Scalable Query N-Gram Embedding for Improving Matching and Relevance in Sponsored Search

Xiao Bai
Yahoo Research
xbai@oath.com

Erik Ordentlich
Yahoo Research
eord@oath.com

Yuanyuan Zhang
Yahoo Research
yuanyuanz@oath.com

Andy Feng*
Nvidia
andyf@nvidia.com

Adwait Ratnaparkhi*
Roku Inc.
aratnaparkhi@roku.com

Reena Somvanshi
Oath Inc.
reen@oath.com

Aldi Tjahjadi
Oath Inc.
aldi@oath.com

ABSTRACT

Sponsored search has been the major source of revenue for commercial web search engines. It is crucial for a sponsored search engine to retrieve ads that are relevant to user queries to attract clicks as advertisers only pay when their ads get clicked. Retrieving relevant ads for a query typically involves infi rst matching related ads to the query and thenfi ltering out irrelevant ones. Both require understanding the semantic relationship between a query and an ad. In this work, we propose a novel embedding of queries and ads in sponsored search. The query embeddings are generated from constituent word n-gram embeddings that are trained to optimize an event level word2vec objective over a large volume of search data. We show through a query rewriting task that the proposed query n-gram embedding model outperforms the state-of-the-art word embedding models for capturing query semantics. This allows us to apply the proposed query n-gram embedding model to improve query-ad matching and relevance in sponsored search. First, we use the similarity between a query and an ad derived from the query n-gram embeddings as an additional feature in the query-ad relevance model used in Yahoo Search. We show through online A/B test that using the new relevance model tofi lter irrelevant ads offline leads to 0.47% CTR and 0.32% revenue increase. Second, we propose a novel online query to ads matching system, built on an open-source big-data serving engine [30], using the learned query n-gram embeddings. Online A/B test shows that the new matching technique increases the search revenue by 2.32% as it significantly increases the ad coverage for tail queries.

*The work was done when the authors were with Yahoo Research.

KDD 2018, August 19-23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219897

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Information systems** → **Sponsored search advertising**;

KEYWORDS

N-gram embedding; sponsored search; query-ad matching; query-ad relevance; distributed training; Apache Spark; Vespa.

ACM Reference Format:
Xiao Bai, Erik Ordentlich, Yuanyuan Zhang, Andy Feng, Adwait Ratnaparkhi*, Reena Somvanshi, and Aldi Tjahjadi. 2018. Scalable Query N-Gram Embedding for Improving Matching and Relevance in Sponsored Search. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Sponsored search has been the major source of revenue for commercial web search engines and has significantly contributed to the multi-billion dollar online advertising industry. In sponsored search, an advertiser only pays if its ads get clicked by the user. Under this pay-per-click business model, it is crucial for commercial search engines to retrieve advertisements that are highly relevant to user' search intent in order to attract clicks.

Retrieving relevant ads for a query is similar to retrieving relevant web pages for a query in web search, but there is one major difference. In sponsored search, advertisers can define how they want their ads to be matched to a query through different match types, while in web search there is no such restriction. Therefore, instead of directly retrieving ads based on their relevance as in web search, ads are typically matched to queries through techniques of different match types, such as Exact match, Phrase match, and Broad match. Relevant ads are then selected byfi ltering irrelevant ones using machine learned query and ad relevance model. Clearly, both query to ads matching and query and ad relevance modeling require good understanding of queries and ads, especially the semantic relationship between them.

In sponsored search, an ad is usually composed of a title, a description, a display URL and a landing URL. Ad title and description

expresses the main message an advertiser wants to pass through the ad. Clicking on an ad will bring a user to the landing page of the advertiser through landing URL. A display URL can be considered as an anchor text of the landing URL to make the overall ad look more attractive. An advertiser also provides a list of bidded terms for each ad, expressing for which queries she wants the ad to be shown, and a bid price she wants to pay when the ad gets a click.

Each bidded term is associated with a match type, which can be Exact, Phrase or Broad. Exact match requires the bidded term associated with a retrieved ad to be exactly the same as the user query. Phrase match requires that the bidded term associated with a retrieved ad should be a sub-phrase of the query. Broad match only requires the bidded term associated with a retrieved ad to be related to the query and there is no restriction on the words used. Exact match is the most common match type for query-ad matching. Although advertisers have strong motivation to provide relevant bidded terms to their ads, it is almost impossible for them to come up with a comprehensive list of bidded terms that can catch all the users (queries) that may be interested in their products or businesses. The difficulty of enumerating all relevant bidded terms also limits the users Phrase match can reach. Broad match is more flexible for retrieving ads. A common approach for Broad match is query rewriting that rewrites user queries to similar ones such that the rewrites can trigger some ads through Exact or Phrase matches [4, 10, 13, 23]. Most of these approaches heavily rely on historical information of known queries and can only be used in an offline fashion to produce a table or index of query rewrites for a set of anticipated queries. More recently, with the success of distributed language models in natural language processing, a new Broad match approach that does not rely on query rewriting has been proposed. Grbovic et al. [9] propose to learn query and ad embeddings from sponsored search sessions and use the cosine similarity between query and ad vectors offline to produce a Broad match index of top ads for each query having an embedding. Although this approach is able to retrieve relevant ads for queries that have been seen in the past, it has no coverage for queries that are new to the search engine and ads that have not been clicked in any search session in the past. This gap in coverage naturally brings the necessity of having a morefl exible embedding approach to capture the semantic similarity between a query and an ad for better Broad match.

Moreover, commercial web search engines mainly rely on text-based features extracted from queries and ads to model the relevance between them. As shown in [1], by avoiding click-based features as in earlier relevance models [11] for sponsored search, the resulting relevance model becomes much more robust to tail query and ad pairs that have little or no click history but account for a significant fraction of matched query-ad pairs. However, the relevance model proposed in [1] only uses two sets of Latent Semantic Indexing (LSI) [5] related features and all the other features are purely based on textual overlap between query and ad. There is clear room to improve the model if features capturing better semantic similarity between query and ad are available.

In this paper, we propose to learn low-dimensional representations for the word n-grams appearing in search queries using a distributed language model and historical sponsored search logs. Specifically, we consider queries, ad clicks and search result clicks that happen in the same search session as surrounding context to each other to learn word2vec-style [18] vectors for queries, ad clicks and search result clicks but where the input vector for a query is further modeled by applying a combining function, such as a component-wise average, to the vectors corresponding to its word n-grams. By learning an input vector representation at the word n-gram level instead of the query level, this approach can address the problem of no coverage for new queries as the full query representation can be estimated by applying the combining function used during training to the corresponding n-gram representations. Moreover, since the representations are learned from search sessions and users usually type several queries that are related to each other to accomplish a search task, the representations allow capturing semantic similarity between a query and an ad that goes beyond simple textual overlap.

In this paper, we focus on sharing our experience in applying the proposed query n-gram embedding to improve the modeling of query-ad relevance and to improve Broad match for Yahoo Search. The main contributions of this work are as follows:

- We propose a novel query n-gram embedding as well as a large-scale training pipeline for training the embedding. We show through a query rewriting task that the learned query n-gram embedding can significantly increase the coverage for new queries compared to a query-level embedding while better capturing editorial ratings than this and other baseline approaches.
- We present a novel semantic feature based on the learned query n-gram embedding that can be leveraged to improve the query-ad relevance model used in Yahoo Search. The relevance model leveraging the new feature improves the AUC of the model without the feature by 1.80%. We apply the new relevance model as afi lter to remove irrelevant Broad match ads in Yahoo Search. A/B testing results on live traffic of Yahoo Search demonstrate that the proposed system can improve the ad CTR by 0.47% and search revenue by 0.32% while reducing the number of ads shown to users.
- We present a novel online Broad match system, built on an open-source big-data serving engine [30], using the learned query n-gram embedding. We use this system to retrieve Broad match ads for a large variety of queries. A/B testing results on live traffic of Yahoo Search demonstrate that the proposed system can improve the ad coverage and search revenue by 2.32% and 2.12% respectively.

2 RELATED WORK

In this section, we review the related work on semantic representation learning, query and ad Broad matching, as well as query and ad relevance modeling in sponsored search.

Semantic representation learning. Word representation learning has been an active research area for some time [5, 16] and it has been shown to significantly improve many NLP applications [28, 29]. Traditional language models represent words either as high-dimensional sparse vectors through one-hot encoding or as low-dimensional dense vectors through word co-occurrance [16, 26] or latent semantic analysis [5]. More recently, neural language models are used for learning distributed word representations [2, 27]

and such models have attracted particular attention since the development of the highly scalable continuous bag-of-words (CBOW) and skip-gram with negative sampling (SGNS) models [17, 18].

Following the same concept as [17, 18] that relies on the context of a word to learn the word representation, Grbovic et al. [9] propose to learn semantic representation of search events, i.e., queries and ads in sponsored search, by setting the context of an event to include a number of preceding and succeeding events in the search session in which it occurs. This Search2Vec model shows better performance than the word2vec approaches [18] in a query to ad matching task but suffers from the cold-start problem as there is no coverage for new queries and tail queries. As a result, the application of this model is limited given the high dynamics of the queries that a commercial web search engine may receive. The query word n-gram model we propose in this work particularly addresses the coverage problem of Search2Vec [9] by learning representations for constituent words of queries.

The idea of applying the word2vec training objective (SGNS and/or CBOW) [18] to generating vectors for longer phrases and documents from constituent word (i.e., subphrase) vectors and optimizing the generation model through training has also been considered recently in other settings [3, 14, 15, 19, 21]. In these works, the subphrase vector models are also applied to the output (or context) phrases. Bojanowski el al. [3] applies modeling analogous to this work but to partial word units (as opposed to phrase units like in this work), such as letters and syllables, and does retain output vectors for entire words. This requires less memory due to smaller vocabulary size at the word level. We are not aware of any work in which the vectors are trained freely for output phrases/sentences (e.g., queries in our work), but in a constrained and generated manner for input phrases. Our large vocabulary training system [20] is a key to enable this approach as we need to store output vectors for up to several hundred million generalized words/phrases.

In the context of search, there are other works that learn semantic representations by mining the search clickthrough graphs [7, 12]. Jiang et al. [12] propose ClickSim that relies on several iterations of content information propagation on the web search click graph to learn high-dimensional vectors for both queries and web documents. As shown in the experiments, our query word n-gram model helps to capture semantic similarity between queries and ads better than ClickSim by taking into account the event context of queries and ads in the same sessions.

Broad match of query and ad in sponsored search. Broad match has been an effective way for advertisers to expend their audiences beyond Exact and Phrase matches in sponsored search. A common approach for Broad match is to rewrite user queries to similar ones such that the rewrites can be matched to relevant ads via Exact or Phrase matches [10, 13, 23]. Most of the existing works rely on various sources of external knowledge, such as queries in the same search sessions [10, 13] and pseudo-relevance feedbacks from algorithmic search results [23], to perform off-line query rewriting. There approaches can only be applied offline to repeating queries that appear in the head or torso of the query distribution. To expand Broad match to tail queries, Broder et al. [4] proposes an online query rewriting approach that builds an inverted index for popular queries seen in the past with unigrams, bigrams and taxonomy features offline to enable online retrieval of similar queries against the index. The Broad matching approach we propose also works online, but it is different from [4] as it relies on query and query n-gram embeddings to capture semantics of words while [4] mostly relies on syntatic match, and our matching is based on query and ad similarity directly while [4] does matching through query rewriting. Similar to us, Grbovic et al. [9] also propose to do Broad match using query and ad embeddings learned from search sessions. However, the Search2Vec model they propose only learns embeddings for entire queries, limiting its application to tail and new queries for which no vector representations are learned.

Query-ad relevance in sponsored search. Relevance is a long standing problem for web search. There are relatively fewer works [1, 11, 24] that deal with relevance between query and ad in sponsored search because relevance is not directly used for ad retrieval and click-through-rate (CTR) prediction attracts more attention for revenue improvement. Raghavan el al. [24] implement a language model and demonstrate significant performance improvements over the baseline vector space model (TF-IDF) system in their sponsored search system. Hillard et al. [11] develop a machine learning model that uses past user clicks in addition to text overlap between query and ads to measure the relevance between a query and an ad. Opposite to [11], Aiello et al. [1] claim that click-based features risk to bias relevance modeling toward head queries and ads that have rich click history. Instead, they propose a machine learning model that only relies on text-based features and discuss several successful applications of their model in a commercial web search engine. The text-based features used in [1] are either syntactic or based on word co-occurrance so that they only marginally capture semantics between queries and ads. The query word n-gram embedding model we propose can be used to generate semantic features to improve a machine learning relevance model similar to that of [1].

3 SEMANTIC QUERY N-GRAM EMBEDDING

To address the above discussed shortcomings of the state-of-the-art query-ad matching and relevance modeling approaches, we propose in this section a better embedding approach that learns vector representations at word n-gram level from sponsored search sessions. This allows generating semantic representations for new and tail queries and ads, on-the-fly, by composing the learned query n-gram embeddings. Wefi rst describe how we learn the query n-gram embeddings at scale through distributed training in Section 3.1 and 3.2. We then report the quality of the learned embeddings through a query rewriting task in Section 3.3.

3.1 Learning Query N-Gram Representations

Similarly to [9, 10], the proposed embedding is derived from a large corpus of search session event data comprising user queries, search result clicks, and ad clicks appearing in order of event occurrence. A session is defined as an uninterrupted sequence of such events and a new session is considered to start when a user does not have any event for more than 30 minutes since her last event [8]. Queries are presented in their text formats, while search result clicks and ad clicks are encoded with an ID that uniquely identifies a web document hyperlink or an ad.

In the previously proposed Search2Vec model [9], embeddings were obtained for all sufficiently frequently occurring query, ad click, and search result clicks using the skip-gram with negative examples (SGNS) word2vec algorithm [18]. This raises a significant problem as the embedding are not applicable to any new and tail queries, accounting for about 50% queries a search engine receives. To address this problem, instead, we proposed to learn embeddings not only for entire queries, ads, and search links, but also for unigrams, bigrams, and more generally, n-grams of word appearing in queries. These n-gram vectors can then be combined to synthesize vectors for entire queries, and even ad titles, descriptions, and display URLs as we will see in Section 4.1. As explained below, the query word n-gram vectors along with the other types of vectors are trained to optimize the SGNS word2vec objective.

More formally, the vocabulary or the set of "words" for which vectors are trained includes queries, ads, and search links, respectively denoted by $\mathcal{Q}$, $\mathcal{A}$, and $\mathcal{H}$. In the proposed system, the vocabulary also includes $\mathcal{SQ}$, which comprises a set of word n-grams of queries in $\mathcal{Q}$. For example, if the query *"lazy boy recliner sofas"* is in $\mathcal{Q}$, then $\mathcal{SQ}$ may include the unigrams *"lazy"*, *"boy"*, *"recliner"*, *"sofa"*, and possibly the bigrams *"lazy boy"*, *"boy recliner"*, *"recliner sofa"*, and more generally, trigrams, e.g., groups of three consecutive words in the query, etc..

Recall that in SGNS Search2Vec modeling a pair of "input" and "output" vectors $\mathbf{u}(w)$ and $\mathbf{v}(w)$ are trained for each vocabulary word w. In the proposed system, this is still the case for "words" corresponding to clicked ads and search links, but not for queries and query n-gram words. In the case of queries ($w \in \mathcal{Q}$), only output vectors $\mathbf{v}$ are trained, and in the case of query n-gram words, only input vectors $\mathbf{u}$ are trained. Given a mapping $\mathbf{sw} : \mathcal{Q} \rightarrow \mathcal{SQ}^*$, from query words to a sequence of word n-grams of the query (e.g., unigrams and/or bigrams appearing in the query) and a model $\mathbf{m}$ for combining a *sequence* of vectors: (e.g., component-wise average: $\mathbf{m}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m) = (1/m)\sum_j \mathbf{v}_j$), for $w \in \mathcal{Q}$, if $\mathbf{sw}(w) = sw_1, sw_2, sw_3, \ldots, sw_m$, we set

$$\mathbf{u}(w) = \mathbf{m}(\mathbf{u}(sw_1), \mathbf{u}(sw_2), ..., \mathbf{u}(sw_m)). \quad (1)$$

The original SGNS word2vec training objective is then used to optimize the various vectors, including the n-gram vectors $\mathbf{u}(sw_j)$, assuming the above procedure for generating $\mathbf{u}(w)$ when $w \in \mathcal{Q}$. More specifically, given a corpus consisting of a sequence of search sessions $s_1, s_2, \ldots, s_n$ each comprising a sequence of "words", i.e., $s_i = w_{i,1}, w_{i,2}, \ldots, w_{i,m_i}$, with $w_{i,j} \in \mathcal{V} = \mathcal{A} \cup \mathcal{Q} \cup \mathcal{H}$, the objective is to maximize the log likelihood:

$$\sum_{i=1}^{n} \sum_{j:w_{i,j}\in\mathcal{V}} \sum_{\substack{k\neq j:|k-j|\leq b_{i,j} \\ w_{i,k}\in\mathcal{V}}} \Bigg[\log \sigma\big(\mathbf{u}(w_{i,j})\mathbf{v}^{\mathsf{T}}(w_{i,k})\big) + \sum_{\tilde{w}\in\mathcal{N}_{i,j,k}} \log\Big(1 - \sigma\big(\mathbf{u}(w_{i,j})\mathbf{v}^{\mathsf{T}}(\tilde{w})\big)\Big)\Bigg] \quad (2)$$

over input (row) vectors $\mathbf{u}(w)$ for $w \in \mathcal{A} \cup \mathcal{H} \cup \mathcal{SW}$ and output (row) vectors $\mathbf{v}(w)$ for all $w \in \mathcal{A} \cup \mathcal{H} \cup \mathcal{Q}$, with $\mathbf{u}(w)$ generated according to (1) when $w \in \mathcal{Q}$, where, additionally,

- $\sigma(\cdot)$ denotes the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$;
- window sizes $b_{i,j}$ are randomly selected so that each inner sum includes between 1 and a maximum B terms, as in [18] and its open–source implementation;
- negative examples $\mathcal{N}_{i,j,k}$ associated with positive output word $w_{i,k}$ are selected randomly according to a probability distribution suggested in [18];
- the vocabulary $\mathcal{V}$ consists of the set of "words" for which vectors are to be trained.

Training involves minibatch SGD optimization of (2) with respect to the various query, ad, hyperlink, and query word n-gram vectors.

A key feature of this approach is that, the query word n-gram vectors are optimized in a closed-loop-fashion along with all of the other vectors to yield good performance with respect to the original Search2Vec [9] training objective. This is motivated by the excellent results obtained using this objective when query input vectors are freely optimized as in [9], with the philosophy that we should depart from this framework as little as possible while enabling some desired functionality, which, in this case, is the ability to generate query vectors "on thefl y" using the corresponding n-gram vectors. Specifically, given a query w = *"lazy boy recliner sofas"*, $\mathbf{sw}(w)$ yields m unigrams and bigrams of w as $\mathbf{sw}$("lazy boy recliner sofas") = ("lazy", "boy", "recliner", "sofas", "lazy boy", "boy recliner", "recliner sofas")), the vector representation of w is obtained through component-wise average $\mathbf{m}$:

$$\mathbf{m}(\mathbf{u}(sw_1), \mathbf{u}(sw_2), ..., \mathbf{u}(sw_m)) = \frac{1}{m}\sum_j \mathbf{u}(sw_j)$$

For computational convenience, during training, we restrict the modeling of query vectors using query word n-gram vectors to queries with 8 or fewer words. In our data sets, this accounts for about 97% of all query instances. Longer queries are treated as single "words", like the ads and search links. The models, however, are applied without query length limitations in our evaluations and applications.

3.2 Distributed Training at Scale

The proposed query n-gram model, referred to as QueryNGram2Vec, has lower memory requirements for training than a full SGNS Search2Vec model since it is not necessary to maintain an *input* vector for each query as these are generated from the query n-gram vectors. Nevertheless, the memory requirements may still be substantial since we continue to have output vectors for all queries and both input and output vectors for search links and ads. Also the QueryNGram2Vec model training has to accommodate tens of millions of query n-gram vectors. We aim to experiment at a scale of 38 billion queries and ad and search link clicks, and a vocabulary having ~40 million search links, ~20 million ads, ~120 million queries, and ~50-60 million query n-grams. Storage of the corresponding 300 dimensional input and output vectors during training requires close to 350GB of memory, continuing the need for distributed training on our commodity machine clusters.

To this end, we extend the distributed Search2Vec training system of [20] to handle the query n-gram optimization algorithm explained above. The system, as shown in Figure 1, is similar to the one described in [20], with the query n-gram vectors, like the other vectors from before, sharded by dimension across a number of

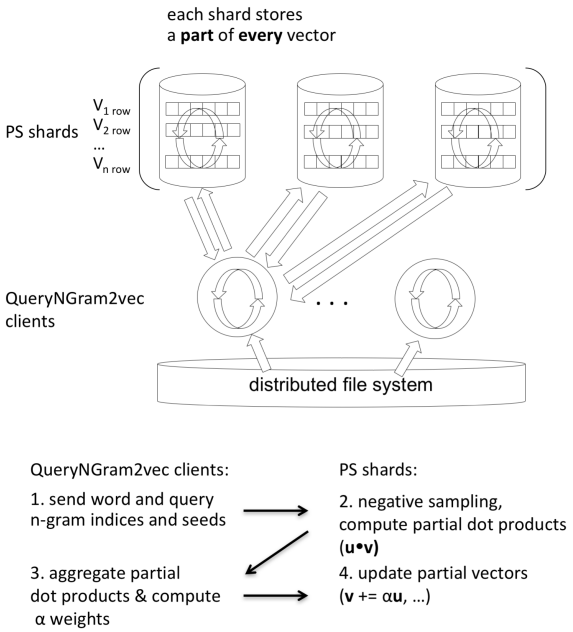

Figure 1: Distributed training of QueryNGram2Vec model.

parameter servers. The communication and computation proceeds as before, except that now query n-gram indices are also communicated from clients to server shards and the shards retrieve the corresponding query n-gram vectors from their local memory stores to compose the query vector components for computing the partial dot products with their respective context vector components, as required by the distributed SGD update. Similarly, the additional per-shard SGD updates required for the query n-gram vectors are computed and applied locally on each shard. As a result, the additional query n-gram vectors (like all of the other vectors) are not transmitted over the network and their inclusion incurs only minimal additional network bandwidth. Using this system, it currently takes about 40 hours to train a QueryNGram2Vec model of the size mentioned above, and, invoking the terminology of [20] (Section 6.2), using 15 parameter server shards, 10 iterations, 200 clients with 8 threads each, a minibatch of 200, a subsampling threshold (see [18]) of 1e-5, a maximum window size of 10 (including both directions), and 5 negative examples per positive.

3.3 Query N-Gram Representations Evaluation

We train the query word n-gram model proposed in Section 3.1 using search session data from Yahoo, as described above. We discard all sessions having only 1 event and all queries, ad clicks and search result clicks that occurred less than 10 times in the training window. The model consists of 300-dimensional vectors for 104.9 million queries, 3.4 million unigrams, 43.8 million bigrams, 18.5 million ads, and 40 million search links. The models evaluated are trained using the system described in Section 3.2 and parameters mentioned therein. Additionally, the learning rate alpha was set to 0.02 and reduced to 0.002 in equal steps after each full iteration.

To evaluate the quality of the learned query word n-gram representations and how well they can capture the semantics of the n-grams appearing in queries, we conducted a query rewrite experiment. We used an annotated dataset that contains 141K query to query rewrite pairs that are editorially labeled as Perfect, Excellent, Good, Fair or Bad. This dataset consists of 25K unique queries and each query has at least 2 rewrites. We used the cosine similarity between the vectors of the query and the rewrite to rank the rewrites of each query and computed NDCG (Normalized Discounted Cumulative Gain) scores for the query. We used the average NDCG@k with k=1, 3, 5 across all queries to compare the proposed model with the following baselines.

- **BM25 [25]**: We use the BM25 similarity between query and rewrite to rank the rewrites of each query.
- **GloVe [22]**: We use the 300-dimensional GloVe word vectors that are learned through the aggregated global word-word co-occurrence statistics in Wikipedia 2014 and Gigaword 5[1]. We consider GloVe as a baseline since it shows better performance than CBOW [17] for several NLP tasks. The vocabulary size is 400K. We average the vectors of the unigrams appearing in a query or a rewrite as its vector representation and use the cosine similarity between query and rewrite to rank the rewrites of each query. If no vector exists for a query or a rewrite, their cosine similarity is set to 0.
- **Search2Vec [9]**: We train the Search2Vec model using the same dataset as the proposed QueryNGram2Vec model. The only difference is that instead of learning representations for unigrams and bigrams in the queries, representations are directly learned for the entire queries. The model consists of 300-dimensional vectors for 104.9 million queries, 18.5 million ads, and 40 million search links. If the vector of a query or a rewrite does not exist in the vocabulary, their cosine similarity is set to 0.
- **QueryNGram2Vec-Unigram**: This is a variant of the proposed QueryNGram2Vec model trained with the same dataset. Instead of learning representations of both unigrams and bigrams appearing in queries, only those of unigrams are learned in addition to representations for queries, ads, and search links. The model consists of 300-dimensional vectors for 104.9 million queries (output vectors only), 3.4 million unigrams, and 18.5 million ads. A query vector is obtained by averaging only the vectors of its unigrams. If no unigram vector exists for either a query or its rewrite, their cosine similarity is set to 0.
- **QueryNGram2Vec-Bigram**: This is another variant of the proposed QueryNGram2Vec model trained with the same dataset. Only vectors of bigrams appearing in queries are learned in addition to query vectors, ad vectors, and search link vectors. The model consists of 300-dimensional vectors for 104.9 million queries (output vectors only), 43.8 million bigrams, 18.5 million ads, and 40 million search links. A query vector is obtained by averaging only the vectors of its

[1] http://nlp.stanford.edu/data/glove.6B.zip

bigrams. If no bigram vector exists for either a query or its rewrite, their cosine similarity is set to 0.

Table 1 summarizes the coverage of the above models on the query-rewrite dataset and the NDCG@k for ranking query rewrites. We observe that although QueryNGram2Vec does not have 100% coverage for the query-rewrite pairs we consider, as the model includes n-grams only for queries that happened at least 10 times in the training data, it leads to up to 2.9% higher NDCG than BM25 that relies on pure text match. We also observe that QueryNGram2Vec almost doubles the coverage of Search2Vec while having 7.4% higher NDCG@1 than Search2Vec. Since Search2Vec learns the representations at query level, it fails to model new queries and tail queries that rarely occurred in the past. By constructing the query vectors using the vectors of corresponding unigrams and bigrams, QueryNGram2Vec is able to achieve better NDCG than Search2Vec as well as its alternatives that use only unigram or bigram vectors. These demonstrate the ability of the proposed QueryNGram2Vec model to learn query word n-gram semantics from search logs.

4 APPLICATIONS OF SEMANTIC QUERY N-GRAM REPRESENTATIONS

In this section, we describe two applications of the proposed query n-gram embedding model in sponsored search. Thefi rst application is to use the learned query n-gram representations to capture the semantic similarity between query and ad, and then use the similarity as a feature to improve a query-ad relevance based ad qualityfi lter. The second application is a new Broad match algorithm that works in an online fashion to retrieve relevant ads for queries that are lying in the tail of the frequency distribution. We provide details on the application scenario, offline evaluation, system implementation, and online performance for the two applications in Section 4.1 and Section 4.2, respectively.

4.1 Improving Query-Ad Relevance Modeling

4.1.1Qu ery-ad Relevance Model. In sponsored search, advertisers only pay a search engine if their ads get clicked [6]. Showing relevant ads to users is the only way to ensure clicks. Clearly, ad relevance is crucial for both user experience and search engine revenue. However, different from web search where documents are retrieved based on relevance, ads are mainly matched to ads through Exact and Phrase matches depending on advertiser provided bidded terms. Although Broad match provides a morefl exible way of matching ads, most Broad match algorithms rely on query rewriting techniques [4, 10, 13, 23] and cannot directly assess the relevance between a query and an ad. Therefore, relevance evaluation mainly happens as an ad qualityfi lter after the matching phase in a sponsored search system [1].

In our system, query and ad relevance is modeled using a Gradient Boosted Decision Tree (GBDT). The features are text based and are extracted from query and ad title, ad description and ad display URL. As shown in [1], avoiding using click history as a feature makes the query-ad relevance model robust to tail query and ad pairs that has little or no history. This is particularly important as the majority of ads in the ad inventory that get retrieved at matching phase do not have a chance of being shown to users and thus do not have any click history.

Before incorporating the new feature based on the proposed query n-gram embedding, our relevance model relies on 185 text-based features as reported in [1] and 3 what we call ClickSim [12] features. The 185 text-based features mainly consist of the following feature groups: **common counts** and **Jaccard similarity** of unigrams, bigrams, q-grams[2] between query and each ad component (i.e., title, description and display URL); **TF-IDF**, **BM25** and **LSI [5] cosine similarities** between query and each ad component; **common brands** between query and each ad component; **semantic coherence** (i.e., max, min and mean) of the cosine similarities between the LSI representations of any two ad components; and **hash embedding** of each co-occurrence of unigrams in a query and an ad component. **ClickSim** [12] was proposed by Jiang et al. [12] as a vector propagation algorithm on the web search click graph to learn vector representations for queries and documents in the semantic space. We use the cosine similarities between a query and each ad component as features for our query-ad relevance model. Similar features have also shown significant improvement in a learning-to-rank model for web search [12].

As we can see, the only features that are able to capture semantic similarity between query and ad are LSI and ClickSim. We observe that LSI is very generic and ClickSim is learned from web search and thus may not capture semantics of ads in the best way. This motivates us to use the proposed query n-gram embedding to compute additional semantic features for our query-ad relevance model.

4.1.2Qu eryNGram2Vec as a Feature inQu ery-Ad Relevance model. We compute 3 features that measure the cosine similarity between the vectors of query and ad title, description and display URL respectively. Recall that our query n-gram embedding model learns vector representations for unigrams and bigrams appearing in queries, as well as vector representations for the entire ads. For either a query or an ad component, wefi rst generate all the possible unigrams and bigrams from its text, we then average their corresponding QueryNGram2Vec vectors to obtain its vector representation. Although the model also learns vectors for some ads having a click history, this accounts for a small fraction of all the query-ad matching candidates on which the query-ad relevance model is applied. We thus decide to rely on unigram and bigram vectors to construct ad vectors from the ad textfields.

We train a GBDT relevance model with 870K editorially labelled query-ad pairs, following the same 6-scale editorial guidelines as [1]. We consider Irrelevant as negative and the rest as positive. We evaluate the performance of the model using two datasets: (1) 42K query-ad pairs for 431 random queries and all their candidate ads after the matching phase (**All-Match**); (2) 11K query-ad pairs for 10.4K random queries that have no textual overlap between query and ad title (**Semantic-Match**).

As baselines, we compare models trained with different combination of features, i.e., (1) 185 text-based features as in [1]; (2) 185 text-based features plus 3 ClickSim [12] features; (3) 185 text-based features plus 3 GloVe [22] features that compute the cosine similarity between the GloVe vector representations for a query

[2]By "q-gram", we mean q-letter-gram that appears in a word, such as 4-grams "lond", "ondo" and "ndon" for word "london". It is different from the query n-gram we present in this paper which refers to the word-level n-grams, such as unigrams "KDD", "2018" and bigram "KDD 2018" for query "KDD 2018".

Table 1: Coverage and NDCG of QueryNGram2Vec for ranking query rewrites.

Model	Coverage	NDCG@1	NDCG@3	NDCG@5
QueryNGram2Vec	**98.15%**	**0.9268**	**0.9615**	**0.9706**
QueryNGram2Vec-Unigram	98.15%	0.9225	0.9600	0.9692
QueryNGram2Vec-Bigram	71.25%	0.8613	0.9371	0.9533
Search2Vec [9]	49.90%	0.8628	0.9292	0.9461
GloVe [22]	96.59%	0.9047	0.9518	0.9633
BM25 [25]	100.00%	0.9071	0.9530	0.9645

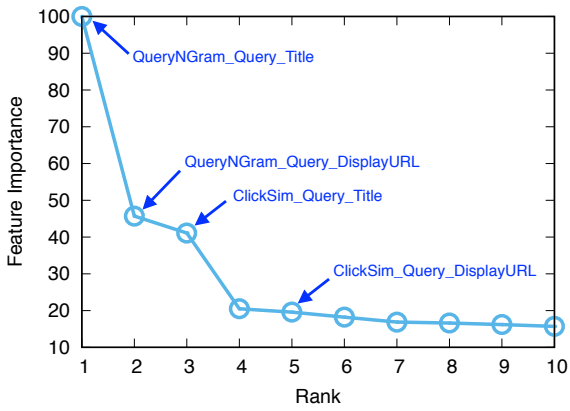

Figure 2: Top-10 feature importances for the model with 185 text-based, 3 ClickSim and 3 QueryNGram2Vec features.

and the three ad components. We also compare using the cosine similarities between QueryNGram2Vec, Clicksim and GloVe query and ad title representations as a proxy of relevance to show the power of the proposed query n-gram embedding to capture the semantics between query and ad.

We use AUC (Area under the ROC curve) to evaluate the ability of different approaches to predict whether an ad is relevant to a query. Table 2 shows that using the cosine similarity between QueryNGram2Vec vectors of query and ad title alone can achieve an AUC improvement of 2.44% and 9.07% versus using Clicksim or GloVe alone on the All-Match dataset. Moreover, compared to the model using 185 text-based and 3 ClickSim features, using QueryNGram2Vec features instead of ClickSim features leads to 1.19% and 2.70% AUC improvements on the All-Match and Semantic-Match datasets, respectively. Finally, we observe that by adding QueryNGram2Vec based additional features to the current model further improves the AUC by 1.80% and 3.96% for the two datasets, respectively. The AUC improvement are more evident for the Semantic-Match dataset, demonstrating that the proposed QueryNGram2Vec model is more effective in capturing semantics in n-grams than ClickSim and GloVe models.

Figure 2 shows the importance scores of the top-10 important features. QueryNGram2Vec features for ad title and display URL are the 2 most important features, followed by ClickSim features for ad title and display URL.[3] This again confirms the importance of having semantic features in a query-ad relevance model, and the power of the proposed query n-gram embedding model.

[3]The corresponding features for ad description do not appear in top-10, mainly because ad description usually significantly overlaps with ad title on text.

4.1.3 Relevance as a Filter for Broad Match: Implementation and Online Bucket Tests. The primary way of applying a query-ad relevance model in sponsored search is to use it as an ad qualityfi lter to remove irrelevant ads from all the candidates after matching phase. In this section, we explain how we apply the relevance model to filter irrelevant ads generated by Broad match offline, focusing on the extraction of QueryNGram2Vec features, as well as the impact of the new relevance model on online bucket tests.

Our system runs dozens of Broad match algorithms to generate query-ad candidates for the queries that were seen in the past on a daily basis. This means billions of query-ad pairs need to be scored by the query-ad relevance model each day. We run a separate pipeline to extract QueryNGram2Vec features for the Broad query-ad pairs, and then apply the relevance model with other features tofi lter ads having relevant score below a given threshold. Since our query n-gram embedding model is trained periodically, once it is refreshed, we run a grid job on Hadoop clusters to extract the QueryNGram2Vec features for all the Broad match query-ad pairs. On the next day, we identify the new queries that appear on that day, and only run the QueryNGram2Vec feature extraction pipeline for the query-ad pairs of the selected queries. This significantly reduces the computation cost as the daily delta is only 0.004% of all the Broad match query-ad pairs.

To evaluate the impact of the proposed QueryNGram2Vec features on the relevance model, we ran an A/B test on the search engine from which we sample our training data with 3% randomly selected users for control and test buckets, and for Desktop and Mobile devices, respectively. The control buckets used the GBDT model with 185 text-based features and 3 ClickSim features, and the test buckets used the GBDT model with 185 text-based features, 3 ClickSim features and 3 QueryNGram2Vec features. Table 3 shows the relative changes of 3 key metrics in the test buckets compared to the control buckets. We observe that by improving the accuracy of the relevancefi lter using the proposed QueryNGram2Vec features, we are able to increase the CTR by 0.26% and 0.47%, and increase the revenue, i.e., RPM (Revenue Per Mille search), by 0.38% and 0.32% for Desktop and Mobile users respectively. Interestingly, we also reduce the ad coverage, i.e., percentage of queries having at least one ad in their search result pages, by 0.35% for Mobile users. Showing fewer ads to users may be considered as an improvement of user experience since users are usually more interested in organic search results.

Figure 3 compares the distribution of ad relevance, according to the 6-scale editorial guidelines, for the same 500 random queries in the control and test buckets for Desktop. We only report the relative change of percentage of each editorial label in test bucket compared to control bucket for confidentiality. We observe 15.58%

Table 2: AUC improvements using QueryNGram2Vec as a feature for query-ad relevance model.

Approach	Setting	All-Match	Semantic-Match
Query and ad title cosine similarity	QueryNGram2Vec	0.8093	0.6908
	ClickSim [12]	0.7900 (-2.44%)	0.6739 (-2.51%)
	GloVe [22]	0.7420 (-9.07%)	0.5840 (-18.29%)
GBDT	185 [1]	0.8293 (-2.94%)	0.6966 (-7.01%)
	185 text-based [1] + 3 ClickSim [12]	0.8537	0.7454
	185 text-based [1] + 3 QueryNGram2Vec	0.8629 (+1.19%)	0.7655 (+2.70%)
	185 text-based [1] + 3 GloVe [22]	0.8294 (-2.93%)	0.7003 (-6.44%)
	185 text-based [1] + 3 ClickSim [12] + 3 QueryNGram2Vec	**0.8691 (+1.80%)**	**0.7749 (+3.96%)**

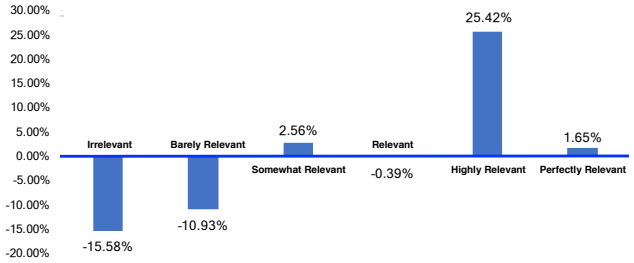

Figure 3: Distribution of query and ad relevance according to 6-scale editorial guidelines for the same 500 random queries in buckets of desktop users. Relative changes of test bucket w.r.t. control bucket are reported.

Table 3: Relative metric changes in test buckets w.r.t. control buckets using relevance model as afi lter for Broad match. The differences reported with * are statistically significant according to a t-test with p-value lower than 0.05.

Device	Ad Coverage	CTR	RPM
Desktop	-0.05%	+0.26%*	+0.38%*
Mobile	-0.35%*	+0.47%*	+0.32%*

decrease of *Irrelevant* ads in test bucket and an increase of 25.42% of *Highly Relevant* ads and 1.65% of *Perfectly Relevant* ads. This further confirms the improvement of our query-ad relevance model using QueryNGram2Vec features.

4.2 Online Broad Matching of Query and Ads

4.2.1Qu eryNGram2Vec as a Broad Match Type. Broad match has been widely used by sponsored search to improve ad coverage. Most Broad match algorithms rely on an offline process to generate query-ad candidates. Different from the traditional query rewriting based approaches, one successful Broad match algorithm used in our system applies the Search2Vec model [9] to learn query and ad embeddings from search sessions and then matches a query to its k most similar ads according to the cosine similarity between their vector representations. However, because the representations are learned for entire queries, this approach can only be applied offline for queries that have been seen in the past. This motivates us to design a new matching algorithm that can match semantically related ads to new and tail queries.

The QueryNGram2Vec model described in Section 3 is a natural candidate to achieve this goal. Since vector representations are learned for query n-grams, once a new query arrives at the system, its vector can be generated, online, by averaging the query n-gram vectors. The resulting query vector is then used in a low latency k-nearest neighbor computation against the set of vectors that are trained for clicked ads (by the same model), and whose campaigns are still active, to retrieve the ad candidates. For ads that do not have vector representations directly trained with the model (because they were either not clicked or new to the system), their vectors can be generated by averaging the vectors of the corresponding n-grams. We apply the QueryNGram2Vec match on queries that have at least two words excepting those having local intent (i.e., queries searching for a location explicitly like “restaurant in London” or implicit like “restaurant near me"). This design choice is simply because we have better strategies to cover such queries in our system.

Before going into details of the system design for the QueryNGram2Vec matcher, wefi rst present results of an offline evaluation to compare its query coverage and ad relevance modeling against the Search2Vec model [9].

To evaluate the query coverage, we randomly sampled 3 million queries (1.67 million unique queries) having at least two words and no local intent from one day that directly follows the date on which the Search2Vec and QueryNGram2Vec models are trained. We joined this set of queries with both models, and observe that the Search2Vec match can only cover 49.59% unique queries that account for 67.76% of the query volume. As expected, the QueryNGram2Vec match significantly improves the coverage to 99.98% for both unique queries and the entire query volume.

To evaluate the relevance of the matched ads, we randomly sampled 11K queries along with the ads appearing on their search result pages. We focus on the query-ad pairs for which both QueryNGram2Vec and Search2Vec models have vector representations for the query and the ad. This choice is deliberate: (1) both algorithms match a query against the same ad collection that is covered by the models; (2) we want to assess whether using a query vector composed from n-grams would degrade the matching quality compared to using a query vector directly learned from the training data. This resulted in 21K query-ad pairs that are labeled according to the 6-scale editorial guidelines [1] for this experiment.

We compare the AUC of the QueryNGram2Vec and Search2Vec matches by considering *Irrelevant* as negative and the rest as positive, and *Irrelevant* and *Barely Relevant* as negative and the rest as positive respectively. We also compare a global NDCG that rank all the query-ad pairs according to their matching scores. We do not use a micro NDCG as on average each query only has two ads in the dataset. Table 4 shows that using QueryNGram2Vec as a match

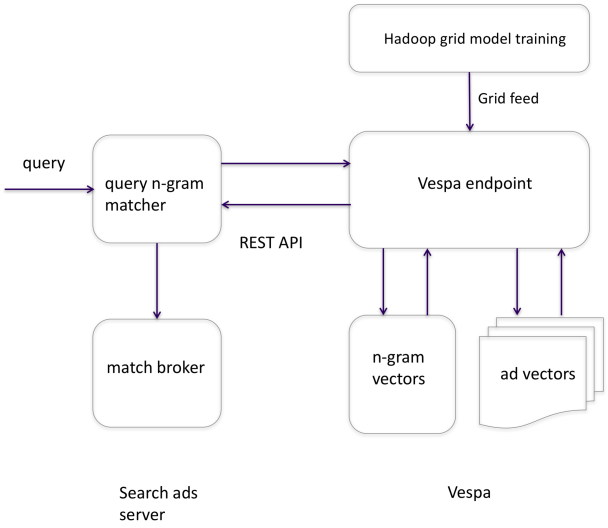

Figure 4: Vespa based query n-grams ad matching system.

type can achieve up to 5.56% better AUC and 1.39% better NDCG than using Search2Vec.

These results indicate that using the proposed query n-gram level modeling approach is beneficial not only for increasing query coverage over of the Search2Vec model, but also for improving the relevance of ads to queries.

4.2.2 OnlineQu ery to Ads Matching: Design and Implementation. An on-line query matching system based on the proposed QueryNGram2Vec model has been implemented in Yahoo sponsored search serving. Figure 4 depicts the architecture of the online QueryNGram2Vec matching system which leverages Vespa [30][4], the recently open sourced machine-learning capable search engine. The system works as follows. New QueryNGram2Vec models are periodically trained on a Hadoop grid and loaded into the Vespa serving nodes, afterfi ltering out inactive ads. The ads server, acting as a Vespa client, sends queries to Vespa via a REST API call. Vespa looks up the query n-gram vectors, averages them, and computes k-nearest neighbor ad vectors under cosine similarity, subject to a threshold, and returns the retrieved ads to the ads server for further processing.

We found that Vespa per node queries-per-second (QPS) and latency can be improved significantly, at little cost to ad retrieval recall, by clustering the ad vectors and using a two phase approximate k nearest neighbor search. In thefi rst phase, k_c nearest neighbor cluster centroids to the query vector are computed and in the second phase, a k nearest neighbor search is computed, but restricted to the clusters of ad vectors corresponding to the k_c nearest centroids found in thefi rst phase. It was found that for a model with 1 million ad vectors, which was the target size for out bucket testing, with 100 clusters and $k_c = 10$, the two phase search yielded over 90 percent recall at top 10, 50 and 100. This resulted in a correspondingly linear increase in QPS/node, from 50 using full search to 500 using the two phase search. Average latency dropped from 69ms to 23ms. We used Spark MLLib's distributed k-means algorithm to compute the clustering.

The system also allows the option to incorporate a Vespa built-in text-similarity metric called nativeRank into the cosine based ad matching via separate thresholds on the cosine similarity, the nativeRank metric, and a linear combination of the two with configurable coefficients.

4.2.3 Online Bucket Test Results. The QueryNGram2Vec Broad matching of query and ads described above was evaluated in an A/B test on buckets that consist of 3% randomly selected users using Mobile devices to access Yahoo Search. The control bucket ran all the existing match types while the test bucket ran QueryNGram2Vec as an additional match type. We used bucket traffic data collected under loose thresholds to optimize the matcher parameters (i.e., coefficients and thresholds) to maximize retained impressions subject to a constraint on CTR. Table 5 summarizes the key metrics of this A/B test confined to Yahoo ads server only while Yahoo Search is partially served by Bing. We observe a statistically significant increase of ad coverage by 2.32% and ad revenue by 2.12%. This is because we are serving more relevant ads through the new semantic Broad match on tail queries and attract 1.78% more clicks on them. Note that the reported increase of the ad coverage does not imply any user experience degradation as the overall (i.e., Yahoo+Bing) ad coverage only increases by 0.15%. The overall revenue increases by 0.96% by using the online QueryNGram2Vec match.

5 CONCLUSIONS

We propose in this paper a novel embedding of queries and click events in sponsored search. The query embeddings are generated from constituent word n-grams in queries and are trained to optimize an event-level word2vec objective over a large volume of search data. We show through a query rewriting task that the proposed query n-gram embedding outperforms the state-of-the-art word embedding models for capturing semantics between two pieces of short texts (e.g., queries). We present two applications of the query n-gram embedding in sponsored search, i.e., as a feature to improve a query-ad relevance model, and as a Broad match type to improve query to ads matching, especially for tail queries. Online A/B tests show that both applications help to significantly improve user experience and search revenue.

ACKNOWLEDGMENTS

The authors would like to thank the editors for evaluating the relevance of query-ad and query-query datasets used in this work. The authors would also like to thank Jo Kristian Bergum, Debajyoti Dutta, Justin Lin, Avishek Saha and Jeff Yuan who greatly contributed to the implementation and deployment of the proposed approaches.

REFERENCES

[1] Luca Maria Aiello, Ioannis Arapakis, Ricardo A. Baeza-Yates, Xiao Bai, Nicola Barbieri, Amin Mantrach, and Fabrizio Silvestri. 2016. The Role of Relevance in Sponsored Search. In *Proceedings of the 25th ACM CIKM (CIKM '16)*. 185–194.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (March 2003), 1137–1155.

[4] https://github.com/vespa-engine/vespa

Table 4: Relevance of query-ad matched by QueryNGram2Vec and Search2Vec.

Match Type	AUC with following labels as negative		NDCG
	Irrelevant	Irrelevant+BarelyRelevant	
Search2Vec	0.6454	0.6303	0.9451
QueryNGram2Vec	0.6813 (+5.56%)	0.6532 (+3.63%)	0.9582 (+1.39%)

Table 5: Relative metric changes in test bucket w.r.t. control bucket adding QueryQGram2Vec as a Broad match type for our ads server. The differences reported with * are statistically significant according to a t-test with p-value lower than 0.05.

Device	Ad Coverage	Click Yield	CTR	RPM
Mobile	+2.32%*	+1.78%*	-0.47%	+2.12%*

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv: 1607.04606* (2016).

[4] Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, Vanja Josifovski, Donald Metzler, Lance Riedel, and Jeffrey Yuan. 2009. Online Expansion of Rare Queries for Sponsored Search. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. 511–520.

[5] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* 41, 6 (1990), 391–407.

[6] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review* 97, 1 (March 2007), 242–259.

[7] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based Translation Models for Web Search: From Word Models to Phrase Models. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. 1139–1148.

[8] Daniel Gayo-Avello. 2009. A Survey on Session Detection Methods in Query Logs and a Proposal for Future Evaluation. *Inf. Sci.* 179, 12 (May 2009), 1822–1843.

[9] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, Ricardo Baeza-Yates, Andrew Feng, Erik Ordentlich, Lee Yang, and Gavin Owens. 2016. Scalable Semantic Matching of Queries to Ads in Sponsored Search Advertising. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 375–384.

[10] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. 383–392.

[11] Dustin Hillard, Stefan Schroedl, Eren Manavoglu, Hema Raghavan, and Chirs Leggetter. 2010. Improving Ad Relevance in Sponsored Search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. 361–370.

[12] Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly, Jr., Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. 185–194.

[13] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating Query Substitutions. In *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*. 387–396.

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv: 1607.01759* (2016).

[15] Rémi Lebret and Ronan Collobert. 2015. "The Sum of Its Parts": Joint Learning of Word and Phrase Representations with Autoencoders. *arXiv preprint arXiv:1506.05703* (2015).

[16] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers* 28, 2 (1996), 203–208.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. 3111–3119.

[19] Mark Dredze Mo Yu. 2015. Learning Composition Models for Phrase Embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 227–242.

[20] Erik Ordentlich, Lee Yang, Andy Feng, Peter Cnudde, Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, and Gavin Owens. 2016. Network-Efficient Distributed Word2Vec Training System for Large Vocabularies. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 1139–1148.

[21] Xiaochang Peng and Daniel Gildea. 2016. Exploring phrase-compositionality in skip-gram models. *arXiv preprint arXiv: 1607.06208* (07 2016).

[22] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*. 1532–1543.

[23] Filip Radlinski, Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. 2008. Optimizing Relevance and Revenue in Ad Search: A Query Substitution Approach. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. 403–410.

[24] Hema Raghavan and Rukmini Iyer. 2010. Probabilistic First Pass Retrieval for Search Advertising: From Theory to Practice. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. 1019–1028.

[25] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3.. In *TREC*, Donna K. Harman (Ed.), Vol. Special Publication 500-225. 109–126.

[26] Douglas L. T. Rohde, Laura M. Gonnerman, and David C. Plaut. 2006. An improved model of semantic similarity based on lexical co-occurence. *COMMUNICATIONS OF THE ACM* 8 (2006), 627–633.

[27] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of Research. MIT Press, Chapter Learning Representations by Back-propagating Errors, 696–699.

[28] Fabrizio Sebastiani. 2002. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.* 34, 1 (2002), 1–47.

[29] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*. 384–394.

[30] Vespa. 2017. http://docs.vespa.ai/documentation/overview.html. Open source. (2017).