Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search

Mihajlo Grbovic[†], Nemanja Djuric[†], Vladan Radosavljevic[†],
Fabrizio Silvestri[‡], Narayan Bhamidipati[†]
{mihajlo, nemanja, vladan, silvestr, narayanb}@yahoo-inc.com
Yahoo Labs
[†]701 First Ave, Sunnyvale, CA, USA
[‡]125 Shaftesbury Ave, London, England

ABSTRACT

Search engines represent one of the most popular web services, visited by more than 85% of internet users on a daily basis. Advertisers are interested in making use of this vast business potential, as very clear intent signal communicated through an issued query allows effective targeting of users. This idea is embodied in a sponsored search model, where each advertiser maintains a list of keywords they deem indicative of increased user response rate with regards to their business. According to this targeting model, when a query is issued all advertisers with a matching keyword are entered into an auction according to the amount they bid for the query and the winner gets to show their ad. One of the main challenges is the fact that a query may not match many keywords, resulting in lower auction value, lower ad quality, and lost revenue for advertisers and publishers. Possible solution is to expand a query into a set of related queries and use them to increase the number of matched ads, called query rewriting. To this end, we propose rewriting method based on a novel query embedding algorithm, which jointly models query content as well as its context within a search session. As a result, semantically similar queries are mapped into vectors close in the embedding space, which allows expansion of a query via simple K-nearest neighbor search. The method was trained on more than 12 billion sessions, one of the largest corpus reported thus far, and evaluated on both public TREC data set and an in-house sponsored search data set. The results show that the proposed approach significantly outperformed existing state-of-the-art, strongly indicating its benefits and monetization potential.

Categories and Subject Descriptors

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

SIGIR '15, August 09 - 13, 2015, Santiago, Chile

http://dx.doi.org/10.1145/2766462.2767709 ...$15.00.

General Terms

Information Retrieval; Query Rewriting; Algorithms.

Keywords

Query rewriting; word embeddings.

1. INTRODUCTION

In recent years targeted advertising has become one of the largest and most lucrative advertising channels. Despite the fact that traditional offline advertising still accounts for the majority of advertising expenditures [34], future potential of this burgeoning field is clearly exemplified by reported ad revenue of over 20 billion dollars in the first half of 2013 in the US alone, combined with a remarkable growth of around 20% on a yearly basis [21]. The size and importance of the online advertising, as well as the interesting open questions that the scale and variety of the targeting tasks bring, has drawn attention of many researchers from both industry and academia, resulting in a number of novel methods and improvements to the flourishing field [1, 18, 32].

Due to a large diversity of the internet medium, targeted advertising has evolved to encompass many different outlets for the advertisers interested in reaching their target audience. These include behavioral targeting [12] (where users are targeted based on their general browsing behavior), e-mail retargeting [18] (targeting users based on their e-mail interaction patterns), site re-targeting (targeting users based on their historical search queries), to name a few. In this work we consider a task of sponsored search advertising [17, 22], a very popular advertising model that targets users with the most relevant ads by considering an immediate query issued by a user. It has become one of the prevalent means of advertising due to a fact that the current query carries a strong signal about an immediate intent of the user, resulting in a highly effective targeting model [15].

In the sponsored search model each advertiser maintains a list of keywords that they deem relevant to their product (e.g., Nike may maintain a keyword list containing terms such as "running shoes" and "athletics"). In addition to a list of keywords, each advertiser also specifies a monetary bid amount they are willing to pay if their ad is shown and clicked by the user. Then, when a user issues a query in a search engine, the query is compared against each advertisers' keyword list, and all advertisers with the matching keyword are entered into an auction. Finally, according to advertiser's bid amount and the estimated quality and rel-

evance of an ad that they wish to show, one advertiser and their corresponding ad are chosen for user targeting. Value of the auction increases when number of matched advertisers and their bids are high, which directly results in a better ad quality and higher revenue for both advertisers and publishers (i.e., websites that host the ads).

Due to a large number of queries that the users could possibly issue, a common occurrence is that an exact query match cannot be found, as the advertisers usually cannot cover all relevant queries related to their product or service. However, even when the exact match does not exist, most often there does exist a non-matching keyword that is still highly related to the query. For example, query "purina one" and bidded keyword "dog food" are strongly related yet a string match would fail to make the connection, which directly translates into lost revenue. To mitigate this problem, query rewriting is used to expand the original query by providing K related ones for which ads are available, and which can be used instead to qualify advertisers for an auction [9, 23, 40]. In the above example, a query rewriting method can be used to rewrite "purina one" into related queries such as "dog food", "cat food", "purina pro plan", and others, thus increasing likelihood of retrieving more high-quality, relevant ads, likely to be clicked by a user that issued the query.

In this paper we address this critical step in sponsored search, and propose a novel query rewriting algorithm motivated by recent advances in distributed neural language models [14, 30, 31]. We explore and expand upon these approaches for the task of query rewriting, resulting in significant performance improvements over the current state-of-the-art methods. Key contributions are summarized below:

- We describe an application of distributed language models to query rewriting, where we propose to use three novel methods for learning distributed, low-dimensional query representations that compactly capture their semantic meaning;
- We propose and discuss how to add ad clicks and search result clicks to query context, which allows specialization of representations for various tasks of critical interest to search engine companies (e.g., query suggestion, query-to-ad matching, query categorization), indicating even wider applicability of the method in the advertising domain;
- We trained and evaluated the models using more than 12 billion search sessions, resulting in rewriting results of high quality. Empirical results show that the proposed approach significantly outperform the existing state-of-the-art methods.

2. RELATED WORK

In this section we describe related work in the domains of query rewriting and neural language modeling that motivated the approach proposed in this work.

2.1 Query rewriting for sponsored search

Owing to the importance of query rewriting for the success of query-ad auctions, various algorithms have been proposed in the literature to address this critical step. These include graph-based methods such as Query Flow Graph (QFG) [6, 7] that learn from users' browsing behavior, as well as methods that exploit syntactical relationships between queries [11, 23]. However, disadvantage of existing solutions is that they mostly do not take into account complex semantic relationships between queries, which may lead to suboptimal rewrites. For example, the rewrites typically result in the original queries with added or removed terms, such as "purina one" being rewritten as "purina" or "purina one pro". Thus, they are often obvious and have already been considered by the advertisers that targeted the original query, and do not add a significant value to the auction.

We note that query rewriting task is related to the problem of query suggestion, albeit the goals of the two techniques are quite different. In the case of query suggestion the objective is to provide users with queries that are important for meeting users' information needs [2, 7, 39], while in query rewriting the task is to expand query to increase both the number and quality of retrieved ads. We refer an interested reader to [36] for an overview of query log mining approaches as applied to query suggestion task.

2.2 Neural language models

In a number of natural language processing (NLP) applications, including information retrieval, part-of-speech tagging, chunking, and many others, the specific objective can be generalized to the task of assigning a probability value to a sequence of words. To this end, language models have been developed, defining a mathematical model to capture statistical properties of words and the dependancies among them [3, 27]. Traditionally, language models represent each word as a feature vector using one-hot representation, where a word vector has the same length as the size of a vocabulary, and a vector element that corresponds to the observed word is equal to 1, and 0 otherwise. However, this approach often exhibits significant limitations in practical tasks, suffering from high dimensionality of the problem and severe data sparsity, resulting in suboptimal performance.

Neural language models have been proposed to address these issues, inducing low-dimensional, distributed embeddings of words by means of neural networks [4, 13, 38]. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. Historically, inefficient training of the neural network-based models has been an obstacle to their wider applicability, being proportional to the size of the vocabulary which may grow to several millions in practical tasks. However, this issue has been successfully addressed with recent advances in the field, particularly with the development of highly scalable continuous bag-of-words (CBOW) and skip-gram (SG) language models [30, 31] for learning word representations. These powerful, efficient models have shown very promising results in capturing both syntactic and semantic relationships between words in large-scale text corpora, obtaining state-of-the-art results on a plethora of NLP tasks. More recently, the concept of distributed representations has been extended beyond word representations to sentences and paragraphs [14, 28], relational entities [8, 37], general text-based attributes [26], descriptive text of images [25], nodes in graph structure [33], and other applications going beyond NLP domain for which they were originally proposed.

3. PROPOSED APPROACH

To address the shortcomings of the existing state-of-the-art methods for query rewriting, we propose to take a rad-

ically new approach to this task, motivated by the recent success of distributed language models in NLP applications [31, 38]. In the context of NLP, distributed models are able to learn word representations in a low-dimensional continuous vector space using a surrounding context of the word in a sentence, where in the resulting embedding space semantically similar words are close to each other [31]. Our objective is to take advantage of this property for the task of query rewriting, and to learn query representations in a low-dimensional space where semantically similar queries would be close. As a result, and in contrast to rewriting methods commonly used in practice, related queries could have a high similarity score even if they do not have any shared terms. Clearly, such approach would allow us to reduce complex task of query rewriting to a trivial K-nearest-neighbor search in the new embedding space.

However, application of distributed language models to the task of query rewriting is not an easy endeavour. Finding distributed query representation, as opposed to finding word representations, brings very unique challenges quite different from those found in everyday NLP problems. First, there are significant differences between language used in everyday language and web searches [10, 35]. For example, web search users often use summarization when searching for content (e.g., typing "vacations Spain" instead of "vacations in Spain"), thus omitting frequent words. Further, some n-grams that rarely appear in everyday language often appear together in web search [24], and queries are characterized by more spelling mistakes. Moreover, contrary to everyday language where linguistic rules and notions of words and sentences are clearly defined, search queries are composed of terms where there is no existing notion of "sentence of queries" or the surrounding context equivalent to natural language domain.

In this paper we address these issues, and propose three query rewriting methods that bring the state-of-the-art distributed language models closer to the setting of sponsored search: 1) context2vec, where we exploit the fact that user query is recorded with a timestamp, from which we create "query sentences" and apply state-of-the-art language model [31]; 2) content2vec, where we propose to learn distributed query vector representations from its content without considering session information, which is equivalent to paragraph2vec method from [28]; and 3) context-content2vec, where we propose to use a novel two-level architecture [14] that jointly models content of queries (i.e., word tokens that form the query) along with the query context (defined as other temporally close queries within a search session). Empirical results suggest that the resulting rewrites are highly related to the original queries, while being more diverse than rewrites produced by the current state-of-the-art methods. In addition, using a large data base of query-bids collected for thousands of advertisers, we show that the rewrites produced by the proposed method match the highest percentage of bidded queries, providing a strong positive impact on the overall revenue of both publishers and advertisers.

In addition to search query logs, search engines systematically log a number of other valuable signals that can help better explain and model the user intent. For example, ad clicks and clicks on search results are also recorded, and may provide a supplementary intent signal that can be used to improve query representations. To help exploit these two valuable sources of information, we propose to incorporate both ad clicks and search result clicks into user query sessions, and show how the additional signals are seamlessly handled by the proposed models. The benefits of including the ad and search clicks for query rewriting task are clearly confirmed by the empirical results, showing increased relevance of query rewrites and higher coverage of advertisers' bid terms. Moreover, in addition to improving query representations, in this way we also learn low-dimensional representations of ad clicks and search clicks in the same embedding space, which opens doors for using the proposed methods on a number of important online tasks, such as query-to-ad matching, query suggestion, and query categorization, to name a few.

4. METHODOLOGY

In this section we describe the proposed methodology for the task of query rewriting in sponsored search. As discussed earlier, the goal is to find queries related to the issued one, which would allow us to retrieve relevant ads that were not matched by the original query. To solve this problem, we propose to learn representation of queries in low-dimensional space from historical search logs using neural language models. Query rewriting can then be performed by finding K nearest neighbors of the issued query in the learned embedding space.

More specifically, given a set $\mathcal{S}$ of S search sessions obtained from online users, where each session $s = (q_1, \dots, q_{M_s}) \in \mathcal{S}$ is defined as an uninterupted sequence of M_s queries, and each query $q_m = (w_{m1}, w_{m2}, \dots w_{mT_m})$ consists of T_m words, our objective is to find D-dimensional real-valued representation $\mathbf{v}_{q_m} \in \mathcal{R}^D$ of each query q_m such that semantically similar queries lie nearby in the new space.

We propose three approaches for learning query representations that address specifics of the web search environment. We first propose context2vec and content2vec methods that consider query context and query content, respectively, motivated by previous work on learning word representations from news articles [30, 31]. We then detail context-content2vec, a two-level architecture for joint modeling of both query context and query content, which results in better, more useful query representations.

4.1 Model 1: context2vec

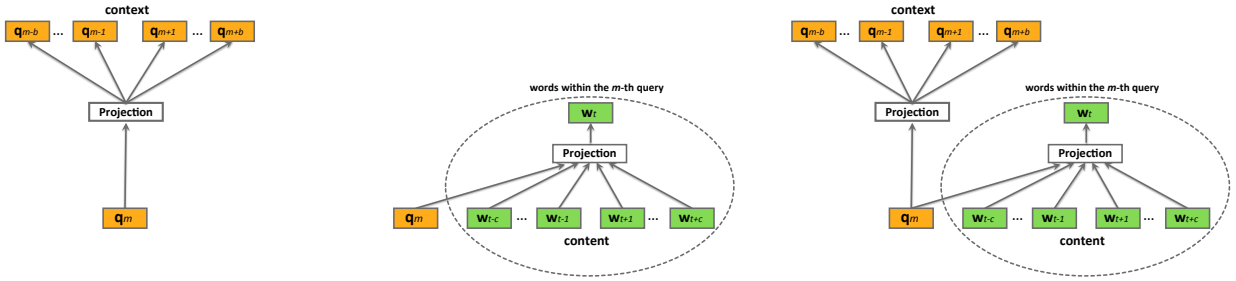
The context2vec model involves learning low-dimensional representations of queries from search logs by using a notion of a search session as a "sentence" and queries within the session as "words", borrowing the terminology from NLP domain (see Figure 1a). More specifically, context2vec learns query representations using the skip-gram model [31] by maximizing the objective function over the entire set $\mathcal{S}$ of search sessions, defined as

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{q_m \in s} \sum_{-b \le i \le b, i \neq 0} \log \mathbb{P}(q_{m+i}|q_m). \tag{4.1}$$

Probability $\mathbb{P}(q_{m+i}|q_m)$ of observing a neighboring query q_{m+i} given the current query q_m is defined using soft-max,

$$\mathbb{P}(q_{m+i}|q_m) = \frac{\exp(\mathbf{v}_{q_m}^\top \mathbf{v}'_{q_{m+i}})}{\sum_{q=1}^{V} \exp(\mathbf{v}_{q_m}^\top \mathbf{v}'_q)}, \tag{4.2}$$

where $\mathbf{v}_q$ and $\mathbf{v}'_q$ are the input and output vector representations of query q, b is defined as length of the context for query



(a) context2vec (b) content2vec (c) context-content2vec

Figure 1: Query embedding models

sequences, and V is the number of unique queries in the vocabulary. From equations (4.1) and (4.2) we can see that context2vec models temporal context of query sequences, where queries with similar contexts (i.e., with similar neighboring queries) will have similar vector representations in the projected semantic space.

4.2 Model 2: content2vec

We propose content2vec method to simultaneously learn vector representations of queries and the words contained within them, motivated by the paragraph2vec algorithm [28]. The content2vec architecture is illustrated in Figure 1b. Training data set was derived from user search logs by disregarding the query timestamps, and consisted of queries q_m and their containing words $q_m = (w_{m1}, w_{m2}, \dots w_{mT_m})$. During training, query vectors are learned so they predict the words in their content, while word vectors are learned so they predict their context words within the query. More specifically, objective of content2vec is to maximize the log-likelihood over the set $\mathcal{S}$ of all query sessions,

$$\mathcal{L} = \sum_{s\in\mathcal{S}} \Big(\sum_{q_m\in s} \log \mathbb{P}(q_m | w_{m1} : w_{mT_m}) + \sum_{w_{mt}\in q_m} \log \mathbb{P}(w_{mt} | w_{m,t-c} : w_{m,t+c}, q_m) \Big), \tag{4.3}$$

where c is length of the context for words within the query. The probability $\mathbb{P}(w_{mt} | w_{m,t-c} : w_{m,t+c}, q_m)$ is defined using a soft-max function,

$$\mathbb{P}(w_{mt} | w_{m,t-c} : w_{m,t+c}, q_m) = \frac{\exp(\bar{\mathbf{v}}^\top \mathbf{v}'_{w_{mt}})}{\sum_{w=1}^{V} \exp(\bar{\mathbf{v}}^\top \mathbf{v}'_w)}, \tag{4.4}$$

where $\mathbf{v}'_{w_{mt}}$ is the output vector representation of w_{mt}, and $\bar{\mathbf{v}}$ is averaged vector representation of the context including corresponding q_m, defined as

$$\bar{\mathbf{v}} = \frac{1}{2c+1} (\mathbf{v}_{q_m} + \sum_{-c\le j\le c, j\neq 0} \mathbf{v}_{w_{m,t+j}}), \tag{4.5}$$

where $\mathbf{v}_w$ is the input vector representation of w. Similarly, the probability $\mathbb{P}(q_m | w_{m1} : w_{mT_m})$ is defined as

$$\mathbb{P}(q_m | w_{m1} : w_{mT_m}) = \frac{\exp(\bar{\mathbf{v}}_m^\top \mathbf{v}'_{q_m})}{\sum_{w=1}^{V} \exp(\bar{\mathbf{v}}_m^\top \mathbf{v}'_w)}, \tag{4.6}$$

where $\mathbf{v}'_{q_m}$ is the output vector representation of q_m, and $\bar{\mathbf{v}}_m$ is averaged input vector representation of all the words within the query q_m, computed as

$$\bar{\mathbf{v}}_m = \frac{1}{T_m} \sum_{t=1}^{T_m} \mathbf{v}_{w_{mt}}. \tag{4.7}$$

4.3 Model 3: context-content2vec

Both context2vec and content2vec models have limited modeling power in the light of the available data. This is due to the fact that they are not capable of exploiting all the available query log information as they model only one aspect of the data at hand (i.e., either make use of the context of queries in search sessions or their content, respectively). To overcome this limitation, we propose a two-layer context-content2vec [14] specifically tailored for the purpose of modeling query logs. The two-layer architecture of the proposed model is illustrated in Figure 1c. The upper layer models the temporal context of query sequences in search sessions, based on the assumption that temporally closer queries are statistically more dependent. On the other hand, the bottom layer models the content information of word sequences found within queries.

More formally, given S sessions of queries together with their content, objective of the two-layer context-content2vec model is to maximize the log-likelihood of the training data,

$$\mathcal{L} = \sum_{s\in\mathcal{S}} \sum_{q_m\in s} \Big(\sum_{-b\le i\le b, i\neq 0} \log \mathbb{P}(q_{m+i} | q_m) + \alpha_m \log \mathbb{P}(q_m | w_{m1} : w_{mT_m}) + \sum_{w_{mt}\in q_m} \log \mathbb{P}(w_{mt} | w_{m,t-c} : w_{m,t+c}, q_m) \Big), \tag{4.8}$$

where α weights are hyperparameters that trade off between minimization of the log-likelihood of query sequences (i.e., query context) and the log-likelihood of word sequences (i.e., query content). Denoting frequency of the m^{th} query as K_m, we set the hyperparameters as $\alpha_m = \frac{1}{\log(1+K_m)}$, such that rarely seen queries rely more on content and frequently seen queries more on context.

As can be observed, context-content2vec model combines the two earlier models, using the context2vec model in the upper layer and the content2vec model in the lower layer. Note that the probabilities from equation (4.8) are defined in equations (4.2), (4.4), and (4.6). In order to learn from both contextual and content information, the training data set is a union of context and content training data sets defined previously in sections 4.1 and 4.2.

4.4 Training and inference

The models are optimized using stochastic gradient ascent, suitable for large-scale problems. However, computation of gradients $\nabla\mathcal{L}$ in (4.1), (4.3), and (4.8) is proportional to the vocabulary size V, which is computationally expensive in practical tasks as V could easily reach hundreds of millions. As an alternative, we used negative sampling approach proposed in [31], which significantly reduces the computational complexity.

When the vector representations of all queries are found, we can perform query rewriting in a straightforward manner. For a given user query q, we generate K query rewrites using K nearest neighbors (K-NN) of query q in the low-dimensional space with respect to a cosine distance [31].

5. EXPERIMENTS

In this section we describe the considered data set and give empirical evaluation of the proposed rewriting methods. We learned embeddings for more than 45 million queries, using one of the largest search data sets reported so far, comprising over 12 billion sessions collected on the US website of Yahoo Search. The vectors were used to produce query rewrites, evaluated in terms of relevance and ad coverage and compared to the state-of-the-art methods from the literature.

5.1 Training data sets

5.1.1 Query content data

For purposes of learning query embeddings from query content, we took 45 million most frequent queries and formed dataset $D_{content} = \{(q_m, w_{m1}, w_{m2}, \ldots, w_{mT_m})\}$, where q_m is the m-th query and $(w_{m1}, w_{m2}, \ldots w_{mT_m})$ are the words contained within q_m.

5.1.2 Search session data

Search sessions are defined as uninterrupted sequences of web search activity. Following [16], the session ends when a user is inactive for more than 30 minutes. A new session is initiated with the following search query.

To be able to learn query embeddings from interactions of queries within a search session we created dataset $D_{context} = \{s_i, i = 1, ..., S\}$, derived by sessionizing user search logs data into sessions s_i, that were in turn represented as a set of queries ordered in time, $s_i = (q_{i1}, q_{i2}, ..., q_{iM_i})$. In a case of repetitive queries, such as $s_i = (q_{i1}, q_{i2}, q_{i3} = q_{i2}, q_{i4} = q_{i2}, q_{i5})$, repetitions were de-duplicated to obtain $s_i = (q_{i1}, q_{i2}, q_{i5})$. Finally, search sessions that contained only a single search query were discarded.

5.1.3 Ad clicks and search link clicks

In a search session, queries are often accompanied by ad clicks and search link clicks. These events can be used as additional context to improve query representations and specialize them for a specific task. For example, while one of the main goals of query rewriting in sponsored search is to produce relevant alternatives to the original query, another important goal is for rewrites to match as many bid terms as possible to increase the auction value. Thus, to produce more commercial query rewrites, we use ad click events within query contexts when learning vector representations.

For this purpose we extend the session data set by adding ad click events to user sessions, where each ad click is uniquely identified by ad identification number.

(a) automotive-related ad (b) air travel-related ad

Figure 2: Examples of most similar queries to select ads

The data set $D_{ad} = \{s_i, i = 1, ..., S\}$ consists of sessions s_i comprising both search queries and ad clicks, $s_i = (q_{i1}, q_{i2}, a_{i3}, q_{i4}, a_{i5}..., q_{iM_i})$, where q_{im} refers to search queries, and a_{im} refer to ad click events. Moreover, to further improve learned query representations we also considered adding search link clicks to the user sessions (i.e., clicks that a user made on links given as search results, also referred to as *organic* results), motivated by the idea behind QFG approach. To this end, we expanded data set D_{ad} to obtain data set $D_{ad+link} = \{s_i, i = 1, ..., S\}$, formed by further adding link click events l_{im} to search sessions s_i.

5.2 Training details

5.2.1 Handling "search stopwords"

Unlike in everyday language, where the most frequent tokens are articles and prepositions (e.g., *a*, *the*, *in*), in search data the most frequent tokens are navigational queries, such as *google*, *yahoo*, *facebook*, etc. Due to the fact that over a longer period they occur in a direct neighborhood of majority of other queries, there is a risk that the embedding space will shrink. We have empirically observed this phenomenon, and during training the navigational queries tend to pull all other queries towards them, preventing the vectors from spreading further away in the hyperspace.

The authors of [31] suggested to deal with the frequent tokens in news articles by downsampling. This heuristic involved discarding words with probability $\mathbb{P}(w_i) = 1 - \sqrt{\frac{\tau}{f(w_i)}}$, where $f(w_i)$ is the frequency of word w_i, and τ is a user-set parameter. However, this approach is not applicable in a context of web search which poses quite different requirements. In particular, we aim to learn good representations of navigational queries while at the same time prevent them from adversely influencing representation learning for non-navigational queries. To achieve this objective, we propose a one-direction learning rule for navigational queries. In particular, their vectors are being updated by vectors of queries appearing in the context, but are not used to update the vectors of other queries in their context. We identified navigational queries editorially by evaluating the most frequent 3,000 queries for navigational intent.

5.2.2 Training parameters

Models were trained using a cluster of 9 machines with 256GB of RAM memory and 24 cores. Dimensionality of the

Table 1: Differences in query rewrites of context2vec and context-content2vec for tail queries

Query	cx2vec	cx-cn2vec
minnesota insurance exam crossword puzzles	satellite tv otego	minnesota insurance
	satellite tv menominee	minnesota insurance license practice exams
	satellite tv west end	online insurance exam crossword puzzles
	satellite tv townsend	colorado insurance exam crossword puzzles
	satellite tv lake sara	online minnesota insurance exam crossword puzzles
microwave food safety	staphylococcal enteritis definition	microwave oven food safety
	salmonella enteritis definition	microwave baby food safety
	listeria monocytogenes prevention	microwave food safety studies
	e coli cdc	microwave food safety issues
	preventing cross contamination	foodsafety.com
what to cook in cast iron skillet	steak sauce substitute	cast iron skillet recipes
	montreal seasoning ingredients	how to cook with cast iron skillet
	ground turkey breakfast sausage	how to cook in a cast iron skillet
	how to cook steak on cast iron skillet	how to cook with a cast iron skillet
	reseason cast iron	chicken in cast iron skillet
iphone 6 repair services	mp3attic music	at&t iphone repair service
	donar ovulos en elche	iphone 5c repair service
	credit a la consommation rapide	iphone repair
	smart phone repair service	iphone service repair
	social security disability bronx ny	iphone repair services

Table 2: Examples of query rewrites with and without ad clicks in training data (bolded rewrites matched bid-terms)

makeup (no ads)	makeup (with ads)	snowboarding (no ads)	snowboarding (with ads)	seafood (no ads)	seafood (with ads)
makeup tips	lipstick	snowbaording	snowboards	sea food	**seafood restaurant**
fashion makeup	**mac makeup**	snow boarding	**snowboarding gear**	crab legs	**seafood restaurants**
make up	**makeup sets**	snowboarding information	**burton snowboarding**	best seafood	crab shack
makeup pictures	eye shadow	snowboarding jumps	**snowboard deals**	oysters	**seafood market**
makeup images	**makeup covergirl**	snowboard pics	**snowboards on sale**	lobster recipes	sea food
makeup tutorial	makeup items	shaun white snowboarding	snowboarding mountains	**seafood market**	**sea food menu**

embedding space was set to $D = 300$, context neighborhood size was set to 5 and content neighborhood size was set to 7. Finally, we used negative sampling to speed up the training, and used 10 random samples in each vector update.

5.3 Query rewriting models

We considered the following approaches in our empirical analysis, where each resulted in vector representation for 45 million most frequent queries.

1) word2vec$_{\text{news}}$ model was used as a simple baseline. Query vectors were constructed by summing publicly available word vectors for word tokens in a query (whitespace was used as a token separator), trained on Google News data set with English stopwords removed[1].

2) word2vec$_{\text{search}}$ query vectors constructed by summing word vectors for word tokens in a query, trained using $D_{content}$ data set.

3) content2vec model (cn2vec) was trained using $D_{content}$ where queries were used as a global context to the containing words, as illustrated in Figure 1b.

4) context2vec model (cx2vec) was trained using $D_{context}$, as illustrated in Figure 1a.

5) context-content2vec model (cx-cn2vec) used both $D_{content}$ and $D_{context}$ data to train query vectors, leveraging the two-layer architecture from Section 4.3. Since the model learns from both content and context, one of the motivations behind cx-cn2vec is to improve embeddings for queries that were not seen in many sessions. In Table 1 we give several illustrative examples of such tail queries. Unlike cx2vec, cx-cn2vec relies more on content in case of rare queries, and thus provides better rewrites. For example, at the time of creation of our data set the query "iphone 6 repair service" was a tail query, resulting in poor cx2vec rewrites such as "mp3attic music" or "social security disability bronx ny". On the other hand, cx-cn2vec provided more relevant rewrites.

[1]https://code.google.com/p/word2vec

6) cx-cn2vec$_{\text{ad}}$ model was trained using $D_{context}$ and D_{ad} (i.e., context data with ad clicks added). Training resulted in additional 8.5 million ad vectors.

As illustrated in Table 2, we can see that addition of D_{ad} resulted in quite different query rewrites than using a data set without ad clicks. For example, rewrites for query "makeup" mainly contain terms that are related to tips, tutorials, and pictures when no ads were used in training. Conversely, rewrites for the same query when clicks on ads are considered contain more commercial search terms.

In addition, given ad and query vectors in the same embedding space, we can easily retrieve similar queries for any given ad. This by-product is extremely useful for suggesting new bid keywords for specific categories of ads (note that in our system ads are categorized into one or more interest categories, such as "sports" or "travel"). As an example, in Figure 2 we show $K = 5{,}000$ most similar queries to ads in "automotive" and "air travel" categories. Keywords with higher cosine similarity are shown with larger font sizes. We can observe that the key concepts of the category are well captured with the most similar queries.

7) cx-cn2vec$_{\text{ad+link}}$ model was trained using $D_{content}$ and $D_{ad+link}$ (i.e., context data with ad clicks and link clicks added). Training resulted in additional 19 million search link vectors and 8.5 million ad vectors.

8) QFG$_{\text{ad+link}}$ model was trained using a click-flow graph constructed from $D_{ad+link}$ dataset, with additional 19 million search link vectors and 8.5 million ad vectors.

To produce rewrites for out-of-dictionary queries, embedding models generated their vectors by summing the existing vectors of word tokens within queries (excluding stopwords). Unlike the embedding methods, QFG could not produce rewrites for queries that were not seen in the graph.

Our evaluation did not include topic models such as LDA [5] or PLSA [19], as earlier research [20] found that these methods perform poorly on short text documents.

Table 3: Mean and standard deviation (in brackets) of query similarity (cosine distance) between pairs of editorially judged query rewrites (top: in-dictionary queries, bottom: out-of-dictionary queries)

grade	pairs	cn2vec	cx2vec	cx-cn2vec	cx-cn2vec$_{ad}$	**cx-cn2vec$_{ad+link}$**	word2vec$_{news}$	word2vec$_{search}$	QFG$_{ad+link}$
Excellent	1,518	0.630 (0.136)	0.658 (0.107)	0.669 (0.107)	0.668 (0.114)	**0.733 (0.094)**	0.818 (0.146)	0.648 (0.151)	0.329 (0.667)
Good	5,531	0.599 (0.136)	0.621 (0.125)	0.637 (0.100)	0.632 (0.104)	**0.683 (0.097)**	0.770 (0.152)	0.614 (0.155)	0.205 (0.574)
Fair	4,021	0.550 (0.167)	0.565 (0.129)	0.577 (0.124)	0.566 (0.130)	**0.605 (0.132)**	0.749 (0.190)	0.567 (0.173)	0.114 (0.366)
Bad	4,229	0.398 (0.196)	0.363 (0.170)	0.349 (0.184)	0.336 (0.187)	**0.425 (0.179)**	0.517 (0.280)	0.395 (0.201)	0.166 (0.584)
avg. p-value	-	1.39e-15	5.44e-26	8.27e-28	2.99e-30	**1e-100**	4.121e-07	1.014e-14	0.013
Excellent	2,119	0.791 (0.166)	0.623 (0.141)	0.628 (0.134)	0.623 (0.143)	**0.668 (0.147)**	0.824 (0.145)	0.790 (0.157)	-
Good	11,305	0.752 (0.155)	0.587 (0.135)	0.592 (0.130)	0.584 (0.137)	**0.612 (0.141)**	0.796 (0.145)	0.756 (0.156)	-
Fair	11,146	0.715 (0.136)	0.561 (0.145)	0.565 (0.139)	0.558 (0.146)	**0.584 (0.147)**	0.769 (0.175)	0.707 (0.141)	-
Bad	7,849	0.635 (0.199)	0.383 (0.211)	0.387 (0.209)	0.382 (0.212)	**0.410 (0.208)**	0.509 (0.311)	0.602 (0.208)	-
avg. p-value	-	4.99e-26	1.137e-27	4.0423e-32	1.196e-30	**2.926e-40**	2.038e-16	9.388e-22	-

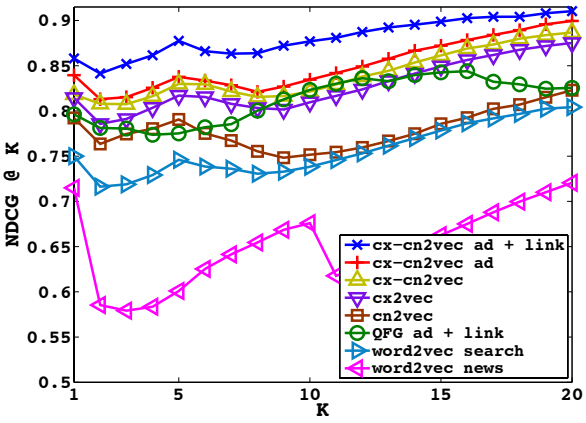


Figure 3: NDCG@K for different competing methods

5.4 Evaluation

In the following two sections we show our main experimental results, where we report performance of the competing approaches in terms of relevance and ad coverage of query rewrites. We also provide examples of rewrites by cx-cn2vec$_{ad+link}$ method in an online video[2].

5.4.1 Relevance

We used editorial judgments of query rewrites to compare query rewriting methods in terms of relevance.

In-house data. The first data set we used is an in-house data set consisting of the query and several rewrites produced by current production system that were graded editorially. The editors were instructed to assign the following grades: *bad*, *fair*, *good*, and *excellent*. In total, the data set includes more than 40,000 (query, rewrite, grade) tuples, such as ("nfl news", "latest nfl news", "excellent"), ("nfl news", "nfl shooting", "fair"), ("nfl news", "nfl rumor", "good"), ("metro transit", "metro pcs", "bad").

Given a query we ranked the rewrite candidates based on a model's output score and retrieved the top K candidates. Next, we computed the NDCG metric using editorial grades of rewrites as labels (0 for bad, 1 for fair, 2 for good, and 3 for excellent) at values of K ranging from 1 to 20. In Figure 3 we report the results for different models.

By considering the reported results, several conclusions can be drawn. Regarding models that did not utilize search session data (i.e., word2vec$_{news}$, word2vec$_{search}$, and cn2vec), we can see that the word embeddings specifically tailored for search queries perform better than using word embeddings learned using news data. In addition, learning query vectors as global context of words using cn2vec leads to slightly better results than directly summing the words. However, all three models performed worse than the QFG$_{ad+link}$ baseline method, which made use of the co-occurrence of queries, ads, and links in search sessions. Query embeddings trained directly on search sessions, i.e. cx2vec, already outperform QFG$_{ad+link}$. Further improvements were observed when embeddings were learned from both context and content. Finally, incorporating ad and link click events showed incremental boost in relevance. The largest gain was observed when links were added as an additional context.

In addition, in Table 3 we report average similarities of (query, rewrite) pairs in each editorial grade group. It is important to note that scores are comparable only within the same method and not across methods. For conclusive comparison we calculated the level of separation between the groups by p-value of t-test, which tests the hypothesis that the means of two neighboring grades are equal. By comparing p-values reported we can quantify which method does the best job of separating the four grade groups. Findings are similar to the ones from Figure 3. Additionally, we find that QFG$_{ad+link}$ method has issues with "bad" grade group, and that without incorporating search context, the embedding models have similar average scores in "good" and "fair" grade groups. As expected, separation improves when embedding models incorporate search context, with the standard deviation between the groups reducing even further when ads and links were considered as an additional context.

In the case when one or both queries from the editorial grade list were not found in a model, we generated the vectors by summing vectors of query tokens. Bottom part of Table 3 shows results for such cases. We can observe that similar conclusion hold even for such out-of-dictionary queries.

TREC data. The second data we used was a publicly available TREC Web Track data set[3] from 2009 to 2013, containing a total of 250 queries. Using the competing methods we produced 5 rewrites for each query, and evaluated the result editorially. The editors were given instructions to rate the rewrites in the following way: grade 0 if the rewrite is irrelevant, grade 1 if relevant, and grade 2 if it is excellent.

[2]http://youtu.be/pvfFQSCYhqI

[3]http://trec.nist.gov/

Table 4: Examples of rewrites obtained using the competing methods (bolded rewrites matched bid-terms)

Original	$\text{QFG}_{\text{ad+link}}$	cn2vec	cx-cn2vec	$\text{cx-cn2vec}_{\text{ad}}$
wedding budget calculator	wedding budget	monthly budget calculator	**wedding planning checklist**	**wedding budget worksheet**
	wedding cost calculator	online budget calculator	wedding budget template	wedding vendors
	wedding calculator	**budget wedding**	**wedding budget worksheet**	**the knot**
	wedding cost breakdown	budget calculator free	wedding checklist printable	wedding planning checklist
	wedding budget worksheet	average wedding budget	wedding costs average	**wedding wire**
gmat prep classes	gmat prep	gmat prep online	**gmat preparation courses**	**gmat study books**
	gmat classes	gmat prep courses	sample gmat tests	**gmat test prep classes**
	gmat	**gmat online prep**	**gmat prep class**	**gmat prep class**
	kaplan gmat course	online gmat prep	which is easier gre or gmat	**kaplan gre courses**
	kaplan gmat	best gmat prep courses	how much is the gmat	free gmat sample tests
how to build a fence	building a fence	how to build fence	**how to build a fence on a hill**	how to build a fence minecraft
	build your own fence	build fence	how to fence a yard	fancy fences and gates
	how to build a wood fence	build a fence	how to build a fence gate	**how to build a metal fence**
	do it yourself fence	how to build a cheap fence	how to build a brick wall fence	**home depot com fencing**
	how to build a privacy fence	build your own fence	**how to build a fence video**	**back yard fences**
solar panels	**solar panels for homes**	solar panels for your home	**solar power**	**solar power**
	solar electric panels	solar panels for residential homes	**solar energy**	**solar panels for homes**
	solar power	**solar panels for**	**solar panels for homes**	**solar panels on sale**
	ebay solar panels	solar panels on ebay	**solar power systems**	**solar panel kits**
	how to make solar panels	davis solar	solar panel	**solar panels for sale**

Editorial grades for each method as well as the Levenshtein distance [29] were averaged and reported in Table 5. We can see that the $\text{cx-cn2vec}_{\text{ad+link}}$ query embedding method, learned from query content and search session context including ad and link clicks, returned the most relevant rewrites and outperformed the baseline QFG method. Once more, learning query embeddings from content of queries on its own was not enough to outperform QFG. Interestingly, there was just a small difference between cx2vec and cx-cn2vec models. It can be explained by the fact that cx-cn2vec generally improves rewrites for tail queries, while TREC data set mostly consists of frequent queries.

In addition to providing highly relevant rewrites, $\text{cx-cn2vec}_{\text{ad+link}}$ showed the highest diversity, a favorable property for rewrite algorithms. Considering the performance in terms of Levenshtein distance, we can observe that rewrites produced by models which learned from content are less diverse than the ones produced by approaches that modeled query context during training.

Examples of query rewrites obtained by the competing methods are given in Table 4. As can be seen, there exist significant differences between the rewrites of the competing approaches. As discussed earlier, the content-based model cn2vec is sensitive to cases when the query words appeared in different contexts across the query dictionary. For example, words "budget" and "calculator" from the query "wedding budget calculator" mostly appeared in more general financial contexts. For this reason in the first 5 rewrites we have queries that are not related to wedding. The model that utilized both content and context data cx-cn2vec outputs highly relevant, interesting rewrites that capture a number of meanings and contexts of the original query. The model that considered ads during training cx-cn2vec_{ad} produced more commercial rewrites, referencing stores and sales. We also bolded queries that matched the actual bidterms in the system, discussed in more detail in the following experiment. Examples of query rewrites produced by $\text{cx-cn2vec}_{\text{ad+link}}$ are shown in an online video[4]. The video mostly covers queries from TREC 2010 data set.

[4]http://youtu.be/n5kHKyKQAa8

Table 5: Comparison of query rewrite methods (TREC data)

Method	Editorial grade	Levenshtein dist.
$\text{QFG}_{\text{ad+link}}$	1.0441	11.70
$\text{word2vec}_{\text{news}}$	0.9189	10.91
$\text{word2vec}_{\text{search}}$	0.9492	11.32
cn2vec	0.9571	11.37
cx2vec	1.1273	13.79
cx-cn2vec	1.1343	13.13
$\text{cx-cn2vec}_{\text{ad}}$	1.2281	13.62
$\text{cx-cn2vec}_{\text{ad+link}}$	**1.2457**	13.25

5.4.2 *Ad coverage*

In the previous section we showed the advantage of proposed methods in terms of relevance. In this section we evaluate how beneficial those rewrites are from the ad matching perspective. It is of fundamental interest to sponsored search to ensure that the rewrites being provided as alternatives to the users' queries match as many additional relevant bidterms as possible. The percentage of rewrites that match bid terms is defined as *coverage*. We conducted an off-line experiment to compare query rewriting methods: 1) on an in-house dataset of 2,000 editorially selected queries which the editors deemed representative of the query set; and 2) on the TREC data set. For each query in the data sets we generated $K = 5$ rewrites and look-up bidterms in our current sponsored search demand. We report average coverage (relative improvement over QFG) over the entire set of queries for each of the two data sets. Table 6 summarizes the results of our analysis.

To estimate monetary value of each query rewriting method we used two proxy measures. First, for the query rewrites produced by each method we look-up the bid amounts for the queries that were matched in the bidterm database. We report the total sum of these amounts, and refer to this metric as "revenue potential". In addition, for each rewrite method we calculated effective cost per mille (eCPM). Given a query rewrite q, and an ad a for which q was a bidterm, we calculated eCPM value e_{qa} by multi-

Table 6: Relative improvement over the QFG method of different query rewrite methods on query-bid data

	In-house data			TREC data		
Method	**Coverage**	**Revenue potential**	**eCPM**	**Coverage**	**Revenue potential**	**eCPM**
$QFG_{ad+link}$	1.00	1.00	1.00	1.00	1.00	1.00
$word2vec_{news}$	0.76	0.39	0.46	0.32	0.66	0.73
$word2vec_{search}$	0.87	0.58	0.77	0.57	0.84	0.75
cn2vec	0.89	0.62	0.84	0.59	0.84	0.74
cx2vec	1.16	1.80	1.41	1.41	1.16	1.20
cx-cn2vec	1.18	1.86	1.38	1.44	1.21	1.19
cx-cn2vec$_{ad}$	**1.20**	**1.89**	**1.60**	**1.52**	**1.35**	**1.31**
cx-$cn2vec_{ad+link}$	1.18	1.88	1.45	1.50	1.28	1.22

plying cost per click (CPC) dollar amount and the click-through rate (CTR) of that (query, ad) pair, computed as number of ad clicks divided by number of ad impressions. Finally, we report the weighted sum of eCPM's, $\sum_q \sum_a w_q e_{qa}$, where the weight w_q is proportional to the number of times the query appeared in the search logs. Considering the results presented in Table 6, we can see that query embedding models trained using news documents and query content achieved lower average coverage than QFG method. However, QFG was in turn outperformed by the coverage of rewrites produced using query embedding approaches.

Moreover, by taking the results from both experiments into account, it can be concluded that cx-$cn2vec_{ads+links}$ is the best choice as it achieves the highest relevance while maintaining large ad coverage. When link clicks were added on top of ad clicks we observed a large improvement in relevance, with ad coverage remaining almost the same.

6. CONCLUSION

In this paper we described novel query rewriting methods based on recently proposed neural language models. The methods learn low-dimensional, distributed representations of search queries based on context (context2vec), content (content2vec), or combined context and content (context-content2vec) of the queries within search sessions. To specialize the query rewrites for sponsored search application, we further incorporated ad clicks and search link clicks into the training data. We evaluated the proposed methods using both in-house and publicly available TREC data sets. When compared to the current state-of-the-art approaches, we showed that context-content2vec generates the most relevant query rewrites, while at the same time maintains high level of ad coverage. The results clearly indicate significant advantages of context-content2vec over the state-of-the-art query rewrite algorithms, and suggest high monetization potential of the query embedding approach to the task of sponsored search advertising. In our ongoing work, we plan to experiment with more involved search sessionization algorithms and navigational query detection algorithms.

7. REFERENCES

[1] M. Aly, A. Hatch, V. Josifovski, and V. K. Narayanan. Web-scale user modeling for targeting. *WWW*, 2012.

[2] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology*, EDBT'04, pages 588–596, Berlin, Heidelberg, 2004. Springer-Verlag.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[7] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. Efficient query recommendations in the long tail via center-piece subgraphs. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 345–354, New York, NY, USA, 2012. ACM.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[9] D. E. Bowman, M. L. Hamrick, T. R. Kohn, R. E. Ortega, and J. R. Spiegel. Refining search queries by the suggestion of correlated terms from prior searches, Dec. 21 1999. US Patent 6,006,225.

[10] A. Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[11] A. Z. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search advertising using web relevance feedback. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1013–1022, New York, NY, USA, 2008. ACM.

[12] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. *KDD*, 2009.

[13] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[14] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hierarchical neural language models for joint representation of streaming documents and their content. In *International World Wide Web Conference (WWW)*, 2015.

[15] D. C. Fain and J. O. Pedersen. Sponsored search: A brief history. *Bulletin of the American Society for Information Science and Technology*, 32(2):12–13, 2006.

[16] D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Inf. Sci.*, 179(12):1822–1843, May 2009.

[17] M. Grbovic, N. Djuric, V. Radosavljevic, and N. Bhamidipati. Search retargeting using directed query embeddings. In *International World Wide Web Conference (WWW)*, 2015.

[18] M. Grbovic and S. Vucetic. Generating ad targeting rules using sparse principal component analysis with constraints. *WWW*, 2014.

[19] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[20] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88. ACM, 2010.

[21] A. K. Jain, L. Hong, and S. Pankanti. Iab internet advertising revenue report: 2013 first six months' results. Technical report, Interactive Advertising Bureau, 2013.

[22] B. J. Jansen and T. Mullen. Sponsored search: An overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131, 2008.

[23] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396. ACM, 2006.

[24] F. Keller and M. Lapata. Using the web to obtain frequencies for unseen bigrams. *Computational linguistics*, 29(3):459–484, 2003.

[25] R. Kiros, R. Zemel, and R. Salakhutdinov. Multimodal neural language models. In *Proceedings of the 31th International Conference on Machine Learning*, 2014.

[26] R. Kiros, R. S. Zemel, and R. Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. *arXiv preprint arXiv:1406.2710*, 2014.

[27] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR*, pages 120–127. ACM, 2001.

[28] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[29] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics-Doklady*, volume 10, 1966.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[32] S. Pandey, M. Aly, A. Bagherjeiran, A. Hatch, P. Ciccolo, A. Ratnaparkhi, and M. Zinkevich. Learning to target: what works for behavioral targeting. *CIKM*, 2011.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. *arXiv preprint arXiv:1403.6652*, 2014.

[34] PwC. Global entertainment and media outlook: 2014-2018. Technical report, 2014.

[35] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, volume 33, pages 6–12. ACM, 1999.

[36] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4:1–174, Jan. 2010.

[37] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[38] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[39] H. Vahabi, M. Ackerman, D. Loker, R. Baeza-Yates, and A. Lopez-Ortiz. Orthogonal query recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 33–40, New York, NY, USA, 2013. ACM.

[40] W. V. Zhang and R. Jones. Comparing click logs and editorial labels for training query rewriting. In *WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges*, 2007.