Relevance-based Word Embedding

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

Learning a high-dimensional dense representation for vocabulary terms, also known as a word embedding, has recently attracted much attention in natural language processing and information retrieval tasks. The embedding vectors are typically learned based on term proximity in a large corpus. This means that the objective in well-known word embedding algorithms, e.g., word2vec, is to accurately predict adjacent word(s) for a given word or context. However, this objective is not necessarily equivalent to the goal of many information retrieval (IR) tasks. The primary objective in various IR tasks is to capture *relevance* instead of term proximity, syntactic, or even semantic similarity. This is the motivation for developing unsupervised relevance-based word embedding models that learn word representations based on query-document relevance information. In this paper, we propose two learning models with different objective functions; one learns a relevance distribution over the vocabulary set for each query, and the other classifies each term as belonging to the relevant or non-relevant class for each query. To train our models, we used over six million unique queries and the top ranked documents retrieved in response to each query, which are assumed to be relevant to the query. We extrinsically evaluate our learned word representation models using two IR tasks: query expansion and query classification. Both query expansion experiments on four TREC collections and query classification experiments on the KDD Cup 2005 dataset suggest that the relevance-based word embedding models significantly outperform state-of-the-art proximity-based embedding models, such as word2vec and GloVe.

KEYWORDS

Word representation, neural network, embedding vector, query expansion, query classification

ACM Reference format:
Hamed Zamani and W. Bruce Croft. 2017. Relevance-based Word Embedding. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080831

SIGIR '17, Shinjuku, Tokyo, Japan
© 2017 ACM. 978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080831

1 INTRODUCTION

Representation learning is a long-standing problem in natural language processing (NLP) and information retrieval (IR). The main motivation is to abstract away from the surface forms of a piece of text, e.g., words, sentences, and documents, in order to alleviate sparsity and learn meaningful similarities, e.g., semantic or syntactic similarities, between two different pieces of text. Learning representations for words as the atomic components of a language, also known as word embedding, has recently attracted much attention in the NLP and IR communities.

A popular model for learning word representation is neural network-based language models. For instance, the word2vec model proposed by Mikolov et al. [24] is an embedding model that learns word vectors via a neural network with a single hidden layer. Continuous bag of words (CBOW) and skip-gram are two implementations of the word2vec model. Another successful trend in learning semantic word representations is employing global matrix factorization over word-word matrices. GloVe [28] is an example of such methods. A theoretical relation has been discovered between embedding models based on neural network and matrix factorization in [21]. These models have been demonstrated to be effective in a number of IR tasks, including query expansion [11, 17, 40], query classification [23, 41], short text similarity [15], and document model estimation [2, 31].

The aforementioned embedding models are typically trained based on term proximity in a large corpus. For instance, the word2vec model's objective is to predict adjacent word(s) given a word or context, i.e., a context window around the target word. This idea aims to capture semantic and syntactic similarities between terms, since semantically/syntactically similar words often share similar contexts. However, this objective is not necessarily equivalent to the main objective of many IR tasks. The primary objective in many IR methods is to model the notion of *relevance* [20, 34, 43]. In this paper, we revisit the underlying assumption of typical word embedding methods, as follows:

> *The objective is to predict the words observed in the documents relevant to a particular information need.*

This objective has been previously considered for developing relevance models [20], a state-of-the-art (pseudo-) relevance feedback approach. Relevance models try to optimize this objective given a set of relevant documents for a given query as the indicator of user's information need. In the absence of relevance information, the top ranked documents retrieved in response to the query are assumed to be relevant. Therefore, relevance models, and in general all pseudo-relevance feedback models, use an online setting to obtain training data: retrieving documents for the query and then using the top retrieved documents in order to estimate the relevance distribution. Although relevance models have been proved to be effective in many IR tasks [19, 20], having a retrieval run for each query to obtain the training data for estimating the relevance distribution is not always practical in real-world search engines. We, in this paper, optimize a similar objective in an offline setting, which enables us to predict the relevance distribution without any retrieval runs during the test time. To do so, we consider the top retrieved documents for millions of training queries as a training set and learn embedding vectors for each term in order to predict the words observed in the top retrieved documents for each query. We develop two relevance-based word embedding models. The first one, the relevance likelihood maximization model (RLM), aims to model the relevance distribution over the vocabulary terms for each query, while the second one, the relevance posterior estimation model (RPE), classifies each term as relevant or non-relevant to each query. We provide efficient learning algorithms to train these models on large amounts of training data. Note that our models are unsupervised and the training data is generated automatically.

To evaluate our models, we performed two sets of extrinsic evaluations. In the first set, we focus on the query expansion task for ad-hoc retrieval. In this set of experiments, we consider four TREC collections, including two newswire collections (AP and Robust) and two large-scale web collections (GOV2 and ClueWeb09 - Cat. B). Our results suggest that the relevance-based embedding models outperform state-of-the-art word embedding algorithms. The RLM model shows better performance compared to RPE in the context of query expansion, since the goal is to estimate the probability of each term given a query and this distribution is not directly learned by the RPE model. In the second set of experiments, we focus on the query classification task using the KDD Cup 2005 [22] dataset. In this extrinsic evaluation, the relevance-based embedding models again perform better than the baselines. Interestingly, the query classification results demonstrate that the RPE model outperforms the RLM model, for the reason that in this task, unlike the query expansion task, the goal is to compute the similarity between two query vectors, and RPE can learn more accurate embedding vectors with less training data.

2 RELATED WORK

Learning a semantic representation for text has been studied for many years. Latent semantic indexing (LSI) [8] can be considered as early work in this area that tries to map each text to a semantic space using singular value decomposition (SVD), a well-known matrix factorization algorithm. Subsequently, Clinchant and Perronnin [5] proposed Fisher Vector (FV), a document representation framework based on continuous word embeddings, which aggregates a non-linear mapping of word vectors into a document-level representation. However, a number of popular IR models, such as BM25 and language models, often significantly outperform the models that are based on semantic similarities. Recently, extremely efficient word embedding algorithms have been proposed to model semantic similarly between words.

Word embedding, also known as distributed representation of words, refers to a set of machine learning algorithms that learn high-dimensional real-valued dense vector representation $\vec{w} \in \mathbb{R}^d$ for each vocabulary term w, where d denotes the embedding dimensionality. GloVe [28] and word2vec [24] are two well-known word embedding algorithms that learn embedding vectors based on the same idea, but using different machine learning techniques. The idea is that the words that often appear in similar contexts are similar to each other. To do so, these algorithms try to accurately predict the adjacent word(s) given a word or a context (i.e., a few words appeared in the same context window). Recently, Rekabsaz et al. [30] proposed to exploit global context in word embeddings in order to avoid topic shifting.

Word embedding representations can be also learned as a set of parameters in an end-to-end neural network model. For instance, Zamani et al. [39] trained a context-aware ranking model in which the embedding vectors of frequent n-grams are learned using click data. More recently, Dehghani et al. [9] trained neural ranking models with weak supervision data (i.e., a set of noisy training data automatically generated by an existing unsupervised model) that learn word representations in an end-to-end ranking scenario.

Word embedding vectors have been successfully employed in several NLP and IR tasks. Kusner et al. [16] proposed word mover's distance (WMD), a function for calculating semantic distance between two documents, which measures the minimum traveling distance from the embedded vectors of individual words in one document to the other one. Zhou et al. [47] introduced an embedding-based method for question retrieval in the context of community question answering. Vulić and Moens [37] proposed a model to learn bilingual word embedding vectors from document-aligned comparable corpora. Zheng and Callan [46] presented a supervised embedding-based technique to re-weight terms in the existing IR models, e.g., BM25. Based on the well-defined structure of language modeling framework in information retrieval, a number of methods have been introduced to employ word embedding vectors within this framework in order to improve the performance in IR tasks. For instance, Zamani and Croft [40] presented a set of embedding-based query language models using the query expansion and pseudo-relevance feedback techniques that benefit from the word embedding vectors. Query expansion using word embedding has been also studied in [11, 17, 35]. All of these approaches are based on word embeddings learned based on term proximity information. PhraseFinder [14] is an early work using term proximity information for query expansion. Mapping vocabulary terms to HAL space, a low-dimensional space compared to vocabulary size, has been used in [4] for query modeling.

As is widely known in the information retrieval literature [11, 38], there is a big difference between the unigram distribution of words on sub-topics of a collection and the unigram distribution estimated from the whole collection. Given this phenomenon, Diaz et al. [11] recently proposed to train word embedding vectors on the top retrieved documents for each query. However, this model, called local embedding, is not always practical in real-word applications, since the embedding vectors need to be trained during the query time. Furthermore, the objective function in local embedding is based on term proximity in pseudo-relevant documents.

In this paper, we propose two models for learning word embedding vectors, that are specifically designed for information retrieval needs. All the aforementioned tasks in this section can potentially benefit from the vectors learned by the proposed models.

3 RELEVANCE-BASED EMBEDDING

Typical word embedding algorithms, such as word2vec [24] and GloVe [28], learn high-dimensional real-valued embedding vectors based on the proximity of terms in a training corpus, i.e., co-occurrence of terms in the same context window. Although these approaches could be useful for learning the embedding vectors that can capture semantic and syntactic similarities between vocabulary terms and have shown to be useful in many NLP and IR tasks, there is a large gap between their learning objective (i.e., term proximity) and what is needed in many information retrieval tasks. For example, consider the query expansion task and assume that a user submitted the query "dangerous vehicles". One of the most similar terms to this query based on the typical word embedding algorithms (e.g., word2vec and GloVe) is "safe", and thus it would get a high weight in the expanded query model. The reason is that the words "dangerous" and "safe" often share similar contexts. However, expanding the query with the word "safe" could lead to poor retrieval performance, since it changes the meaning and the intent of the query.

This example together with many others have motivated us to revisit the objective used in the learning process of word embedding algorithms in order to obtain the word vectors that better match with the needs in IR tasks. The primary objective in many IR tasks is to model the notion of *relevance*. Several approaches, such as the relevance models proposed by Lavrenko and Croft [20], have been proposed to model relevance. Given the successes achieved by these models, we propose to learn word embedding vectors based on an objective that matters in information retrieval. The objective is to accurately predict the terms that are observed in a set of relevant documents to a particular information need.

In the following subsections, we first describe our neural network architecture, and then explain how to build a training set for learning relevance-based word embeddings. We further introduce two models, relevance likelihood maximization (RLM) and relevance posterior estimation (RPE), with different objectives using the described neural network.

3.1 Neural Network Architecture

We use a simple yet effective feed-forward neural network with a single linear hidden layer. The architecture of our neural network is shown in Figure 1. The input of the model is a sparse query vector $\vec{q}_s$ with the length of N, where N denotes the total number of vocabulary terms. This vector can be obtained by a projection function given the vectors corresponding to individual query terms. In this paper, we simply consider average as the projection function. Hence, $\vec{q}_s = \frac{1}{|q|} \sum_{w \in q} \vec{e}_w$, where $\vec{e}_w$ and $|q|$ denote the one-hot vector representation of term w and the query length, respectively. The hidden layer in this network maps the given query sparse vector to a query embedding vector $\vec{q}$, as follows:

$$\vec{q} = \vec{q}_s \times \mathcal{W}_Q \tag{1}$$

where $\mathcal{W}_Q \in \mathbb{R}^{N \times d}$ is a weight matrix for estimating query embedding vectors and d denotes the embedding dimensionality. The output layer of the network is a fully-connected layer given by:

$$\sigma(\vec{q} \times \mathcal{W}_w + b_w) \tag{2}$$

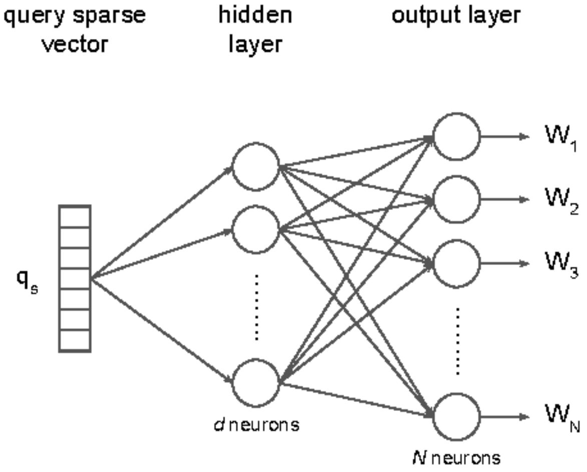


Figure 1: The relevance-based word embedding architecture. The objective is to learn d-dimensional distributed representation for words based on the notion of relevance, instead of term proximity. N denotes the total number of vocabulary terms.

where $\mathcal{W}_w \in \mathbb{R}^{d \times N}$ and $b_w \in \mathbb{R}^{1 \times N}$ are the weight and the bias matrices for estimating the probability of each term. σ is the activation function which is discussed in Sections 3.3 and 3.4.

To summarize, our network contains two sets of embedding parameters, $\mathcal{W}_Q$ and $\mathcal{W}_w$. The former aims to map the query into the "query embedding space", while the latter is used to estimate the weights of individual terms.

3.2 Modeling Relevance for Training

Relevance feedback has been shown to be highly effective in improving retrieval performance [7, 32]. In relevance feedback, a set of relevant documents to a given query is considered for estimating accurate query models. Since explicit relevance signals for a given query are not always available, pseudo-relevance feedback (PRF) assumes that the top retrieved documents in response to the given query are relevant to the query and uses these documents in order to estimate better query models. The effectiveness of PRF in various retrieval scenarios indicates that useful information can be captured from the top retrieved documents [19, 20, 44]. In this paper, we make use of this well-known assumption to train our model. It should be noted that there is a significant difference between PRF and the proposed models: In PRF, the feedback model is estimated from the top retrieved documents of the given query in an online setting. In other words, PRF retrieves the documents for the initial query and then estimates the feedback model using the top retrieved documents. In this paper, we propose to train the model in an offline setting. Moving from the online to the offline setting would lead to substantial improvements in efficiency, because an extra retrieval run is not needed in the offline setting. To learn a model in an offline setting, we consider a fixed-length dense vector for each vocabulary term and estimate these vectors based on the information extracted from the top retrieved documents for large numbers of training queries. Note that our models are unsupervised. However, if explicit relevance data is available, such as click data, without loss of generality, both the explicit or implicit relevant documents can be considered for training our models. We leave studying the vectors learned based on supervised signals for future work.

To formally describe our training data, let $T = \{(q_1, \mathcal{R}_1), (q_2, \mathcal{R}_2), \cdots, (q_m, \mathcal{R}_m)\}$ be a training set with m training queries. The i^{th} element of this set is a pair of query q_i and the corresponding pseudo-relevance feedback distribution. These distributions are estimated based on the top k retrieved documents (in our experiments, we set k to 10) for each query. The distributions can be estimated using any PRF model, such as those proposed in [20, 36, 42, 44]. In this paper, we only focus on the relevance model [20], a state-of-the-art PRF model, that estimates the relevance distribution as:

$$p(w|\mathcal{R}_i) \propto \sum_{d \in F_i} p(w|d) \prod_{w' \in q_i} p(w'|d) \qquad (3)$$

where F_i denotes a set of top retrieved documents for query q_i. Note that the probability of terms that do not appear in the top retrieved documents is equal to zero.

3.3 Relevance Likelihood Maximization Model

In this model, the goal is to learn the relevance distribution $\mathcal{R}$. Given a set of training data, we aim to find a set of parameters $\theta_{\mathcal{R}}$ in order to maximize the likelihood of generating relevance model probabilities for the whole training set. The likelihood function is defined as follows:

$$\prod_{i=1}^{m} \prod_{w \in V_i} \widehat{p}(w|q_i; \theta_{\mathcal{R}})^{p(w|\mathcal{R}_i)} \qquad (4)$$

where $\widehat{p}$ is the relevance distribution that can be obtained given the learning parameters $\theta_{\mathcal{R}}$ and $p(w|\mathcal{R}_i)$ denotes the relevance model distribution estimated for the i^{th} query in the training set (see Section 3.2 for more detail). V_i denotes a subset of vocabulary terms that appeared in the top ranked documents retrieved for the query q_i. The reason for iterating over the terms that appeared in this set instead of the whole vocabulary set V is that the probability $p(w|\mathcal{R}_i)$ is equal to zero for all terms $w \in V - V_i$.

In this method, we model the probability distribution $\widehat{p}$ using the softmax function (i.e., the function σ in Equation (2)) as follows:[1]

$$\widehat{p}(w|q; \theta_{\mathcal{R}}) = \frac{\exp(\vec{w}^T \vec{q})}{\sum_{w' \in V} \exp(\vec{w'}^T \vec{q})} \qquad (5)$$

where $\vec{w}$ denotes the learned embedding vector for term w and $\vec{q}$ is the query vector came from the output of the hidden layer in our network (see Section 3.1). According to the softmax modeling and the log-likelihood function, we have the following objective:

$$\arg\max_{\theta_{\mathcal{R}}} \sum_{i=1}^{m} \sum_{w \in V_i} p(w|\mathcal{R}_i) \left(\log \exp(\vec{w}^T \vec{q_i}) - \log \sum_{w' \in V} \exp(\vec{w'}^T \vec{q_i}) \right) \qquad (6)$$

Computing this objective function and its derivatives would be computationally expensive (due to the presence of the normalization factor $\sum_{w' \in V} \exp(\vec{w'}^T \vec{q})$ in the objective function). Since all the word embedding vectors as well as the query vector are changed during the optimization process, we cannot simply omit the normalization term as is done in [41] for estimating query embedding vectors based on pre-trained word embedding vectors. To make the computations more tractable, we consider a hierarchical approximation of the softmax function, which was introduced by Morin and Bengio [26] in the context of neural network language models and then successfully employed by Mikolov et al. [24] in the word2vec model.

The hierarchical softmax approximation uses a binary tree structure to represent the vocabulary terms, where each leaf corresponds to a unique word. There exists a unique path from the root to each leaf, and this path is used for estimating the probability of the word representing by the leaf. Therefore, the complexity of calculating softmax probabilities goes down from $O(|V|)$ to $O(\log(|V|))$ which is the height of the tree. This leads to a huge improvement in computational complexity. We refer the reader to [25, 26] for the details of calculating the hierarchical softmax approximation.

3.4 Relevance Posterior Estimation Model

As an alternative to maximum likelihood estimation, we can estimate the relevance posterior probability. In the context of pseudo-relevance feedback, Zhai and Laffery [44] assumed that the language model of the top retrieved documents is estimated based on a mixture model. In other words, it is assumed that there are two language models for the feedback set: the relevance language model[2] and a background noisy language model. They used an expectation-maximization algorithm to estimate the relevance language model. In this model, we make use of this assumption in order to cast the problem of estimating the relevance distribution $\mathcal{R}$ as a classification task: *Given a pair of word w and query q, does w come from the relevance distribution of the query q?* Instead of $p(w|\mathcal{R})$, this model estimates $p(R = 1|w, q; \theta_{\mathcal{R}})$ where R is a Boolean variable and $R = 1$ means that the given term-query pair (w, q) comes from the relevance distribution $\mathcal{R}$. $\theta_{\mathcal{R}}$ is a set of parameters that is going to be learned during the training phase.

Therefore, the problem is cast as a binary classification task that can be modeled by logistic regression (which means the function σ in Equation (2) is the sigmoid function):

$$\widehat{p}(R = 1|\vec{w}, \vec{q}; \theta_{\mathcal{R}}) = \frac{1}{1 + e^{(-\vec{w}^T \vec{q})}} \qquad (7)$$

where $\vec{w}$ is the relevance-based word embedding vector for term w. Similar to the previous model, $\vec{q}$ is the output of the hidden layer of the network, representing the query embedding vector.

In order to address this binary classification problem, we consider a cross-entropy loss function. In theory, for each training query, our model should learn to model relevance for the terms appearing in the corresponding pseudo-relevant set and non-relevance for all the other vocabulary terms, which could be impractical, due to the large number of vocabulary terms. Similar to [24], we propose to use the noise contrastive estimation (NCE) [12] which hypothesizes that we can achieve a good model by only differentiating the data from noise via a logistic regression model. The main concept in NCE is similar to those proposed in the divergence from randomness model [3] and the divergence minimization feedback model [44].

[1] For simplicity, we drop the bias term in these equations.

[2] The phrase "topical language model" was used in the original work [44]. We call it "relevance language model" to have consistent definitions in our both models.

Based on the NCE hypothesis, we define the following negative cross-entropy objective function for training our model:

$$\arg\max_{\theta_{\mathcal{R}}} \sum_{i=1}^{m} \Bigg[\sum_{j=1}^{\eta^+} \mathbb{E}_{w_j \sim p(w|\mathcal{R}_i)} \left[\log \widehat{p}(R=1|\vec{w_j}, \vec{q_i}; \theta_{\mathcal{R}})\right] + \sum_{j=1}^{\eta^-} \mathbb{E}_{w_j \sim p_n(w)} \left[\log \widehat{p}(R=0|\vec{w_j}, \vec{q_i}; \theta_{\mathcal{R}})\right] \Bigg] \tag{8}$$

where $p_n(w)$ denotes a noise distribution and $\eta = (\eta^+, \eta^-)$ is a pair of hyper-parameters to control the number of positive and negative instances per query, respectively. We can easily calculate $\widehat{p}(R = 0|\vec{w_j}, \vec{q_i}) = 1 - \widehat{p}(R = 1|\vec{w_j}, \vec{q_i})$. The noise distribution $p_n(w)$ can be estimated using a function of unigram distribution $U(w)$ in the whole training set. Similar to [24], we use $p_n(w) \propto U(w)^{3/4}$ which has been empirically shown to work effectively for negative sampling.

It is notable that although this model learns embedding vectors for both queries and words, it is not obvious how to calculate the probability of each term given a query; because Equation 7 only gives us a classification probability and we cannot simply use the Bayes rule here (since, not all probability components are known). This model can perform well when computing the similarity between two terms or two queries, but not a query and a term. However, we can use the model presented in [41] to estimate the query model using the word embedding vectors (not the ones learned for query vectors) and then calculate the similarity between a query and a term.

4 EXPERIMENTS

In this section, we first describe how we train the relevance-based word embedding models. We further extrinsically evaluate the learned embeddings using two IR tasks: query expansion and query classification. Note that the main aim here is to compare the proposed models with the existing word embedding algorithms, not with the state-of-the-art query expansion and query classification models.

4.1 Training

In order to train relevance-based word embeddings, we obtained millions of unique queries from the publicly available AOL query logs [27]. This dataset contains a sample of web search queries from real users submitted to the AOL search engine within a three-month period from March 1, 2006 to May 31, 2006. We only used query strings and no session and click information was obtained from this dataset. We filtered out the navigational queries containing URL substrings, i.e., "http", "www.", ".com", ".net", ".org", ".edu". All non-alphanumeric characters were removed from all queries. Applying all these constraints leads to over 6 millions unique queries as our training query set. To estimate the relevance model distributions in the training set, we considered top 10 retrieved documents in a target collection in response to each query using the Galago[3] implementation of the query likelihood retrieval model [29] with Dirichlet prior smoothing ($\mu = 1500$) [45].

We implemented and trained our models using TensorFlow[4]. The networks are trained based on the stochastic gradient descent optimizer using the back-propagation algorithm [33] to compute the gradients. All model hyper-parameters were tuned on the training set (the hyper-parameters with the smallest training loss value were selected). For each model, the learning rate and the batch size were selected from $[0.001, 0.01, 0.1, 1]$ and $[64, 128, 256]$, respectively. For RPE , we also tuned the number of positive and negative instances (i.e., η^+ and η^-). The value of η^+ was swept between $[20, 50, 100, 200]$ and the parameter η^- was selected from $[5\eta^+, 10\eta^+, 20\eta^+]$. As suggested in [40], in all the experiments (unless otherwise stated) the embedding dimensionality was set to 300, for all models including the baselines.

4.2 Evaluation via Query Expansion

In this subsection, we evaluate the embedding models in the context of query expansion for the ad-hoc retrieval task. In the following, we first describe the retrieval collections used in our experiments. We further explain our experimental setup as well as the evaluation metrics. We finally report and discuss the query expansion results.

4.2.1 Data. We use four standard test collections in our experiments. The first two collections (AP and Robust) consist of thousands of news articles and are considered as homogeneous collections. AP and Robust were previously used in TREC 1-3 Ad-Hoc Track and TREC 2004 Robust Track, respectively. The second two collections (GOV2 and ClueWeb) are large-scale web collections containing heterogeneous documents. GOV2 consists of the ".gov" domain web pages, crawled in 2004. ClueWeb (i.e., ClueWeb09-Category B) is a common web crawl collection that only contains English web pages. GOV2 and ClueWeb were previously used in TREC 2004-2006 Terabyte Track and TREC 2009-2012 Web Track, respectively. The statistics of these collections as well as the corresponding TREC topics are reported in Table 1. We only used the title of topics as queries.

4.2.2 Experimental Setup. We cleaned the ClueWeb collection by filtering out the spam documents. The spam filtering phase was done using the Waterloo spam scorer[5] [6] with the threshold of 60%. Stopwords were removed from all collections using the standard INQUERY stopword list and no stemming were performed.

For the purpose of query expansion, we consider the language modeling framework [29] and estimate a query language model based on a given set of word embedding vectors. The expanded query language model $p(w|\theta_q^*)$ is estimated as:

$$p(w|\theta_q^*) = \alpha p_{ML}(w|q) + (1 - \alpha)p(\vec{w}|\vec{q}) \tag{9}$$

where $p_{ML}(w|q)$ denotes maximum likelihood estimation of the original query and α is a free hyper-parameter that controls the weight of original query model in the expanded model. The probability $p(\vec{w}|\vec{q})$ is calculated based on the trained word embedding vectors. In our first model, this probability can be estimated using Equation (5); while in the second model, we should simply use the Bayes rule given Equation (7) to estimate this probability. However, since we do not have any information about the probability of each

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

[4] http://tensorflow.org/

[5] http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

Table 1: Collections statistics.

ID	collection	queries (title only)	#docs	avg 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	25m	648	26,917
ClueWeb	ClueWeb 09 - Category B	TREC 2009-2012 Web Track topics 1-200	50m	1506	18,771

Table 2: Evaluating relevance-based word embeddings in the context of query expansion. The superscripts 0/1/2/3/4 denote that the MAP improvements over MLE/word2vec-external/word2vec-target/GloVe-external/GloVe-target are statistically significant. The highest value in each row is marked in bold.

Collection	Metric	MLE	word2vec		GloVe		Rel.-based Embedding	
			external	target	external	target	RLM	RPE
AP	MAP	0.2197	0.2399	0.2420	0.2319	0.2389	**0.2580**^{01234}	0.2543^{01234}
	P@20	0.3503	0.3688	0.3738	0.3581	0.3631	**0.3886**^{01234}	0.3812^{034}
	NDCG@20	0.3924	0.4030	0.4181	0.4025	0.4098	**0.4242**^{01234}	0.4226^{01234}
Robust	MAP	0.2149	0.2218	0.2215	0.2209	0.2172	**0.2450**^{01234}	0.2372^{01234}
	P@20	0.3319	0.3357	0.3337	0.3345	0.3281	**0.3476**^{01234}	0.3409^{024}
	NDCG@20	0.3863	0.3918	0.3881	0.3918	0.3844	**0.3982**^{01234}	0.3955^{0}
GOV2	MAP	0.2702	0.2740	0.2723	0.2718	0.2709	**0.2867**^{01234}	0.2855^{01234}
	P@20	0.5132	0.5257	0.5172	0.5186	0.5128	**0.5367**^{01234}	0.5358^{01234}
	NDCG@20	0.4482	0.4571	0.4509	0.4539	0.4485	**0.4576**^{01234}	0.4557^{01234}
ClueWeb	MAP	0.1028	0.1033	0.1033	0.1029	0.1026	**0.1066**^{01234}	0.1031
	P@20	0.3025	0.3040	0.3053	0.3033	0.3048	**0.3073**	0.3030
	NDCG@20	0.2237	0.2235	0.2252	0.2244	0.2244	**0.2273**^{01}	0.2241

term given a query, we use the uniform distribution. For other word embedding models (i.e., word2vec and GloVe), we use the standard method described in [11]. For all the models, we ignore the terms whose embedding vectors are not available.

We retrieve the documents for the expanded query language model using the KL-divergence formula [18] with Dirichlet prior smoothing ($\mu = 1500$) [45]. All the retrieval experiments were carried out using the Galago toolkit [7].

In all the experiments, the parameters α (the linear interpolation coefficient) and m (the number of expansion terms) were set using 2-fold cross-validation over the queries in each collection. We selected the parameter α from $\{0.1, \ldots, 0.9\}$ and the parameter m from $\{10, 20, ..., 100\}$.

4.2.3 Evaluation Metrics. To evaluate the effectiveness of query expansion models, we report three standard evaluation metrics: mean average precision (MAP) of the top ranked 1000 documents, precision of the top 20 retrieved documents (P@20), and normalized discounted cumulative gain [13] calculated for the top 20 retrieved documents (nDCG@20). Statistically significant differences of MAP, P@20, and nDCG@20 values based on the two-tailed paired t-test are computed at a 95% confidence level (i.e., $p\_value < 0.05$).

4.2.4 Results and Discussion. To evaluate our models, we consider the following baselines: *(i)* the standard maximum likelihood estimation (MLE) of the query model without query expansion, *(ii)* two sets of embedding vectors (one trained on Google News as a large external corpus and one trained on the target retrieval collection) learned by the word2vec model[6] [24], and *(iii)* two sets of embedding vectors (one trained on Wikipedia 2004 plus Gigawords 5 as a large external corpus[7] and the other on the target retrieval collection) learned by the GloVe model [28].

Table 2 reports the results achieved by the proposed models and the baselines. According to this table, all the query expansion models outperform the MLE baseline in nearly all cases, which indicates the effectiveness of employing high-dimensional word representations for query expansion. Similar observations have been made in [11, 17, 40, 41]. According to the results, although word2vec performs slightly better than GloVe, no significant differences can be observed between their performances. According to Table 2, both relevance-based embedding models outperform all the baselines in all the collections, which shows the importance of taking relevance into account for training embedding vectors. These improvements are often statistically significant compared to all the baselines. The relevance likelihood maximization model (RLM) performs better than the relevance posterior estimation model (RPE) in all cases and the reason is related to their objective function. RLM learns the relevance distribution for all terms, while RPE learns the classification probability of being relevance for vocabulary terms (see Equations (5) and (7)).

[6] We use the CBOW implementation of the word2vec model. The skip-gram model also performs similarly.

[7] Available at http://nlp.stanford.edu/projects/glove/.

Table 3: Top 10 expansion terms obtained by the word2vec and the relevance-based word embedding models for two sample queries "indian american museum" and "tibet protesters".

query: "indian american museum"				query: "tibet protesters"			
word2vec		Rel.-based Embedding		word2vec		Rel.-based Embedding	
external	target	RLM	RPE	external	target	RLM	RPE
history	powwows	chumash	heye	demonstrators	tibetan	tibetan	tibetan
art	smithsonian	heye	collection	protestors	lhasa	lama	tibetans
culture	afro	artifacts	chumash	tibetan	demonstrators	tibetans	lama
british	mesoamerica	smithsonian	smithsonian	protests	tibetans	lhasa	independence
heritage	smithsonians	collection	york	tibetans	marchers	dalai	lhasa
society	native	washington	new	protest	lhasas	independence	dalai
states	heye	institution	apa	activists	jokhang	protest	open
contemporary	hopi	york	native	protesting	demonstrations	open	protest
part	mayas	native	americans	lhasa	dissidents	zone	zone
united	cimam	apa	history	demonstrations	barkhor	followers	jokhang

To get a sense of what is learned by each of the embedding models[8], in Table 3 we report the top 10 expansion terms for two sample queries from the Robust collection. According to this table, the terms added to the query by the word2vec model are syntactically or semantically related to individual query terms, which is expected. For the query "indian american museum" as an example, the terms "history", "art", and "culture" are related to the query term "museum", while the terms "united" and "states" are related to the query term "american". In contrast, looking at the expansion terms obtained by the relevance-based word embeddings, we can see that some relevant terms to the whole query were selected. For instance, "chumash" (a group of native americans)[9], "heye" (the national museum of the American Indian in New York), "smithsonian" (the national museum of the American Indian in Washington DC), and "apa" (the American Psychological Association that actively promotes American Indian museums). A similar observation can be made for the other sample query (i.e., "tibet protesters"). For example, the word "independence" is related to the whole query that was only selected by the relevance-based word embedding models, while the terms "protestors", "protests", "protest", and "protesting" that are syntactically similar to the query term "protesters" were considered by the word2vec model. We believe that these differences are due to the learning objective of the models. Interestingly, the expansion terms added to each query by the two relevance-based models look very similar, but according to Table 2, their performances are quite different. The reason is related to the weights given to each term by the two models. The weights given to the expansion terms by RPE are very close to each other because its objective is to just classify each term and all of these terms are classified with a high probability as "relevant".

In the next set of experiments, we consider the methods that use the top retrieved documents for query expansion: the relevance model (RM3) [1, 20] as a state-of-the-art pseudo-relevance feedback model, and the local embedding approach recently proposed by Diaz et al. [11] with the general idea of training word embedding models on the top ranked documents retrieved in response to a given query. Similar to [11], we use the word2vec model to train word embedding vectors on top 1000 documents. The results are reported in Table 4. In this table, ERM refers to the embedding-based relevance model recently proposed by Zamani and Croft [40] in order to make use of semantic similarities estimated based on the word embedding vectors in a pseudo-relevance feedback scenario. According to Table 4, the ERM model that uses the relevance-based word embedding (RLM[10]) outperforms all the other methods. These improvements are statistically significant in most cases. By comparing the results obtained by local embedding and those reported in Table 2, it can be observed that there are no substantial differences between the results for local embedding and word2vec. This is similar to what is reported by Diaz et al. [11] when the embedding vectors are trained on the top documents in the target collection, similar to our setting. Note that the relevance-based model was also trained on the target collection.

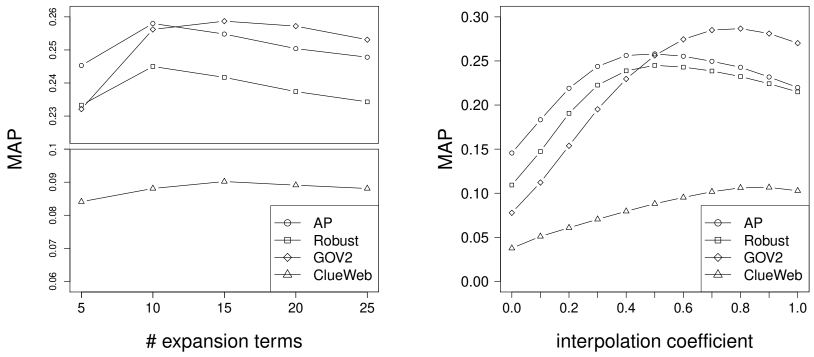
Table 4: Evaluating relevance-based word embedding in pseudo-relevance feedback scenario. The superscripts 1/2/3 denote that the MAP improvements over RM3/Local Embedding/ERM with Local Embedding are statistically significant. The highest value in each row is marked in bold.

Collection	Metric	RM3	Local Emb.	ERM	
				Local	RLM
AP	MAP	0.2927	0.2412	0.3047	**0.3119**^{12}
	P@20	0.4034	0.3742	0.4105	**0.4233**^{12}
	NDCG@20	0.4368	0.4173	0.4411	**0.4495**^{123}
Robust	MAP	0.2593	0.2235	0.2643	**0.2761**^{123}
	P@20	0.3486	0.3366	0.3498	**0.3605**^{123}
	NDCG@20	0.4011	0.3868	0.4080	**0.4173**^{123}
GOV2	MAP	0.2863	0.2748	0.2924	**0.2986**^{123}
	P@20	0.5318	0.5271	0.5379	**0.5417**^{12}
	NDCG@20	0.4503	0.4576	0.4584	**0.4603**^{123}
ClueWeb	MAP	0.1079	0.1041	0.1094	**0.1121**^{12}
	P@20	0.3111	0.3062	0.3145	**0.3168**
	NDCG@20	0.2309	0.2261	0.2328	**0.2360**^{2}

[8] For the sake of space, we only report the expanded terms estimated by the word2vec model and the proposed models.

[9] see https://en.wikipedia.org/wiki/Chumash_people

[10] For the sake of space, we only consider RLM which shows better performance compared to RPE in query expansion.



(a) # expansion terms

(b) interpolation coefficient

Figure 2: Sensitivity of RLM to the number of expansion terms and the interpolation coefficient (α), in terms of MAP.

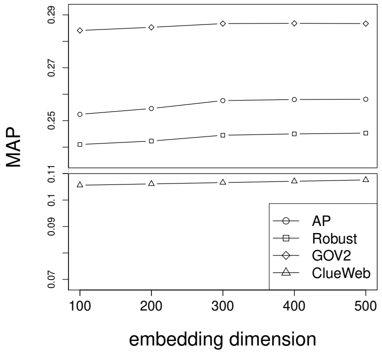


Figure 3: Sensitivity of RLM to the dimension of embedding vectors, in terms of MAP.

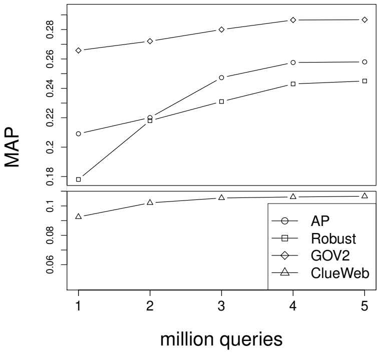


Figure 4: The Performance of RLM with respect to different amount of training data (training queries), in terms of MAP.

An interesting observation from Tables 2 and 4 is that the RLM performance (without using pseudo-relevant documents) in Robust and GOV2 is very close to the RM3 performance, and is slightly better in the GOV2 collection. Note that RM3 needs two retrieval runs[11] and uses top retrieved documents, while RLM only needs one retrieval run. This is an important issue in many real-world applications, since the efficiency constraints do not always allow them to have two retrieval runs per query.

Parameter Sensitivity. In the next set of experiments, we study the sensitivity of RLM as the best performing word embedding model in Table 2 to the expansion parameters. Figure 2a plots the sensitivity of RLM to the number of expansion terms where the parameter α is set to 0.5. According to this figure, in both newswire collections, the method shows its best performance when the queries are expanded with only 10 words. In the GOV2 collection, 15 words are needed for the method to show its best performance.

Figure 2b plots the sensitivity of the methods to the interpolation coefficient α (see Equation 9) where the number of expansion terms is set to 10. According to the curves correspond to AP and Robust, the original query language model needs to be interpolated with the model estimated using relevance-based word embeddings with equal weights (i.e., $\alpha = 0.5$). This shows the quality of the estimated distribution via the learned embedding vectors. In the GOV2 collection, a higher weight should be given to the original query model, which indicates that the original query plays a key role in achieving good retrieval performance in this collection.

We also study the performance of RLM as the best performing word embedding model for query expansion with respect to the embedding dimensionality. The results are shown in Figure 3, where the query expansion performance generally improves as we increase the embedding dimensionality. The performances become stable when the dimension is larger than 300. This experiment suggests that 400 dimensions would be enough for the relevance-based embedding model.

Due to the large number of parameters in the neural networks, they can require large amounts of training data to achieve good performance. In the next set of experiments, we study how much training data is needed for training our best model. The results are plotted in Figure 4. According to this figure, by increasing the number of training queries from one million to four million queries, the performance significantly increases, and becomes more stable after four million queries.

4.3 Evaluation via Query Classification

In this subsection, we evaluate the proposed embedding models in the context of query classification. In this task, each query is

[11] Diaz [10] showed that for precision-oriented tasks, the second retrieval run can be restricted to the initial rank list for improving the efficiency of PRF models. However, for recall-oriented metrics, e.g., MAP, the second retrieval helps a lot.

Table 5: Evaluating embedding algorithms via query classification. The superscripts 1/2 denote that the improvements over word2vec/GloVe are significant. The highest value in each column is marked in bold.

Method	Precision	F1-measure
word2vec	0.3712	0.4008
GloVe	0.3643	0.3912
Rel.-based Embedding - RLM	0.3943^{12}	0.4267^{12}
Rel.-based Embedding - RPE	**0.3961**^{12}	**0.4294**^{12}

assigned to a number of labels (categories) which are pre-defined and a few training queries are available for each label. This is a supervised multi-label classification task with little training data.

4.3.1 Data. We consider the dataset that was introduced in KDD Cup 2005 [22] for the internet user search query categorization task and was previously used in [41] for evaluating query embedding vectors. This dataset contains 800 web queries submitted by real users randomly collected from the MSN search logs. The queries do not contain "junk" text or non-English terms. The queries were labelled by three human editors. 67 categories were pre-defined and up to 5 labels were selected for each query by each editor.

4.3.2 Experimental Setup. In our experiments, we performed 5-fold cross-validation over the queries and the reported results are the average of those obtained over the test folds. In all experiments, the spelling errors in queries were corrected in a pre-processing phase, the stopwords were removed from queries (using the INQUERY stopword list), and no stemming was performed.

To classify each query, we consider a very simple kNN-based approach proposed in [41]. We first compute the probability of each category/label given each query q and then select the top t categories with the highest probabilities. The probability $p(C_i|q)$ is computed as follows:

$$p(C_i|q) = \frac{\delta(\vec{C_i}, \vec{q})}{\sum_j \delta(\vec{C_j}, \vec{q})} \propto \delta(\vec{C_i}, \vec{q}) \qquad (10)$$

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 training set. We ignore the query terms whose embedding vectors are not available. The number of labels assigned to each query was tuned on the training set from $\{1, 2, 3, 4, 5\}$. In the query classification experiments, we trained relevance-based word embedding using Robust as the collection.

4.3.3 Evaluation Metrics. We consider two evaluation metrics that were also used in KDD Cup 2005 [22]: 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. These metrics are computed in the same way as what is described in [22] for evaluating the KDD Cup 2005 submitted runs. Statistically significant differences are determined using the two-tailed paired t-test computed at a 95% confidence level ($p - value < 0.05$).

4.3.4 Results and Discussion. We compare our models against the word2vec and GloVe methods trained on the external collections that are described in the query expansion experiments. The results are reported in Table 5, where the relevance-based embedding models significantly outperform the baselines in terms of both metrics. An interesting observation here is that contrary to the query expansion experiments, RPE performs better than RLM in query classification. The reason is that in query expansion the weight of each term is considered in order to generate the expanded query language model. Therefore, in addition to the order of terms, their weights should be also effective for improving the retrieval performance with query expansion. In query classification, we only assign a few categories to each query, and thus as long as the order of categories is correct, the similarity values between the queries and the categories do not matter.

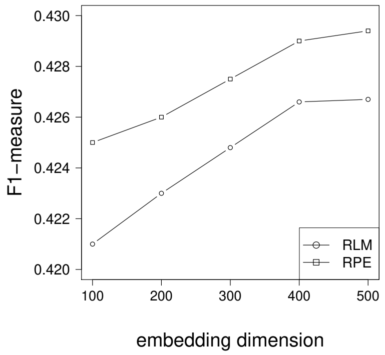


Figure 5: Sensitivity of the relevance-based embedding models to the embedding dimensionality, in terms of F1-measure.

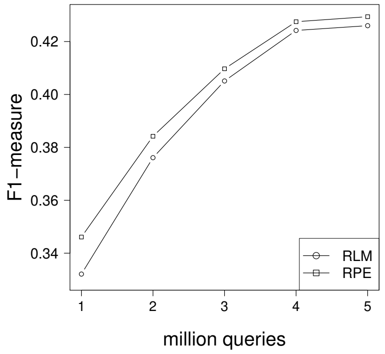


Figure 6: The Performance of relevance-based embedding models with respect to different amount of training data (training queries), in terms of F1-measure.

In the next set of experiments, we study the performance of our relevance-based word embedding models with respect to the embedding dimensionality. The results are plotted in Figure 5. According to this figure, the performance is generally improved by increasing the embedding dimensionality, and becomes stable when the dimension is greater than 400. This is similar to our observation in the query expansion experiments. We also study the amount of data needed for training our models in Figure 6. According to this figure, at least 4 million queries are needed in order to learn accurate relevance-based word embeddings. It can be seen from Figure 6 that RLM needs more training data compared to RPE in order to perform well, because by increasing the amount of training data the learning curves of these two models get closer.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we revisited the underlying assumption in typical word embedding models, such as word2vec and GloVe. Instead of learning embedding vectors based on term proximity, we proposed learning embeddings based on the notion of relevance, which is the primary objective in many IR tasks. We developed two neural network-based models for learning relevance-based word embeddings. The first model, the relevance likelihood maximization model, aims to estimate the probability of each word in a relevance distribution for each query, while the second one, the relevance posterior estimation model, classifies each term as belonging to relevant or non-relevant class for each query. We evaluated our models using two sets of extrinsic evaluation: query expansion and query classification. The query expansion experiments using four standard TREC collections, two newswire and two large-scale web collections, suggested that the relevance-based word embedding models outperform state-of-the-art word embedding algorithms. We showed that the expansion terms chosen by our models are related to the whole query, while those chosen by typical word embedding models are related to individual query terms. The query classification experiments also validated these findings and investigated the effectiveness of our models.

In the future, we intend to evaluate the learned embedding models in other IR tasks, such as query reformulation, query intent prediction, etc. We can also achieve more accurate relevance-based embedding vectors by considering the clicked documents for training query, instead of or in addition to the top retrieved documents.

Acknowledgements. The authors thank Daniel Cohen, Mostafa Dehghani, and Qingyao Ai for their invaluable comments. This work was supported in part by the Center for Intelligent Information Retrieval. 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.

REFERENCES

[1] Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *TREC '04*.

[2] Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. 2016. Analysis of the Paragraph Vector Model for Information Retrieval. In *ICTIR '16*. 133–142.

[3] Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Trans. Inf. Syst.* 20, 4 (2002), 357–389.

[4] P. D. Bruza and D. Song. 2002. Inferring Query Models by Computing Information Flow. In *CIKM '02*. 260–269.

[5] Stephane Clinchant and Florent Perronnin. 2013. Aggregating Continuous Word Embeddings for Information Retrieval. In *CVSC@ACL '13*. 100–109.

[6] Gordon V. Cormack, Mark D. Smucker, and Charles L. Clarke. 2011. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Inf. Retr.* 14, 5 (2011), 441–465.

[7] Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice* (1st ed.). Addison-Wesley Publishing Company.

[8] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. 41, 6 (1990), 391–407.

[9] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR '17*.

[10] Fernando Diaz. 2015. Condensed List Relevance Models. In *ICTIR '15*. 313–316.

[11] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *ACL '16*.

[12] Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *J. Mach. Learn. Res.* 13, 1 (2012), 307–361.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.

[14] Yufeng Jing and W. Bruce Croft. 1994. An Association Thesaurus for Information Retrieval. In *RIAO '94*. 146–160.

[15] Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *CIKM '15*. 1411–1420.

[16] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings to Document Distances. In *ICML '15*. 957–966.

[17] Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query Expansion Using Word Embeddings. In *CIKM '16*. 1929–1932.

[18] John Lafferty and Chengxiang Zhai. 2001. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *SIGIR '01*. 111–119.

[19] Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual Relevance Models. In *SIGIR '02*. 175–182.

[20] Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *SIGIR '01*. 120–127.

[21] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *NIPS '14*. 2177–2185.

[22] Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. 2005. KDD CUP-2005 Report: Facing a Great Challenge. *SIGKDD Explor. Newsl.* 7, 2 (2005), 91–99.

[23] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-yi Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *NAACL '15*. 912–921.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*. 3111–3119.

[25] Andriy Mnih and Geoffrey E Hinton. 2009. A Scalable Hierarchical Distributed Language Model. In *NIPS '09*. 1081–1088.

[26] Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *AISTATS '05*. 246–252.

[27] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A Picture of Search. In *InfoScale '06*.

[28] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP '14*. 1532–1543.

[29] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR '98*. 275–281.

[30] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Hamed Zamani. 2017. Word Embedding Causes Topic Shifting; Exploit Global Context!. In *SIGIR '17*.

[31] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Guido Zuccon. 2016. Generalizing Translation Models in the Probabilistic Relevance Framework. In *CIKM '16*. 711–720.

[32] J. J. Rocchio. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. 313–323.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323 (Oct. 1986), 533–536.

[34] T. Saracevic. 2016. *The Notion of Relevance in Information Science: Everybody knows what relevance is. But, what is it really?* Morgan & Claypool Publishers.

[35] Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. 2014. Learning Concept Embeddings for Query Expansion by Quantum Entropy Minimization. In *AAAI '14*. 1586–1592.

[36] Tao Tao and ChengXiang Zhai. 2006. Regularized Estimation of Mixture Models for Robust Pseudo-relevance Feedback. In *SIGIR '06*. 162–169.

[37] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and Cross-Lingual Information Retrieval Models Based on (Bilingual) Word Embeddings. In *SIGIR '15*. 363–372.

[38] Jinxi Xu and W. Bruce Croft. 1996. Query Expansion Using Local and Global Document Analysis. In *SIGIR '96*. 4–11.

[39] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *WWW '17*. 1531–1540.

[40] Hamed Zamani and W. Bruce Croft. 2016. Embedding-based Query Language Models. In *ICTIR '16*. 147–156.

[41] Hamed Zamani and W. Bruce Croft. 2016. Estimating Embedding Vectors for Queries. In *ICTIR '16*. 123–132.

[42] Hamed Zamani, Javid Dadashkarimi, Azadeh Shakery, and W. Bruce Croft. 2016. Pseudo-Relevance Feedback Based on Matrix Factorization. In *CIKM '16*. 1483–1492.

[43] ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *SIGIR '03*. 10–17.

[44] Chengxiang Zhai and John Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM '01*. 403–410.

[45] Chengxiang Zhai and John Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.* 22, 2 (2004), 179–214.

[46] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *SIGIR '15*. 575–584.

[47] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning Continuous Word Embedding with Metadata for Question Retrieval in Community Question Answering. In *ACL '15*. 250–259.