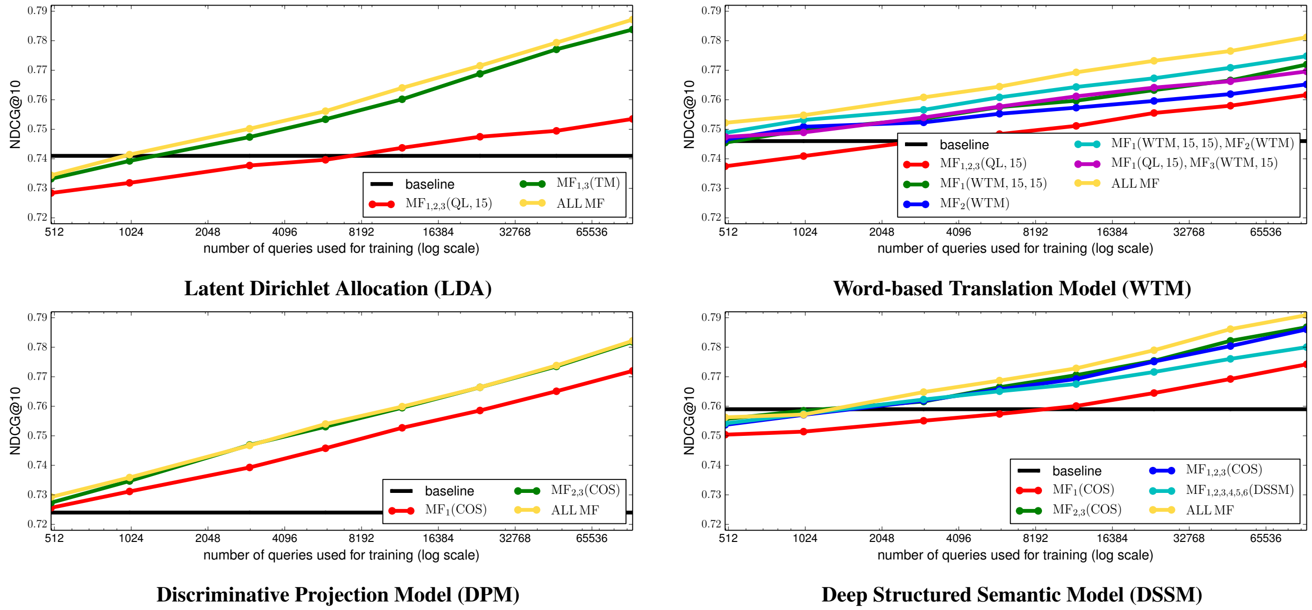
Table 3: Performance of the ranking models based on the latent semantic models with different metafeatures (10-fold cross-validation). Highest scores per latent semantic model are indicated in boldface. The improvements of models with metafeatures over their respective baseline models (i.e., LDA, WTM, DPM and DSSM) are statistically significant ($p < 0.001$).

	NDCG			
Global ranker features	@1	@3	@5	@10
LDA	0.552	0.619	0.667	0.741
+ $\mathrm{MF}_{1,2,3}(\mathrm{QL}, 15)$	0.564	0.631	0.680	0.754
+ $\mathrm{MF}_{1,3}(\mathrm{TM})$	0.619	0.679	0.720	0.784
+ ALL MF	**0.625**	**0.684**	**0.725**	**0.787**
WTM	0.560	0.626	0.673	0.746
+ $\mathrm{MF}_{1,2,3}(\mathrm{QL}, 15)$	0.576	0.645	0.691	0.762
+ $\mathrm{MF}_1(\mathrm{WTM}, 15, 15)$	0.599	0.660	0.704	0.772
+ $\mathrm{MF}_2(\mathrm{WTM})$	0.593	0.655	0.699	0.765
+ $\mathrm{MF}_1(\mathrm{WTM}, 15, 15)$	0.605	0.667	0.710	0.775
+ $\mathrm{MF}_1(\mathrm{QL}, 15)$				
+ $\mathrm{MF}_3(\mathrm{WTM}, 15)$	0.597	0.658	0.701	0.770
+ ALL MF	**0.617**	**0.675**	**0.717**	**0.781**
DPM	0.537	0.602	0.648	0.724
+ $\mathrm{MF}_1(\mathrm{COS})$	0.605	0.664	0.706	0.772
+ $\mathrm{MF}_{2,3}(\mathrm{COS})$	0.614	0.675	0.717	**0.782**
+ $\mathrm{MF}_{1,2,3}(\mathrm{COS})$				
(= ALL MF)	**0.615**	**0.676**	**0.718**	**0.782**
DSSM	0.568	0.640	0.687	0.759
+ $\mathrm{MF}_1(\mathrm{COS})$	0.609	0.668	0.710	0.774
+ $\mathrm{MF}_{2,3}(\mathrm{COS})$	0.627	0.685	0.725	0.787
+ $\mathrm{MF}_{1,2,3}(\mathrm{COS})$	0.624	0.683	0.723	0.786
+ $\mathrm{MF}_{1,2,3,4,5,6}(\mathrm{DSSM}, 80)$	0.617	0.673	0.715	0.780
+ ALL MF	**0.634**	**0.691**	**0.731**	**0.791**

RQ1 (c). We find that the gains obtained from metafeatures in terms of NDCG@10 are proportional to the logarithm of the size of the training data set: the average Pearson correlation coefficient, $r = 0.860$ (with all values in the interval $[0.825; 0.921]$). For some models, viz. LDA, WTM, DSSM, the performance achieved with a

Table 4: Relative improvements of the rankings given by the global ranker trained with the individual the latent semantic models' scores and all metafeatures defined for the model over the rankings given by the individual latent sematic models' scores (10-fold cross-validation).

	NDCG			
Latent semantic model	@1	@3	@5	@10
LDA	+13.2%	+10.5%	+8.7%	+6.2%
WTM	+10.2%	+7.8%	+6.5%	+4.7%
DPM	+14.5%	+12.3%	+10.8%	+8.0%
DSSM	+11.6%	+8.0%	+6.4%	+4.2%



Latent Dirichlet Allocation (LDA)

Word-based Translation Model (WTM)

Discriminative Projection Model (DPM)

Deep Structured Semantic Model (DSSM)

Figure 7: Performance of the latent semantic models with different metafeatures for different sizes of training data. (Best viewed in color.)

relatively small number of queries in the training set (i.e., < 12500) is below the performance of the baseline (not using metafeatures). This is apparently a result of overfitting, as we observed similar results for the global ranker trained with only one feature—the latent semantic model's score. With more queries added for training, all metafeatures end up beating the baseline, and significantly so.

Together, the above results lead to the conclusion that the proposed metafeatures improve the performance of the latent semantic models on the web document ranking task that we consider. Interestingly, through the addition of metafeatures the ranking of latent semantic models has changed: from DSSM $>$ WTM $>$ LDA $>$ DPM (on all metrics) we went to DSSM $>$ LDA $>$ DPM $\approx$ WTM (again, on all metrics) (Table 3). This reveals an important experimental issue: fair comparison of latent semantic models w.r.t. a document ranking task assumes utilizing all potentially useful knowledge mined during the building of these models.

5.2 Experiment 2

The outcomes of Experiment 2 are listed in Table 5. We find that the metafeatures yield a statistically significant improvement over the output of the global ranker trained with the latent semantic models' scores only. The relative improvements in NDCG scores at truncation levels 1, 3, 5 and 10 are 7.6%, 6.17%, 5.38% and 3.75%, respectively.

Table 5: Performance of the model combinations with the latent semantic models' scores only and the latent semantic models' scores and metafeatures (10-fold cross validation). The improvements are statistically significant ($p < 0.001$)**.**

	NDCG			
Global ranker features	@1	@3	@5	@10
models' outputs	0.604	0.664	0.706	0.772
+ models' metafeatures	**0.650**	**0.705**	**0.744**	**0.801**

Next, we take a closer look at the changes brought about by the inclusion of metafeatures in the combination of latent semantic models. Table 6 provides a matrix of the changes in relevance labels for the documents returned in the top position for each query in the evaluation data set by the combination of the latent semantic models' scores only (w/o MF) and by the combination extended with the metafeatures (w/ MF). We counted the number of times that the retrieved document is in each of the five relevance categories, from bad to perfect, for both combinations.

We observe that for 70% of the queries, the relevance labels of the documents returned in the top position did not change. For the remaining queries, the number of cases in which the highest ranked document returned by the combination w/ MF is more relevant than one returned by the combination w/o MF exceeds the number of cases in which the combination w/o MF returns a more relevant document than the combination w/ MF for all relevance label pairs. Furthermore, the relative number of cases in which the combination w/ MF returns a more relevant document than the combination w/o MF is smaller for low-performing queries (for which the retrieved documents' labels fall into *bad*, *fair* or *good* categories) than for high-performing queries (for which the retrieved documents' labels fall are *excellent* or *perfect*).

In sum, metafeatures help to bring high-quality documents to the top position rather than that they help to replace non-relevant documents with partially relevant ones.

6. CONCLUSION AND FUTURE WORK

In this paper, we have introduced an approach to increase the effectiveness of latent semantic models on a web document ranking task. The approach is based on so-called *metafeatures*—feature vectors that provide a fine-grained description of a latent semantic model's prediction or some aspects of the prediction. We demonstrated our approach using four latent semantic models, two of which are traditional latent semantic models (Latent Dirichlet Allocation [6, 34] and Word-based Translation Model [4]) used in many IR applications, while the other two are recently proposed latent semantic models (Discriminative Projection Model [16, 36]

Table 6: Change matrix of the relevance labels of the highest ranked documents returned by the combination of the latent semantic models' scores only (w/o MF) and the combination of latent semantic models' scores and metafeatures (w/ MF). The cell in row X and column Y contains the number of queries for which the documents ranked on the top positions by the combinations w/o MF and w/ MF belong to categories X and Y, respectively.

		w/ MF				
		bad	fair	good	excellent	perfect
w/o MF	bad	12694	3139	4111	486	534
	fair	3096	14446	6294	1152	931
	good	2943	4816	38396	2656	830
	excellent	206	301	1141	4607	215
	perfect	118	196	239	63	7593

and Deep Structured Semantic Model [20]) that specialize on the web document ranking task.

Our experimental results show that through the use of metafeatures the performance of a combination of latent semantic models on the document ranking task can be improved by 7.6% and 3.8% in NDCG scores at truncation levels 1 and 10. Moreover, through the use of metafeatures we manage to achieve better performance for each individual latent semantic model by itself than the traditional way of combining all four latent semantic models' scores.

We believe that a latent semantic model's metafeatures along with its score provide a richer representation of the model's prediction than the model's score by itself. We expect metafeatures to be useful in other applications that use the scores of latent semantic models, e.g., online advertising [7], question answering [5], paraphrase detection [11], and textual entailment [31].

As to the limitations of our study, the large size of some metafeatures might increase the chances of overfitting for many machine learning algorithms. To mitigate the issue, dimensionality reduction techniques, such as Principal Component Analysis (PCA) [22], Autoencoder Neural Networks (AENNs) [10], Restricted Boltzmann Machines (RBM) [17] and t-SNE [32], can be considered.

Our approach can be extended to other models whose scores are used as features for a machine learning algorithm. For instance, in web search, a similar approach can be applied to the PageRank model [26]. The score assigned to a document D by the PageRank model is a sum of features over documents that contain a reference to D. Providing information about components of the sum (e.g., mean, variance) might help the global ranker when it attempts to rank documents that have received very similar PageRank scores.

Acknowledgments

We would like to thank Tom Kenter and Ilya Markov for reading the paper and making useful comments and suggestions.

This research was supported by Amsterdam Data Science, the Dutch national program COMMIT, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, and the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, 652.-002.001, 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

7. REFERENCES

[1] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Supervised semantic indexing. In *CIKM 2009*, pages 187–196. ACM, 2009.

[2] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information Retrieval*, 13(3):291–314, 2010.

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[4] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *SIGIR 1999*, pages 222–229. ACM, 1999.

[5] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR 2000*, pages 192–199. ACM, 2000.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Machine Learning Research*, 3:993–1022, 2003.

[7] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *SIGIR 2007*, pages 559–566. ACM, 2007.

[8] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. *Int. J. Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

[9] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6): 391–407, 1990.

[10] D. DeMers, G. Cottrell, et al. Non-linear dimensionality reduction. In *NIPS 1993*, pages 580–580, 1993.

[11] B. Dolan, C. Quirk, and C. Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004*, pages 350–356, 2004.

[12] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[13] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

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

[15] J. Gao, X. He, and J.-Y. Nie. Clickthrough-based translation models for web search: from word models to phrase models. In *SIGIR 2010*, pages 1139–1148. ACM, 2010.

[16] J. Gao, K. Toutanova, and W.-t. Yih. Clickthrough-based latent semantic models for web search. In *SIGIR 2011*, pages 675–684. ACM, 2011.

[17] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[18] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR 1999*, pages 50–57. ACM, 1999.

[19] V. Hollink, J. Kamps, C. Monz, and M. de Rijke. Monolingual document retrieval for European languages. *Information Retrieval*, 7: 33–52, 2004.

[20] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM 2013*, pages 2333–2338. ACM, 2013.

[21] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR 2000*, pages 41–48. ACM, 2000.

[22] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.

[23] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR 2001*, pages 111–119. ACM, 2001.

[24] H. Li and J. Xu. Semantic matching in search. *Foundations and Trends in Information Retrieval*, 7(5):343–469, 2014.

[25] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report 1999-66, Stanford InfoLab, 1999.

[27] M. J. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[28] R. Salakhutdinov and G. Hinton. Semantic hashing. *International*

Journal of Approximate Reasoning, 50(7):969–978, 2009.

[29] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM 2014*, pages 101–110. ACM, 2014.

[30] K. M. Svore and C. J. Burges. A machine learning approach for improved BM25 retrieval. In *CIKM 2009*, pages 1811–1814. ACM, 2009.

[31] P. D. Turney. Measuring semantic similarity by latent relational analysis. In *IJCAI'05*, pages 1136–1141, 2005.

[32] L. Van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *J. Machine Learning Research*, 9:2579–2605, 2008.

[33] Q. Wang, J. Xu, H. Li, and N. Craswell. Regularized latent semantic indexing. In *SIGIR 2011*, pages 685–694. ACM, 2011.

[34] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR 2006*, pages 178–185. ACM, 2006.

[35] W. Wu, Z. Lu, and H. Li. Learning bilinear model for matching queries and documents. *The Journal of Machine Learning Research*, 14(1):2519–2548, 2013.

[36] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek. Learning discriminative projections for text similarity measures. In *CoNLL*, pages 247–256. ACL, 2011.