Estimating Embedding Vectors for Queries

Hamed Zamani
Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
zamani@cs.umass.edu

W. Bruce Croft
Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
croft@cs.umass.edu

ABSTRACT

The dense vector representation of vocabulary terms, also known as word embeddings, have been shown to be highly effective in many natural language processing tasks. Word embeddings have recently begun to be studied in a number of information retrieval (IR) tasks. One of the main steps in leveraging word embeddings for IR tasks is to estimate the embedding vectors of queries. This is a challenging task, since queries are not always available during the training phase of word embedding vectors. Previous work has considered the average or sum of embedding vectors of all query terms (AWE) to model the query embedding vectors, but no theoretical justification has been presented for such a model. In this paper, we propose a theoretical framework for estimating query embedding vectors based on the individual embedding vectors of vocabulary terms. We then provide a number of different implementations of this framework and show that the AWE method is a special case of the proposed framework. We also introduce pseudo query vectors, the query embedding vectors estimated using pseudo-relevant documents. We further extrinsically evaluate the proposed methods using two well-known IR tasks: query expansion and query classification. The estimated query embedding vectors are evaluated via query expansion experiments over three newswire and web TREC collections as well as query classification experiments over the KDD Cup 2005 test set. The experiments show that the introduced pseudo query vectors significantly outperform the AWE method.

CCS Concepts

•Information systems → Query representation; Query reformulation;

Keywords

Word embedding; query embedding vector; pseudo query vector; query expansion; query classification

ICTIR '16, September 12-16, 2016, Newark, DE, USA

DOI: http://dx.doi.org/10.1145/2970398.2970403

1. INTRODUCTION

Computing semantic similarity between vocabulary terms has been an important issue in natural language processing (NLP) and information retrieval (IR) for many years. Many approaches, such as latent semantic indexing [10] and the information content-based method [31], have been proposed to capture semantically similar terms. Recent developments in distributed semantic representations based on dense vectors, also called word embeddings, have been proven to be highly effective in many NLP tasks, such as word analogy [27] and named-entity recognition [11]. Word embedding techniques are unsupervised learning algorithms that assign each term a low-dimensional (compared to the vocabulary size) vector in a "semantic vector space". In this space, the embedding vectors of semantically or syntactically similar terms are designed to be close to each other. Word2vec [27] and GloVe [29] are examples of successful implementations of word embedding vectors. Word2vec and GloVe learn the word embedding vectors using neural network-based language model and matrix factorization technique, respectively.

Following the impressive results achieved by word embedding techniques in NLP tasks, these techniques have begun to be studied in IR tasks [28, 35, 39, 40]. However, there are still several issues that need to be addressed in order to effectively use word embeddings in many IR tasks. In this paper, we address one of the main problems in this area: how to compute the embedding vectors of search queries? This is a challenging problem, since (1) search queries are not always available during the training time of embedding vectors, and (2) queries may contain several keywords that do not co-occur with each other frequently in the corpus, which makes training the embedding vectors for such queries problematic. Therefore, previous studies, such as [21, 24, 28, 35, 39], have decided to average or sum the word embedding vectors of all query terms to generate the query embedding vector. However, to the best of our knowledge, there has been no theoretical justification for such a decision.

In this paper, we propose a theoretical framework based on maximum likelihood estimation to estimate query embedding vectors using the individual embedding vectors of vocabulary terms. This framework which is independent of the embedding learning algorithms, maximizes the likelihood of a query language model and the probabilistic distribution over vocabulary terms computed based on the query embedding vector. The proposed framework consists of two main components. The first component is the similarity function that computes the similarity of two given embedding vectors.

The second component is the query language model. We develop our framework based on two different and effective similarity functions: the softmax and the sigmoid transformations of the cosine similarity. For the softmax function, we derive a closed-form formula to estimate query embedding vectors. For the sigmoid function which has recently been introduced to transform embedding similarity scores [37], a gradient-based approach is proposed to maximize the defined objective function. Furthermore, we estimate the query language models (the second component) using maximum likelihood estimation and a pseudo-relevance feedback technique.

An interesting outcome of our framework is that when the similarities of embedding vectors are computed using the softmax function and the query language model is estimated using maximum likelihood estimation, the computed query embedding vector is equivalent to the average or sum of the embedding vectors for all query terms. Therefore, a theoretical justification is introduced for the heuristic method that has been used in previous work [21, 24, 28, 35, 39] to estimate query embedding vectors.

We extrinsically evaluate the estimated query embedding vectors based on different implementations of the proposed framework using two well-known IR tasks: query expansion and query classification. In the query expansion experiments, we consider three standard TREC collections: Associated Press (AP), the TREC Robust Track 2004 collection, and the TREC Terabyte Track 2004-2006 collection (GOV2). In the query classification experiments, we consider the KDD Cup 2005 test set that contains web queries from real users. Our experiments show that the proposed pseudo query vectors (estimating query embedding vector based on pseudo-relevant documents) in general outperform the methods based on maximum likelihood estimation of query language models, significantly.

To summarize, the contributions of this paper include proposing a theoretical framework for estimating query embedding vectors, providing a theoretical justification for a widely used approach to estimate query embedding vectors (i.e., AWE), introducing pseudo query vectors that outperform the existing AWE method, and evaluating different query embedding vectors in two IR tasks.

2. RELATED WORK

In this section, we first review previous studies on word embeddings applied to IR tasks. We further briefly introduce related work on the query expansion and the query classification tasks.

2.1 Word Embedding for IR

Unlike NLP, where word embeddings have been successfully employed in several tasks, word embedding techniques in IR have only recently begun to be studied. The Fisher Vector (FV) [8] is a document representation framework based on continuous word embeddings, that aggregates a non-linear mapping of word embedding vectors into a document-level representation. Although FV was shown to perform better than latent semantic indexing (LSI) [10] in adhoc retrieval, it does not outperform popular IR frameworks, such as the TF-IDF and the divergence from randomness retrieval models. A number of approaches based on word embedding vectors have been proposed to improve retrieval performance. Zheng and Callan [39] proposed a supervised embedding-based term re-weighting technique applied to the language modeling and BM25 retrieval models. BWESG [35], a bilingual word embedding method, has recently been developed and applied to information retrieval. This model learns bilingual embedding vectors from document-aligned comparable corpora. Ganguly et al. [15] considered the semantic similarities between vocabulary terms to smooth document language models. Zuccon et al. [40] proposed to employ word embeddings within the well-known translation model for IR. Sordoni et al. [33] employed word embedding techniques for expanding queries in a supervised manner. They used click-through and task-specific data.

Computing the semantic similarity between documents has been studied for a number of IR-related tasks. Paragraph vector [24] is a popular method that trains embedding vectors for phrases, paragraphs, and even documents. Word mover's distance (WMD) [22] is an interesting approach that measures the minimum traveling distance from the embedded words of one document to another one. This approach was shown to be effective in document classification. Kenter and de Rijke [20] proposed a supervised approach to compute semantic similarity between short texts. Training word embedding vectors based on additional data, like query logs and click-through data, was also studied in [16, 17, 18], which are out of the scope of this paper. More recently, Diaz et al. [12] proposed to train word embedding vectors on topically-constrained corpora, instead of large topically-unconstrained corpora. These locally-trained embedding vectors were shown to perform well for the query expansion task.

While query embedding vectors play a key role in a number of the aforementioned methods, such as [35, 39], they only considered the average or sum of word embedding vectors of all query terms as the query vector. Therefore, estimating accurate query embedding vectors can improve the performance of many of the embedding-based methods that need to compute query vectors. It should be noted that in a realistic case, queries are not available during the training time of embedding vectors. This makes training of query embedding vectors problematic. In this paper, we focus on estimating embedding vectors for queries based on the embedding vectors of individual terms.

2.2 Query Expansion

Query expansion is the process of adding relevant terms to a query to improve the retrieval performance. There are a number of query expansion methods based on linguistic resources such as WordNet, but they have not substantially improved the retrieval performance [34]. Although a number of data-driven query expansion methods, such as [1], can improve the average retrieval performance, they can be unstable across queries [9]. Therefore, although query expansion is not a new technique for improving retrieval effectiveness, it is still a challenging task [7]. As suggested by Xu and Croft [36], query expansion techniques can use global or local document analysis. Global analysis often relies on external resources or document collections, such as a similarity thesaurus [30] and Wikipedia [13]. On the other hand, local analysis expands queries using the documents that are related to them, such as the top-retrieved documents. This kind of query expansion is called pseudo-relevance feedback (PRF). PRF has been proven to be highly effective in improving retrieval performance [23, 36, 38].

2.3 Query Classification

Query classification, also known as query categorization, has the goal of classifying search queries into a number of pre-defined categories. Two types of classification have been studied. In the first, the labels are query types, such as navigational queries, informational queries, and transactional queries, e.g., [4, 25]. The other task is classifying queries based on their topics. Early work only considered local information about queries, i.e., the terms of queries [2, 3]. More recently, a number of methods were proposed to enrich queries using external information, such as the top-retrieved documents [14, 32] and query context [6]. Given the importance of the query classification task in web search, the 2005 ACM Knowledge Discovery and Data Mining competition (called, KDD Cup 2005) [26] focused on this task: classifying the queries issued by real web users based on query topics. Successful submissions in this competition used search engines to retrieve relevant documents for enriching initial queries [26]. The usefulness of the query classification task for web search has been shown in the literature, e.g., [19].

3. QUERY EMBEDDING VECTORS

Estimating accurate query models is a crucial component in every retrieval framework. It has been extensively studied in existing retrieval models and several approaches have been proposed for estimating query models, especially in the language modeling framework. On the other hand, while word embedding techniques have been shown to be highly effective in capturing semantic similarities in many NLP tasks, their usefulness in IR tasks is still relatively unstudied. In this paper, we focus on query representation in the embedding semantic space: how to estimate accurate embedding vectors for queries? More formally, let E denote a set of d-dimensional embedding vectors for each vocabulary term w. Given a query $q = \{w_1, w_2, \cdots, w_k\}$ with the length of k, the problem is to estimate a d-dimensional vector $\vec{q}$, henceforth called *query embedding vector*, for the query q.

In this section, we propose a probabilistic framework to estimate query embedding vectors based on maximum likelihood estimation. The main idea behind our approach is to maximize the likelihood of an accurate query language model and a probabilistic distribution that can be calculated using the embedding semantic space for each query vector. To do so, let $\delta(\cdot, \cdot)$ denote a similarity function that computes the similarity between two embedding vectors. Hence, the probability of each term w given a query vector $\vec{q}$, henceforth called the *query embedding distribution*, can be calculated as:

$$p(w|\vec{q}) = \frac{\delta(\vec{w}, \vec{q})}{Z} \tag{1}$$

where $\vec{w} \in E$ denotes the embedding vector of the given term w. The normalization factor Z can be calculated as follows:

$$Z = \sum_{w' \in V} \delta(\vec{w'}, \vec{q}) \tag{2}$$

where V denotes the set of all vocabulary terms.[1] On the other hand, assume that there is a query language model θ_q for the query q, that shows how much each word contributes to the query. Our claim is that a query embedding vector $\vec{q^*}$ is a proper query embedding vector, if the query embedding distribution (see Equation (1)) is "close to" the query language model θ_q. In other words, our purpose is to find a query embedding vector that maximizes the following log-likelihood function:

$$\vec{q^*} = \arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log p(w|\vec{q}) \tag{3}$$

The high computational complexity of the normalization factor in calculating the query embedding distribution (see Equation (2)) makes optimizing the log-likelihood function expensive. Note that since the normalization factor Z depends on the query embedding vector, it cannot be computed offline. Therefore, similar to many other optimization problems, we need to relax our objective function. To this end, we assume that the normalization factor Z in Equation (1) is equal for all query vectors. Although this simplifying assumption is not true, our observations indicate that this is not a harmful assumption. To give an intuition about the validity of this assumption, we consider many ($> 200,000$) random query vectors and calculate the normalization factor Z for all of these query vectors. The mean and standard deviation for all Z values are $(1.7 \pm 0.15)^{*10^4}$. The mean value is an order of magnitude larger than the the standard deviation, and this shows that most of the Z values are close to the mean value, which indicates that our assumption is reasonable. It is worth noting that all the following calculations can be done without this assumption, but with high computational cost.

Therefore, based on this relaxation, we can re-write our objective function as follows:

$$\arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log \delta(\vec{w}, \vec{q}) \tag{4}$$

As shown in Equation (4), our framework consists of two main components: the query language model θ_q and the similarity function δ. Since the output of $\delta(\vec{w}, \vec{q})$ is dependent on the query vector $\vec{q}$, the function δ can affect the way that we can optimize the objective function. In the following subsection, we solve this optimization problem for two effective similarity functions. We further discuss how we calculate the query language models.

3.1 Similarity Functions

There are several similarity metrics for word embedding vectors, including the cosine similarity, which have been considered for NLP tasks, e.g., [11, 27]. In the following, we describe two different functions to compute the similarity between embedding vectors. We also explain how our objective function can be optimized based on these functions.[2]

3.1.1 Softmax Function

The similarity function δ for computing the similarity between two embedding vectors can be calculated using the softmax function as follows:

$$\delta(\vec{w}, \vec{w'}) = \exp\left(\frac{\sum_{i=1}^{d} \vec{w}_i \vec{w'}_i}{\|\vec{w}\| \|\vec{w'}\|}\right) \tag{5}$$

[1] In this paper, we assume that the embedding vectors of all query terms are available.

[2] These similarity functions are transformations of the cosine similarity. Since the results obtained by the cosine similarity (without transformation) are substantially lower than those achieved by these transformations, for the sake of space, we do not consider the cosine similarity itself.

where d denotes the dimension of embedding vectors. Without loss of generality, assume that the embedding vectors of all vocabulary terms are unit vectors, and thus their norms are equal to 1. Therefore, our objective function (see Equation (4)) can be re-written as follows:

$$\arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q).\frac{\sum_{i=1}^{d} \vec{w}_i \vec{q}_i}{\|\vec{q}\|} \tag{6}$$

To make this objective function even simpler, we add a constraint to force the query vector be a unit vector. In other words, we consider the following constraint: $\|q\| = 1$. Based on this constraint, we obtain the Lagrange function as follows:

$$\mathcal{L}(\vec{q}, \lambda) = \sum_{w \in V} \left(p(w|\theta_q) \sum_{i=1}^{d} \vec{w}_i \vec{q}_i \right) + \lambda \left(1 - \sum_{i=1}^{d} (\vec{q}_i)^2 \right) \tag{7}$$

where λ denotes the Lagrange multiplier. Using the mathematical optimization method of Lagrange multipliers, we compute the first derivatives of the Lagrange function as follows:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \vec{q}_i} = \sum_{w \in V} \vec{w}_i p(w|\theta_Q) - 2\lambda \vec{q}_i \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_{i=1}^{d} (\vec{q}_i)^2 \end{cases} \tag{8}$$

where $\vec{q}_i$ denotes the i^{th} element of the query vector $\vec{q}$. By setting the above partial derivatives to zero, we can find the stationary point of our objective function as below:

$$\vec{q}_i = \frac{\sum_{w \in V} \vec{w}_i p(w|\theta_Q)}{\sqrt{\sum_{j=1}^{d} (\sum_{w \in V} \vec{w}_i p(w|\theta_Q))^2}} \tag{9}$$

Therefore, the query embedding vector can be calculated using the above closed-form formula.

3.1.2 Sigmoid Function

In this subsection, we consider the sigmoid function as another mapping function for transforming the cosine similarity scores, which was proposed by Zamani and Croft [37]. The similarity function δ can be computed based on the sigmoid function as follows:

$$\delta(\vec{w}, \vec{w'}) = \frac{1}{1 + \exp\left(-a \left(\frac{\sum_{i=1}^{d} \vec{w}_i \vec{w'}_i}{\|\vec{w}\| \|\vec{w'}\|} - c \right) \right)} \tag{10}$$

where a and c are two free parameters. Similar to the previous subsection, without loss of generality, we can assume that embedding vectors of all vocabulary terms are unit vectors. Therefore, the objective function in this case is calculated as below:

$$\arg\max_{\vec{q}} \sum_{w \in V} -p(w|\theta_q) \log \left(1 + \exp\left(-a(\frac{\sum_{i=1}^{d} \vec{w}_i \vec{q}_i}{\|\vec{q}\|} - c) \right) \right) \tag{11}$$

The partial derivative of the above objective function (called $\mathcal{F}$) with respect to each element of the query embedding vector is computed as:

$$\frac{\partial \mathcal{F}}{\partial \vec{q}_i} = \sum_{w \in V} p(w|\theta_q) \left(a(\frac{\vec{w}_i}{\|\vec{q}\|} - \frac{\vec{q}_i \sum_{j=1}^{d} \vec{w}_j \vec{q}_j}{\|\vec{q}\|^3})(1 - \delta(\vec{w}, \vec{q})) \right) \tag{12}$$

where δ is the sigmoid-based similarity function as shown in Equation (10). Query embedding vectors can be now estimated using gradient-based optimization methods. Note that this objective function is not convex, and thus the initial value of the query embedding vectors can affect the results.

3.2 Estimating Query Language Model

As described earlier, the query language model is the other component in our framework to compute the query embedding vectors. Several approaches have been proposed to estimate query language models. In this subsection, we introduce two of these methods that have been widely explored in the language modeling literature.

3.2.1 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a simple yet effective approach for estimating query language models. MLE for a given query q can be calculated by relative counts:

$$p(w|\theta_q) = \frac{\text{count}(w, q)}{|q|} \tag{13}$$

where θ_q, $\text{count}(w, q)$, and $|q|$ denote the unigram query language model, the count of term w in the query q, and the query length, respectively.

3.2.2 Pseudo-Relevance Feedback

Pseudo-relevance feedback (PRF) has been shown to be highly effective at improving retrieval effectiveness. PRF in the language modeling framework estimates a query language model from a small set of top-retrieved documents. In PRF, in addition to updating the weights of query terms, a number of new terms will be added to the query. In this paper, we consider the relevance model (with the i.i.d. sampling assumption) [23], a state-of-the-art PRF method, to estimate the query language model as follows:

$$p(w|\theta_q) \propto \sum_{d \in F} p(w|d) \prod_{w' \in q} p(w'|d) \tag{14}$$

where F denotes the set of feedback documents. The probability of each term in the document (e.g., $p(w|d)$) can be computed by smoothing the maximum likelihood estimation probability. The top m terms with highest probabilities are usually selected in the feedback language models. We call the query embedding vectors estimated using the PRF distributions, *pseudo query vectors* (PQV).

4. DISCUSSION

Average word embedding (AWE) is a popular method to estimate query embedding vectors. Although AWE has been shown to be effective in previous studies, this method of embedding vector construction was ad-hoc. In this section, we first show that AWE is a special case of the proposed framework, and thus there is a theoretical justification for this very simple approach. We further compare the two ways of optimizing the defined objective function based on the softmax and the sigmoid functions, respectively.

4.1 Special Case: Average Word Embedding

AWE, averaging or summing the embedding vectors of all query terms, has been previously used to construct embedding vectors of queries [21, 24, 28, 35, 39]. In this subsection, we show that AWE is a special case of the proposed theoretical framework.

Table 1: Statistics of the collections employed in the query expansion experiments.

ID	collection	queries (title only)	#docs	doc length	#qrels
AP	Associated Press 88-89	TREC 1-3 Ad-Hoc Track, topics 51-200	165k	287	15,838
Robust	TREC Disks 4 & 5 minus Congressional Record	TREC 2004 Robust Track, topics 301-450 & 601-700	528k	254	17,412
GOV2	2004 crawl of .gov domains	TREC 2004-2006 Terabyte Track, topics 701-850	25,205k	648	26,917

Assume that we consider the softmax function as the similarity function to compute the similarity between two embedding vectors. As shown in Subsection 3.1.1, each element of the query embedding vector can be calculated using Equation (9). In this equation, the denominator is just a normalization factor to force the embedding vector be a unit vector. Now, assume that the query language model θ_q is estimated using maximum likelihood estimation, as explained in Subsection 3.2.1. Therefore, the query embedding vector is calculated as:

$$\vec{q}_i \propto \sum_{w \in q} \frac{\text{count}(w, q)}{|q|} . \vec{w}_i \tag{15}$$

where $\vec{q}_i$ denotes the i^{th} element of the query vector. In the above equation, the summation is just over the query terms, since the count of other terms is equal to zero, and thus they will not affect the result. The above equation is equivalent to AWE. To summarize, when the similarity of embedding vectors is calculated using the softmax function and the query language model is estimated using maximum likelihood estimation, the proposed framework will produce the AWE method.

4.2 Softmax vs. Sigmoid

In this subsection, we discuss the differences between using the softmax and the sigmoid functions in estimating query vectors. As calculated above, we come up with a closed-form formula for query embedding vectors when using the softmax function. This formula is easy to compute and very efficient, especially when only a limited number of terms have non-zero probability in the query language model θ_q. In addition, the calculated query vector is the global optimum solution for the defined objective function. The method based on the sigmoid function is an iterative gradient-based method. Since the objective function in this case is not convex, achieving the global optimum solution is not guaranteed. It should be noted that the objective function depends on the similarity function (see Equation (4)) and the local optimum solution of the objective function based on the sigmoid function might produce a better query vector compared to the global optimum of the one with the softmax function.

Furthermore, there are two free parameters involved in the sigmoid function, which make it more flexible than the softmax function. We expect that this flexibility leads to better performance for near-optimal parameter settings.

5. EXPERIMENTS

In this section, we extrinsically evaluate the proposed query embedding vectors in two well-known IR tasks: query expansion and query classification. In all experiments, we employed the word embedding vectors computed using the GloVe method [29]. The word embedding vectors were extracted from a 6 billion token collection (Wikipedia 2014 plus Gigawords 5), unless otherwise stated.[3] The dimension of embedding vectors are set to 300 in all experiments, except those related to studying the sensitivity of methods to the embedding dimension. In the experiments related to the pseudo query vector (PQV) methods (estimating query embedding vectors using pseudo-relevance feedback language models), the number of feedback documents is set to 10. For optimization of the sigmoid-based query embedding vector estimation, we employed the MALLET[4] implementation of the limited-memory BFGS algorithm (LBFGS) [5]. We used the MLE+Softmax method (AWE) as the initial query vector for the sigmoid-based methods.

In the following, we first employ query word embeddings for query expansion. We further consider the constructed query embedding vectors for the query classification task.

5.1 Evaluation via Query Expansion

In the first set of extrinsic evaluations, we consider query word embeddings for expanding query models in the language modeling framework. In the following, we first briefly explain how we expand query language models using the query word embeddings and then introduce our experimental setup. We afterward report and discuss the results.

5.1.1 Query Expansion Using Query Vectors

In these experiments, we use the language modeling framework. To expand the query language models, we first calculate the probability of each term given the query vector using Equation 1. We then linearly interpolate this probability with the maximum likelihood estimation of the original query, as follows:

$$p(w|\theta_q^*) = \alpha \; p_{mle}(w|\theta_q) + (1 - \alpha) \; p(\vec{w}|\vec{q}) \tag{16}$$

where α denotes the interpolation coefficient and controls the weight of the original query language model. We consider the top m terms with highest probabilities in θ_q^* as the expanded query language model.

5.1.2 Experimental Setup

To evaluate the proposed query embedding vectors in the query expansion task, we used three standard TREC collections: AP (Associated Press 1988-1989), Robust (TREC Robust Track 2004 collection), and GOV2 (TREC Terabyte Track 2004-2006 collection). The first two collections contain high-quality news articles and the last one is a large web collection. We report the statistics of these collections in Table 1. The titles of topics are considered as queries. In the experiments, we only considered the queries where the embedding vectors of all terms are available. Therefore, 146 out of 150, 241 out of 250, and 147 out of 150 queries were considered for AP, Robust, and GOV2, respectively.

[3]We consider this corpus, since it is a relatively large corpus containing formal texts with a diverse vocabulary set.
[4]http://mallet.cs.umass.edu/

Table 2: Comparing the proposed query embedding vectors via query expansion. The superscripts 0/1/2 denote that the improvements over QL/MLE+Softmax/MLE+Sigmoid are statistically significant.

Collection	Metric	QL	MLE+Softmax (AWE)	MLE+Sigmoid	PQV+Softmax	PQV+Sigmoid
AP	MAP	0.2236	0.2470^{0}	0.2486^{0}	0.2695^{012}	$\mathbf{0.2717}^{012}$
	P@5	0.4260	0.4452^{0}	0.4507^{0}	0.4493^{0}	$\mathbf{0.4548}^{01}$
	P@10	0.4014	0.4260^{0}	$\mathbf{0.4274}^{0}$	0.4226^{0}	0.4233^{0}
Robust	MAP	0.2190	0.2299^{0}	0.2303^{0}	0.2355^{012}	$\mathbf{0.2364}^{012}$
	P@5	0.4606	$\mathbf{0.4730}^{0}$	0.4714^{0}	0.4564	0.4591
	P@10	0.3979	0.4237^{0}	$\mathbf{0.4245}^{0}$	0.4083^{0}	0.4141^{0}
GOV2	MAP	0.2696	0.2719	0.2727	0.2771^{012}	$\mathbf{0.2798}^{012}$
	P@5	0.5592	0.5837^{0}	$\mathbf{0.5864}^{0}$	0.5755^{0}	$\mathbf{0.5864}^{0}$
	P@10	0.5531	0.5653^{0}	$\mathbf{0.5721}^{01}$	0.5694^{0}	$\mathbf{0.5721}^{01}$

The standard INQUERY stopword list was employed in all experiments, and no stemming was performed.

We considered the KL-divergence retrieval model with the Dirichlet prior smoothing method. All experiments were carried out using the Galago toolkit[5].

Parameters Setting. In all the experiments, the Dirichlet prior smoothing parameter μ was set to Galago's default value of 1500. In all the experiments (unless otherwise stated), the parameters α (the linear interpolation coefficient), m (the number of terms added to each query), and n (the number of feedback terms considered in the PQV methods) were set using 2-fold cross-validation over the queries in each collection. We sweeped the parameter α between $\{0.1, \ldots, 0.9\}$. The values of parameters m and n were also selected from $\{10, 20, ..., 100\}$. The sigmoid parameters (i.e., a and c in Equation 10) were also set using the same procedure from the $[0, 50]$ and $[0.8, 0.9]$ intervals, respectively.

Evaluation Metrics. Mean Average Precision (MAP) of the top-ranked 1000 documents and the precision of the top 5 and 10 retrieved documents (P@5 and P@10) are used to evaluate the retrieval effectiveness. Statistically significant differences of performances are determined using the two-tailed paired t-test computed at a 95% confidence level.

5.1.3 Results and Discussion

In this subsection, we first evaluate the proposed query embedding vectors. We further study the parameters sensitivity of the methods. Finally, we study the sensitivity of the methods to different word embedding vectors. In our experiments, we evaluate all combinations of using softmax vs. sigmoid as similarity function and MLE vs. PRF (PQV) as the query language model. We compare our results with those obtained by the standard maximum likelihood estimation of query language models without query expansion (QL). As explained in Subsection 4.1, the MLE+Softmax method is equivalent to the AWE method, previously used in [21, 24, 35]. Since the contribution of this paper is to estimate accurate query embedding vectors, not proposing the most effective query expansion technique, we do not compare our methods with a large number of state-of-the-art query expansion methods. It should be also noted that queries are not accessible during the training time of word embedding vectors, which makes training of query embedding vectors problematic.

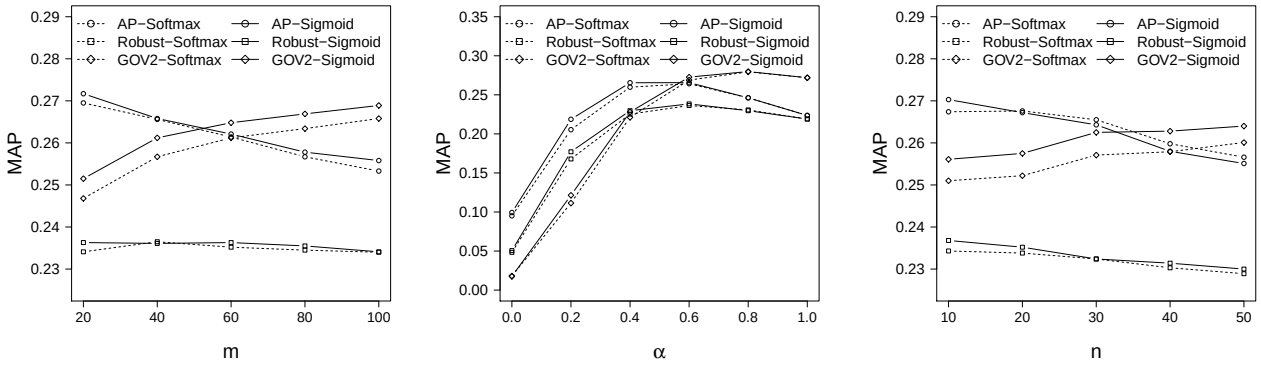
The results of the proposed methods are reported in Table 2. According to this table, the results achieved by the embedding-based query expansion methods outperforms QL in all collections, in terms of MAP. These improvements are often statistically significant, especially in the newswire collections (AP and Robust). In GOV2, expanding query language models using both PQV methods significantly outperforms the QL baseline, in terms of MAP. The relative and absolute MAP improvements in the newswire collections are higher than those in the GOV2 collection. The reason could be related to the word embedding vectors that we used in our experiments. As explained in the beginning of Section 5, the embedding vectors were extracted from the Wikipedia and the Gigawords corpora that contain formal texts. At the end of this subsection, we report the results achieved by word embedding vectors extracted from other corpora. The proposed query expansion methods also outperform QL, in terms of P@5 and P@10, in most cases.

The results achieved by the MLE methods (i.e., MLE+Softmax (AWE) and MLE+Sigmoid) show that MLE+Sigmoid always outperforms MLE+Softmax in terms of the considered evaluation metrics (except in terms of P@5 in Robust). Similar behaviour can be observed by comparing the results achieved by PQV+Softmax and PQV+Sigmoid, i.e., PQV+Sigmoid always outperforms PQV+Softmax in terms of MAP, P@5 and P@10. It is notable that although the improvements of the sigmoid function over the softmax function are not statistically significant, the sigmoid-based methods consistently outperform the softmax-based methods, in terms of all evaluation metrics.

Comparing the results achieved by the PQV methods with those obtained by the MLE methods highlights substantial and statistically significant improvements between employing the aforementioned query language model distributions (i.e., MLE and PRF) in estimating the query embedding vectors. In other words, the PQV methods significantly outperform the MLE methods in all the collections, in terms of MAP. However, this is not the case for precision at top-retrieved documents. The MLE methods are better than the PQV methods in some cases, in terms of P@5 and/or P@10. In particular in the Robust collection, the MLE methods perform well, in terms of P@5 and P@10. These results indicate that using PRF distributions to estimate query embedding vectors can successfully improve the overall retrieval performance, but sometimes it harms the precision of the top-ranked documents.

Parameters Sensitivity. To study the sensitivity of the proposed methods to the free parameters, we set the parameters m (the number of expansion terms), α (the interpolation coefficient in Equation (16)), and n (the number of feedback terms considered to estimate PQVs) to 50, 0.5, and 10, respectively. In each set of experiments, we sweep one

[5]http://www.lemurproject.org/galago.php



(a) # expansion terms (b) interpolation coefficient (c) # feedback terms

Figure 1: Sensitivity of PQV+Softmax and PQV+Sigmoid to the number of expansion terms (m), the interpolation coefficient (α), and the feedback term count used to estimate query embedding vectors (n), in terms of MAP.

of these parameters and plot the performance of methods, in terms of MAP. The results are plotted in Figure 1. For the sake of visualization, we only plot the results of PQV methods, but MLE methods also behave similarly.

According to Figure 1a, the behaviour of performance changes with respect to the number of expansion terms (m) completely depends on the retrieval collection. For instance, by increasing the number of expansion terms in GOV2, the performance significantly increases; while in AP, the performance drops. In Robust, the performance in general slightly decreases by increasing the number of expansion terms, but the performance changes are not significant. The results indicate that in the newswire collections (AP and Robust), a few top expansion terms are highly relevant to the query; while in the web collection (GOV2) there are a number of good expansion terms that are not very close to the estimated query embedding vector, and thus more expansion terms are needed.

Figure 1b plots the sensitivity of the methods to the parameter α. According to this plot, the PQV methods behave similarly in the newswire collections: they achieve their best performance when α is set to 0.6. This means that the weight of the generated language model using the estimated query embedding vectors should be close to the weight of the original query language model. In contrast, in the GOV2 collection, the PQV methods achieve their best results when α is equal to 0.8. This means that more weights should be given to the original query language model. Therefore, quality of the generated language model using query embedding vectors in AP and Robust might be higher than in GOV2.

According to Figure 1c, the number of feedback terms needed to estimate query embedding vectors in the newswire collections is much lower than those needed in the web collection. This plot again shows that the methods behave similarly in the newswire collections. All plots in Figure 1 show that the results achieved by the PQV method based on the sigmoid function in most cases are higher than those obtained based on the softmax function. These differences are higher in the GOV2 collection. This shows that while there is no guarantee to find global optimum solution for the sigmoid-based methods, they generally outperform the softmax-based methods.

Sensitivity to the Embedding Vectors. In this set of experiments, we study the sensitivity of PQV+Sigmoid (the method with highest MAP values) to the employed embedding vectors. We first train embedding vectors with different dimensions on the same corpus (Wikipedia 2004 & Gigawords 5). The achieved MAP values are reported in Table 3. According to this table, increasing the dimension of embedding vectors improves the retrieval performance. The MAP differences are statistically significant in many cases. This shows that the embedding vectors with very dense representations cannot be optimal for capturing semantic similarities in the proposed PQV+Sigmoid method.

Table 3: Sensitivity of PQV+Sigmoid to the dimension of embedding vectors in query expansion, in terms of MAP. The superscripts 1/2/3 denote that the MAP improvements over the dimensions of 50/100/200 are significant.

Collection	Dimension			
	50	100	200	300
AP	0.2455	0.2604^{1}	0.2648^{1}	$\mathbf{0.2717}^{123}$
Robust	0.2305	0.2321	0.2351^{12}	$\mathbf{0.2364}^{12}$
GOV2	0.2751	0.2762	0.2786^{12}	$\mathbf{0.2798}^{12}$

Table 4: The corpora used for training the embedding vectors.

ID	Corpus	#tokens	#vocab.
Wiki	Wikipedia 2004 & Gigawords 5	6b	400k
Web 42b	Web crawl	42b	1.9m
Web 840b	Web crawl	840b	2.2m

To analyze the robustness of the proposed methods to the choices made in training the word embedding vectors, we consider three different corpora: Wiki, Web 42b, and Web 840b. The statistics of these corpora are reported in Table 4.[6] The dimension of embedding vectors is set to 300. The MAP values achieved by employing each of the embedding vectors trained on different corpora are reported in Table 5. Note that to have fair comparisons, we only consider the queries that the embedding vectors of all their query terms are available in all the embedding vector sets. The number of queries that are used for evaluation is mentioned in Table 5. Interestingly, although both web crawl corpora are much larger than the Wiki corpus, the PQV+Sigmoid method achieves its best performance in the newswire collections when the embedding vectors are extracted from the

[6]The embedding vectors are freely available at http://nlp.stanford.edu/projects/glove/.

Table 5: Sensitivity of PQV+Sigmoid to the embedding corpus in query expansion, in terms of MAP. The superscripts 1/2/3 denote that the MAP improvements over Wiki/Web 42b/Web 840b are statistically significant.

Collection	Wiki	Web 42b	Web 840b
AP (146 queries)	**0.2717**^{3}	0.2644	0.2602
Robust (240 queries)	**0.2365**^{2}	0.2329	0.2347
GOV2 (146 queries)	0.2788	0.2765	**0.2795**

Wiki corpus. Conversely, for the GOV2 collection, employing the embedding vectors trained on the Web 840b corpus leads to the best performance. The reason is that the content of documents in the Wiki corpus is more similar to the news articles, compared to web corpora. Therefore, the type of documents used for training embedding vectors is an important factor which should be taken into account.

5.2 Evaluation via Query Classification

In this subsection, we extrinsically evaluate the estimated query embedding vectors in the query classification task. The task is to assign each query a number of labels (categories). Labels are pre-defined and some training data are available for each label.

To classify each query, we consider a very simple approach based on query embedding vectors. We first compute the probability of each category/label given each query q and then select the top t categories with highest probabilities. In fact, this method is based on k-nearest neighbors classification. The probability $p(C_i|q)$ can be easily computed using the following formula:

$$p(C_i|q) = \frac{\delta(\vec{C_i}, \vec{q})}{\sum_j \delta(\vec{C_j}, \vec{q})} \propto \delta(\vec{C_i}, \vec{q}) \quad (17)$$

where C_i denotes the i^{th} category. $\vec{C_i}$ is the centroid vector of all query embedding vectors with the label of C_i. In the above equation, we drop the normalization factor, since it is the same for all categories. For PQV methods, we linearly interpolate the above probability with those computed using the MLE methods with the interpolation of α.

5.2.1 Experimental Setup

We consider the dataset that was previously employed to evaluate the query classification approaches submitted to the KDD Cup 2005: Internet user search query categorization [26]. This evaluation set contains 800 web queries that were issued by real users. These queries were randomly selected and do not contain junk and non-English words/phrases. The queries were tagged by three individual human editors. The KDD Cup 2005 organizers pre-defined 67 categories (labels) and each editor selected up to 5 labels among them for each query. The embedding vectors of all query terms of 700 out of 800 queries are available in our embedding collection. We only consider these 700 queries in our evaluations. The spelling errors in queries are corrected in a pre-processing phase.

In our evaluations, we consider 5-fold cross-validation over the queries and the reported results are the average of all results obtained over the test folds. In each step we have 560 and 140 training and test queries, respectively.

In the experiments related to PQV, we use the Robust collection (see Table 1 for details) to retrieve pseudo-relevant documents. The retrieval details are exactly similar to what were considered in the query expansion experiments (see Subsection 5.1.2). In all experiments, stopwords are removed from queries and no stemming was performed. We employed the INQUERY stopword list.

Table 6: Comparing query embedding vectors via query classification. The superscripts 1/2 denote that the improvements over MLE+Softmax/MLE+Sigmoid are significant. The best result for each evaluation metric is marked by *.

t	Metric	MLE+ Softmax (AWE)	MLE+ Sigmoid	PQV+ Softmax	PQV+ Sigmoid
1	F1	0.2675	0.2657	**0.2802**^{12}	0.2784^{12}
	Prec.	0.5738	0.5700	**0.6009**^{12*}	0.5971^{12}
2	F1	0.3617	0.3590	0.3741^{12}	**0.3752**^{12}
	Prec.	0.4783	0.4747	0.4947^{12}	**0.4961**^{12}
3	F1	0.3916	0.3859	0.4069^{12}	**0.4073**^{12}
	Prec.	0.4106	0.4046	0.4266^{12}	**0.4271**^{12}
4	F1	0.3938	0.3930	0.4142^{12}	**0.4144**^{12*}
	Prec.	0.3589	0.3582	0.3774^{12}	**0.3777**^{12}
5	F1	0.3904	0.3892	**0.4095**^{12}	0.4082^{12}
	Prec.	0.3237	0.3227	**0.3395**^{12}	0.3384^{12}

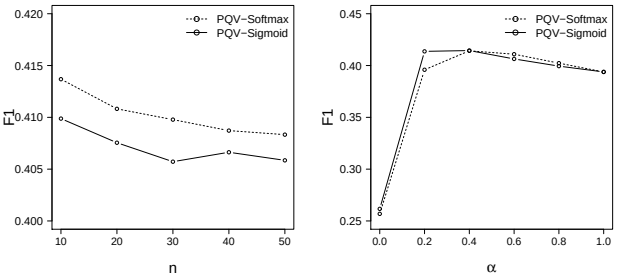
Parameters Setting. In all experiments unless explicitly mentioned, the parameters α (the linear interpolation coefficient), and n (the number of feedback terms considered in the PQV methods) were tuned on the training data. We sweeped the parameter α between $\{0.1, \ldots, 0.9\}$. The values of parameter n were also selected from $\{10, 20, ..., 100\}$.

Evaluation Metrics. We consider two widely used evaluation metrics that were also used in KDD Cup 2005 [26]: precision and F1-measure. Since the labels assigned by the three human editors differ in some cases, all the label sets should be taken into account. We compute these two metrics in the same way as was used to evaluate the KDD Cup 2005 submissions [26]. Statistically significant differences are determined using the two-tailed paired t-test computed at a 95% confidence level.

5.2.2 Results and Discussion

In this subsection, we first evaluate the proposed query embedding vectors. We further study the parameters sensitivity of the proposed methods. Finally, we discuss the results obtained by a number of different word embeddings. Similar to the query expansion experiments, we consider all combinations of softmax vs. sigmoid and MLE vs. PQV for the query classification experiments. As explained earlier, the purpose of this paper is to propose a theoretical framework for estimating query embedding vectors. Thus, we want to extrinsically evaluate different query embedding vectors. Therefore, we do not consider any other baselines.

The results achieved by the proposed methods are reported in Table 6. According to this table, by increasing the number of labels assigned to each query (i.e., t), the precision values decrease in all the methods, which is expected. In other words, assigning only one tag with highest probability achieves the best precision value. Therefore, as shown in Table 6, the best precision is achieved by PQV+Softmax when $t = 1$. Furthermore, the results indicate that PQV methods outperform MLE methods in all cases, in terms of both precision and F1-measure. The F1-measure improvements are always statistically significant. It is an interesting observation that while the PRF distributions in PQV meth-



(a) # feedback terms (b) interpolation coefficient

Figure 2: Sensitivity of PQV methods to the feedback term count used to estimate query embedding vectors (n) and the interpolation coefficient (α), in terms of F1-measure.

ods are estimated using the Robust collection (a newswire collection) and the queries in the KDD Cup 2005 dataset contains general web queries, the pseudo query vectors perform better than the MLE estimations.

In all the proposed methods, the best F1-measure value is achieved when t is equal to 4. The highest F1-measure is obtained by the PQV+Sigmoid method. Although our experimental setup differs from those considered by the KDD Cup 2005 submissions and it is not fair to compare our results with theirs, the best precision and F1-measure values achieved by the proposed method are relatively high and this is a good evidence of the effectiveness of employing word embedding vectors for the query classification task.

Parameters Sensitivity. To study the parameters sensitivity of the proposed PQV methods, we set the number of categories assigned to each query (i.e., t) to 4, where the methods achieve their best F1-measure. In this set of experiments, we set n (the number of feedback terms considered to estimate PQVs) and α (the interpolation coefficient introduced earlier) to 10 and 0.5, respectively. We sweep each parameter and plot the performance of the PQV methods, in terms of F1-measure (the MLE methods do not have these parameters). The results are plotted in Figure 2.

As shown in Figure 2a, by increasing the number of feedback terms for estimating query embedding vectors using PQV, the performance of methods generally decreases. In other words, the best F1-measure is achieved when n is equal to 10. Figure 2b demonstrates that the best performance is achieved when the parameter α is set to 0.4. In fact, when $\alpha = 0$ (the probability of each category is just computed based on the PQV method and it is not interpolated with the one computed based on MLE), the performance is relatively low. The reason could be related to the characteristics of the documents that are used to estimate the PRF distributions (because queries are general web queries, while documents are news articles). However, when this probability is linearly interpolated with the probability obtained based on MLE, the performance increases and the best performance is significantly higher than those obtained by the MLE methods.

Sensitivity to the Embedding Vectors. In this set of experiments, we study the sensitivity of PQV+Sigmoid to the embedding vectors.[7] Similar to the query expansion experiments, we first consider embedding vectors with different dimensions trained on the same corpus (Wikipedia 2004 & Gigawords 5). We set the number of categories per query (t) to 4, where the methods achieve their best F1-measure. The achieved F1-measure values are reported in Table 7. According to this table, increasing the dimension of embedding vectors improves the query classification performance. Similar behaviour can be observed in the query expansion experiments (see Table 3).

[7]For the sake of space, we only consider the method with the best F1-measure (see Table 6).

Table 7: Sensitivity of PQV+Sigmoid to the dimension of embedding vectors in query classification. The superscripts 1/2/3 denote that the improvements over the dimension of 50/100/200 are statistically significant.

Method	Metric	Dimension			
		50	100	200	300
PQV+ Sigmoid	F1	0.3541	0.3886^{1}	0.4123^{12}	$\mathbf{0.4144}^{12}$
	Prec.	0.3227	0.3541^{1}	0.3757^{12}	$\mathbf{0.3777}^{12}$

Table 8: Sensitivity of PQV+Sigmoid to the embedding corpus in query classification. The superscripts 1/2 denote that the improvements over Wiki/Web 42b are significant.

Method	Metric	Wiki	Web 42b	Web 840b
PQV+ Sigmoid	F1	0.4144	0.4284^{1}	$\mathbf{0.4359}^{12}$
	Prec.	0.3777	0.3846^{1}	$\mathbf{0.3972}^{12}$

We further evaluate the proposed methods using the embedding vectors trained on different corpora. The details of these corpora are reported in the query expansion experiments (see Table 4). The results are reported in Table 8. According to this table, the results achieved by the embedding vectors trained on the largest web corpus are higher than those achieved by other embedding vectors. The reason is that the queries were issued by web users and the vectors trained on web collections could provide better semantic representations for this type of queries.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we addressed the problem of estimating dense vector representations for search queries in the embedding semantic space. To this end, we proposed a probabilistic framework to estimate query embedding vectors based on the individual embedding vectors of vocabulary terms. This framework consists of two major components: the similarity function and the query language model. We further developed our framework using two different similarity functions (the softmax and the sigmoid transformations of the cosine similarity) and two well-known query language models (the maximum likelihood estimation and the pseudo-relevance feedback). The proposed framework also provides a theoretical justification for the method that has been heuristically used in previous work: averaging the embedding vectors of all query terms. We extrinsically evaluated the proposed query vectors using two well-known IR tasks: query expansion and query classification. Our extensive query expansion experiments on three newswire and web collections indicate that employing the sigmoid function can consistently outperform the softmax function, although these improvements are not statistically significant. In addition, the embedding vectors estimated using a pseudo-relevance feedback model (called pseudo query vectors) in general improve the other embedding estimations, significantly. This finding was also verified by the query classification experiments over the KDD Cup 2005 test set.

Estimating query embedding vectors can open up many future directions. In this paper, we show that these vectors

can be simply applied in two well-known IR tasks. Employing query vectors in other IR tasks can be studied in future. In addition, we only considered basic query expansion and query classification methods to compare different query embedding vectors. Developing more complex embedding-based approaches to improve the state-of-the-art query expansion and query classification methods is also left for future work. Furthermore, different developments for the components of the proposed framework can be studied in future. For instance, estimating query language models based on mutual-information and more complex language models (bigram, trigram, etc.) can be considered in future studies.

7. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #CNS-0934322. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] J. Bai, J.-Y. Nie, G. Cao, and H. Bouchard. Using Query Contexts in Information Retrieval. In *SIGIR '07*, pages 15–22, 2007.

[2] S. M. Beitzel, E. C. Jensen, O. Frieder, D. Grossman, D. D. Lewis, A. Chowdhury, and A. Kolcz. Automatic Web Query Classification Using Labeled and Unlabeled Training Data. In *SIGIR '05*, pages 581–582, 2005.

[3] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving Automatic Query Classification via Semi-Supervised Learning. In *ICDM '05*, pages 42–49, 2005.

[4] A. Broder. A Taxonomy of Web Search. *SIGIR Forum*, 36(2):3–10, 2002.

[5] J. Byrd, Richard H.and Nocedal and R. B. Schnabel. Representations of Quasi-Newton Matrices and Their Use in Limited Memory Methods. *Math. Program.*, 63(1):129–156, 1994.

[6] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware Query Classification. In *SIGIR '09*, pages 3–10, 2009.

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

[8] S. Clinchant and F. Perronnin. Aggregating Continuous Word Embeddings for Information Retrieval. In *CVSC@ACL '13*, pages 100–109, 2013.

[9] K. Collins-Thompson. Reducing the Risk of Query Expansion via Robust Constrained Optimization. In *CIKM '09*, pages 837–846, 2009.

[10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[11] P. Dhillon, D. P. Foster, and L. H. Ungar. Multi-View Learning of Word Embeddings via CCA. In *NIPS '11*, pages 199–207, 2011.

[12] F. Diaz, B. Mitra, and N. Craswell. Query Expansion with Locally-Trained Word Embeddings. In *ACL '16*, 2016.

[13] H. Fang. A Re-examination of Query Expansion Using Lexical Resources. In *ACL '08*, pages 139–147, 2008.

[14] E. Gabrilovich, A. Broder, M. Fontoura, A. Joshi, V. Josifovski, L. Riedel, and T. Zhang. Classifying Search Queries Using the Web As a Source of Knowledge. *ACM Trans. Web*, 3(2):5:1–5:28, 2009.

[15] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word Embedding Based Generalized Language Model for Information Retrieval. In *SIGIR '15*, pages 795–798, 2015.

[16] M. Grbovic, N. Djuric, V. Radosavljevic, and N. Bhamidipati. Search Retargeting Using Directed Query Embeddings. In *WWW '15*, pages 37–38, 2015.

[17] 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 '13*, pages 2333–2338, 2013.

[18] D. Kang. Query2Vec: Learning Deep Intentions from Heterogeneous Search Logs. Technical report, Carnegie Mellon University, 2015.

[19] I.-H. Kang and G. Kim. Query Type Classification for Web Document Retrieval. In *SIGIR '03*, pages 64–71, 2003.

[20] T. Kenter and M. de Rijke. Short Text Similarity with Word Embeddings. In *CIKM '15*, pages 1411–1420, 2015.

[21] R. Kiros, R. S. Zemel, and R. R. Salakhutdinov. A Multiplicative Model for Learning Distributed Text-Based Attribute Representations. In *NIPS '14*, pages 2348–2356, 2014.

[22] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From Word Embeddings to Document Distances. In *ICML '15*, pages 957–966, 2015.

[23] V. Lavrenko and W. B. Croft. Relevance Based Language Models. In *SIGIR '01*, pages 120–127, 2001.

[24] Q. V. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *ICML '14*, pages 1188–1196, 2014.

[25] U. Lee, Z. Liu, and J. Cho. Automatic Identification of User Goals in Web Search. In *WWW '05*, pages 391–400, 2005.

[26] Y. Li, Z. Zheng, and H. K. Dai. Kdd cup-2005 report: Facing a great challenge. *SIGKDD Explor. Newsl.*, 7(2):91–99, 2005.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*, pages 3111–3119, 2013.

[28] E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana. Improving Document Ranking with Dual Word Embeddings. In *WWW '16*, pages 83–84, 2016.

[29] J. Pennington, R. Socher, and C. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP '14*, pages 1532–1543, 2014.

[30] Y. Qiu and H.-P. Frei. Concept Based Query Expansion. In *SIGIR '93*, pages 160–169, 1993.

[31] P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *IJCAI '95*, pages 448–453, 1995.

[32] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Query Enrichment for Web-query Classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, 2006.

[33] A. Sordoni, Y. Bengio, and J.-Y. Nie. Learning Concept Embeddings for Query Expansion by Quantum Entropy Minimization. In *AAAI '14*, pages 1586–1592, 2014.

[34] E. M. Voorhees. Query Expansion Using Lexical-semantic Relations. In *SIGIR '94*, pages 61–69, 1994.

[35] I. Vulić and M.-F. Moens. Monolingual and Cross-Lingual Information Retrieval Models Based on (Bilingual) Word Embeddings. In *SIGIR '15*, pages 363–372, 2015.

[36] J. Xu and W. B. Croft. Query Expansion Using Local and Global Document Analysis. In *SIGIR '96*, pages 4–11, 1996.

[37] H. Zamani and W. B. Croft. Embedding-based Query Language Models. In *ICTIR '16*, 2016.

[38] C. Zhai and J. Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM '01*, pages 403–410, 2001.

[39] G. Zheng and J. Callan. Learning to Reweight Terms with Distributed Representations. In *SIGIR '15*, pages 575–584, 2015.

[40] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi. Integrating and Evaluating Neural Word Embeddings in Information Retrieval. In *ADCS '15*, pages 12:1–12:8, 2015.