Joint Embedding of Query and Ad by Leveraging Implicit Feedback

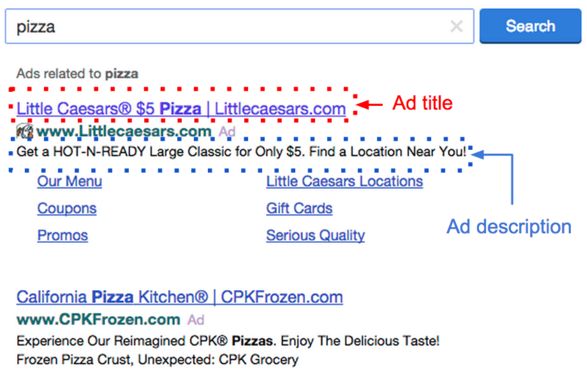
Sungjin Lee and Yifan Hu
Yahoo Labs
229 West 43rd Street, New York, NY 10036, USA
`{junion, yifanhu}@yahoo-inc.com`

Abstract

Sponsored search is at the center of a multibillion dollar market established by search technology. Accurate ad click prediction is a key component for this market to function since the pricing mechanism heavily relies on the estimation of click probabilities. Lexical features derived from the text of both the query and ads play a significant role, complementing features based on historical click information. The purpose of this paper is to explore the use of word embedding techniques to generate effective text features that can capture not only lexical similarity between query and ads but also the latent user intents. We identify several potential weaknesses of the plain application of conventional word embedding methodologies for ad click prediction. These observations motivated us to propose a set of novel joint word embedding methods by leveraging implicit click feedback. We verify the effectiveness of these new word embedding models by adding features derived from the new models to the click prediction system of a commercial search engine. Our evaluation results clearly demonstrate the effectiveness of the proposed methods. To the best of our knowledge this work is the first successful application of word embedding techniques for the task of click prediction in sponsored search.

1 Introduction

Sponsored search is a multibillion dollar market (Easley and Kleinberg, 2010) that makes most search engine revenue and is one of the most successful ways for advertisers to reach their intended audiences. When search engines deliver results to a user, sponsored advertisement impressions (ads) are shown alongside the organic search results (Figure 1). Typically the advertiser pays the search engine based on the *pay-per-click* model. In this model the advertiser pays only if the impression that accompanies the search results is clicked. The price is usually set by a *generalized second-price* (GSP) auction (Edelman et al., 2005) that encourages advertisers to bid truthfully. An advertiser wins if the expected revenue for this advertiser, which is the bid



Ads related to **pizza**

Little Caesars® $5 **Pizza** | Littlecaesars.com
www.Littlecaesars.com Ad
Get a HOT-N-READY Large Classic for Only $5. Find a Location Near You!
Our Menu · Little Caesars Locations
Coupons · Gift Cards
Promos · Serious Quality

California **Pizza** Kitchen® | CPKFrozen.com
www.CPKFrozen.com Ad
Experience Our Reimagined CPK® **Pizzas**. Enjoy The Delicious Taste!
Frozen Pizza Crust, Unexpected: CPK Grocery

Figure 1: Sponsored ads when "pizza" was searched at Yahoo! (*www.yahoo.com*).

price times the expected click probability (also know as *click through rate*, or CTR), is ranked the highest. The price the advertiser pays, known as *cost-per-click* (CPC), is the bid price for the second ranked advertiser times the ratio of the expected CTR between the second and first ranked advertisers. From this discussion it should be clear that CTR plays a key role in deciding both the ranking and the pricing of the ads. Therefore it is very important to predict CTR accurately.

The state of the art search engine typically uses a machine learning model to predict CTR by exploiting various features that have been found useful in practice. These include historical click performance features such as historical click probability for the query, the ad, the user, and a combination of these; contextual features such as temporal and geographical information; and text-based features such as query keywords or ad title and description. Among these, historical click performance features often have the most predictive power for queries, ads and users that have registered many impressions. For queries, ads and users that have not registered many impressions, however, historical CTR may have too high a variance to be useful. Hillard *et al.* (2011) observed that the number of impressions and clicks recorded on query-ad pairs have a very long tail: only 61% of queries has greater than three clicks. They also reported a drastic drop in the accuracy of the click prediction model when fewer historical observations are available. Furthermore, fine-grained historical CTR information takes a huge amount of space, which makes it costly to maintain. On the other hand, text features are always readily available, and thus are particularly useful for those cases for which there is insufficient historical information.

Multiple researchers, for example (Richardson, 2007; Cheng and Cantú-Paz, 2010), reported the usage of text features including simple lexical similarity scores between the query and ads, word or phrase overlaps and the number of overlapping words and characters. Such features rely on the assumption that query-ad overlap is correlated with perceived relevance. While this is true to a certain extent, the use of simple lexical similarity cannot capture semantic information such as synonyms, entities of the same type and strong relationships between entities (e.g. CEO-company, brand-model, part-of). Recently a host of studies on word embedding have been conducted; all map words into a vector space such that semantically relevant words are placed near each other in the space (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014). The use of continuous word vectors has been shown to be helpful for a wide range of NLP tasks by better capturing both syntactic and semantic information than simple lexical features (Socher et al., 2012a).

No previous research on sponsored search has successfully used word embeddings to generate text features. In this paper, we explore the use of word embeddings for click prediction. However, it is clear that conventional word embeddings (which solely rely on word co-occurrence in a context window) can only offer limited discriminative power because queries and ad text are typically very short. In addition, conventional word embeddings cannot capture user intents, preferences and desires. Wang et al. (2013) showed that specific frequently occurring lexical patterns, e.g., *x% off*, *guaranteed return in x days* and *official site*, are effective in triggering users desires, and thus lead to significant differences in CTR. Conventional word embeddings cannot capture these phenomena since they do not incorporate the implicit feedback users provide through clicks and non-clicks. These observations naturally lead us to leverage click feedback to infuse users' intentions and desires into the vector space.

The simplest way to harness click feedback is to train conventional word embedding models on a corpus that only includes clicked impressions, where each "sentence" is constructed by mixing the query and ad text. Having trained a word embedding model, we simply take the average of word vectors of the query and ads respectively to obtain sentence (or paragraph) vectors, which in turn are used to compute the similarity scores between the query and ads. Our experiments show that this method does improve click prediction performance. However, this method has several potential weaknesses. First, the use of only clicked impressions ignores the large amount of negative signals contained in the non-clicked ad impressions. Second, the use of indirect signals (word co-occurrences) can be noisy or even harmful to our ultimate goal (accurate click prediction) when it is combined with direct signals (impressions with click feedback). Third, without explicit consideration about the averaging step in the training process of word embedding models, a simple averaging scheme across word vectors may be a suboptimal. We therefore propose several joint word embedding models; all of these aim to put query vectors close to relevant ad vectors by explicitly utilizing both positive and negative click feedback. We evaluate all these models against a large sponsored search data set from a commercial search engine, and demonstrate that our proposed models significantly improve click prediction performance.

The rest of this paper is organized as follows. In Section 2 we present a brief summary of related work. In Section 3 we give some background information on ad click prediction in sponsored search. In Section 4 we describe our methods. In Section 5 we discuss our experiments. We finish with some conclusions and future directions in Section 6.

2 Related Work

Text features for predicting click probability There have been many studies on the use of text features for click prediction. For example, Dembczynski et al. (2008) used a decision rule-based approach involving such lexical features as the number of words in ad title and description, the number of segments and length of the ad URL, and individual words and terms in ad title and description. Cheng et al. (2010) used a logistic regression model that used both historical click performance features and simple lexical features such as word or phrase overlap between query and ad title and description. Trofimov et al. (2012) used a variant of boosted decision trees with similar features. Richardson et al. (2007) specifically considered new ads (which lack historical click prediction data) and proposed to use the CTR for ad terms, the frequency of certain unigrams (e.g., dollar signs) and general English usage patterns, and simple lexical distance between the query and ads. In all this previous work, text features consisted only of surface-level text features. To the best of our knowledge, there is no previous work adopting semantic-level text features for the purpose of click prediction, in particular word embeddings to measure query-ad relevance. In a similar vein of research, Grbovic et al. (2015) adopted word embeddings to the task of query rewriting for a better match between queries and keywords that advertisers entered into an auction. Using the embeddings, semantically similar queries are mapped into vectors close in the embedding space, which allows expansion of a query via K-nearest neighbor search.

Word embeddings for language processing Recently many NLP systems have obtained improved performance with less human engineering by adopting distributed word representations (Socher et al., 2012a).

In particular, neural word embedding techniques are now known to be effective in capturing syntactic and semantic relationships, and more computationally efficient than many other competitors (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014). On top of word embeddings, a new stream of research on sentence and paragraph embeddings has also emerged to tackle higher level tasks such as sentiment analysis, machine translation and semantic relatedness tasks. *Recursive Neural Networks* (RNNs) have been used for syntactic parsing (Socher et al., 2012b; Socher et al., 2013; Socher et al., 2014; Irsoy and Cardie, 2014). *Long Short-Term Memory* (LSTM) networks have been applied to machine translation (Bahdanau et al., 2014; Sutskever et al., 2014) and semantic processing (Tai et al., 2015). Interestingly *Convolutional Neural Networks* (CNNs), widely used for image processing, have recently emerged as a strong class of models for NLP tasks (Kim, 2014; Blunsom et al., 2014). As apposed to the models above, *PhraseVector* (Le and Mikolov, 2014) takes a less structured but unsupervised approach by treating a piece of text as a token and performing word embedding-like training with an unlimited context window. None of this previous work exactly fits the click prediction task. Since queries and ads are much less structured than usual text, it is not attractive to use models with complex structures, such as RNNs, at the cost of speed and scalability. *PhraseVector* is less structured but it does not support compositionality, suffering from sparseness or requiring to train new vectors for each unseen query and ad. Interestingly, as reported in (Tai et al., 2015), a simple averaging scheme (*mean vector*) was found to be very competitive to more complex models for high level semantic tasks despite its simplicity. These observations lead us to one of our models that aims to improve the *mean vector* method by directly optimizing mean vectors instead of word vectors.

Joint embedding to bridge multiple views. Multiple studies have explored the task of bringing multiple views into the same vector space. For example, there is now a large body of research on joint modeling of text and image information (Frome et al., 2013; Karpathy and Fei-Fei, 2014; Socher et al., 2014). The multimodal embedding space helps find appropriate alignments between image regions and corresponding pieces of text description. Joint embedding has also been applied to question answering (Wang et al., 2014) and semantic understanding (Yang et al., 2014). In contrast to the tasks above, there is no natural component-wise correspondence between queries and ads; instead the relationship is more implicit and pragmatic. Because of this, our methods rely on global rather than component-level signals for model training.

3 Baseline Click Prediction Model

We first present a high-level description of sponsored search. The process consists of several stages. First, given a user query, a list of candidate ads are retrieved, either by exactly matching query terms to the bid terms of the advertiser, or by first using query term expansion to obtain a longer list of matched ads. Some candidate ads may be filtered out based on metrics such as ad quality. Then, a *click prediction model* scores the candidate ads to estimate how likely it is that each will be clicked. This *click probability* serves a crucial role both in the user experience and in the revenue for the search engine. The ads with the highest click probabilities are placed in the search results page. The price-per-click for each ad shown is determined based on the click probabilities and the GSP auction.

Our baseline click prediction model is formulated as a supervised learning problem. Specifically we use *Logistic Regression* (LR) since LR is well suited for probability estimation. Given a variety of features, the probability of a click is expressed as:

$$p(c|q,a,u) = \frac{1}{1 + exp\{\sum_i w_i f_i(q,a,u)\}}, \quad (1)$$

where $c \in \{1, 0\}$ is the label (1: click or 0: non-click), $f_i(q,a,u)$ is the ith feature derived for query-ad-user triple (q,a,u) and w_i is the associated weight. The model is trained using a stochastic gradient descent algorithm on a per impression basis with l_1 regularization to avoid overfitting.

An accurate LR model relies greatly on the effectiveness of its features. Our baseline model is furnished with a rich set of features that are typically used in commercial search engines. The first feature type is based on the historical CTR of user, query, ad triples (if there is enough historical information on this). We use two groups of features of this type: COEC based features and user factor features. The second feature type is based on query and ad text.

Due to the significant decrease of CTR depending on the ad position, it has become common practice to use position-normalized CTR (a.k.a. *Clicks Over Expected Clicks*):

$$COEC = \frac{\sum_p c_p}{\sum_p i_p * CTR_p}, \quad (2)$$

where the numerator is the total number of clicks received by the configuration of interest; the denominator is the *expected clicks* (ECs) that an average ad would receive after i_p times impressions at position p, and CTR_p is the average CTR at position p, calculated over all queries, ads and users. We use user-independent features derived from COEC statistics for specific query-ad pairs. However, many impressions are needed for these statistics to be reliable and therefore data for specific query-ad pairs can be sparse and noisy (only around 70% of queries and about 50% of query-URL, query-bid term pairs have historical CTR). To alleviate this problem, additional COEC statistics over aggregations of queries or ads are also used. The exact description of these aggregations is beyond the scope of this paper, but briefly we exploit the categorization of the ads in ad groups, campaigns, and accounts defined by advertisers.

Since it is well known that personalization features are crucial to obtain accurate click prediction models (Cheng and Cantú-Paz, 2010), we also use features that measure user factors relating to CTR. The user click feedback features capture the inclination of individual users to click on ads in general. The user-query click feedback features indicate the propensity of users to click for certain queries or groups of queries. Finally user-ad features dictate the user preferences on certain ads or advertisers. However the data sparseness problem becomes even more serious when it comes to user-specific features (we use a threshold of 100 for statistical confidence). For example, only about 5% of user-URL pairs and 1% of user, query, URL triples have historical CTR information. Therefore a set of segment-level features can be extracted as back-off features where users, queries and ads are clustered into groups, and group level historical CTRs are collected.

Our second major feature type involves the lexical similarity between query and ad text. These text features assume that users are more likely to click on ads that seem to be relevant to the query, and that perceived relevance is correlated with the degree of query-ad overlap. These features include the number of overlapping words and characters in query-ad URL, query-ad title, and query-ad description, and the number of words and characters in the query. The discrimination power of simple lexical features is relatively limited because query and ad text are typically very short.

Finally, we use other contextual features that are helpful in predicting the click probabilities.; for example, time of day, day of week, and geographic information. To model interactions among features, some features are selected by domain knowledge to be conjoined. All together, our baseline model utilize a comprehensive set of historical CTR, lexical, and contextual features (over a hundred features in total). The fact that this baseline model is highly optimized makes the subsequent performance improvement from our proposed algorithm meaningful. This baseline model is used in production in part of a major search platform.

4 Joint Embedding for Click Prediction

We now describe several methods that jointly embed words in both the query and the ads into the same vector space. In our experiments, we incorporate these methods as features in our click prediction model. We start by defining the notation used in this section. A sponsored search dataset $\mathcal{D}$ is a set of tuples for each ad impression (q, t, d, y) where $q \equiv \{q_j\}$ is a multiword query string, $t \equiv \{t_k\}$, $d \equiv \{d_l\}$ are multiword ad title and description strings, and y is a binary indicator for whether the ad is clicked. We have two choices in defining the vocabulary $\mathcal{V}$ from which words are drawn: we can use a unified vocabulary for both query and ads or define a separate vocabulary for each – $\mathcal{V} \equiv \mathcal{V}_q \oplus \mathcal{V}_a$. In our initial experiments the unified vocabulary constantly yielded better performances, thus we always use the unified vocabulary here. We use bold letters $\mathbf{q}_j, \mathbf{t}_k, \mathbf{d}_l$ to denote the corresponding embedding representations of $\{q_j, t_k, d_l\}$. Finally we use W to represent the vocabulary matrix; in W each column is a word vector.

4.1 Exploiting word2vec embedding

Typically word embeddings are learned from a given text corpus through implicit supervision of predicting the current word given a window of its surrounding text or predicting each word in the window of the current word. The former approach is known as *continuous bag-of-words* (CBOW) and the latter *Skip-gram*. For simplicity's sake we use negative-sampling for training word embedding models (Mikolov et al., 2013b). More formally we define the binary conditional probability for a pair of words (v, w): [1]

$$p(\mathbf{v}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{v}^T \mathbf{w})}, \tag{3}$$

The CBOW algorithm learns word embeddings by minimizing the following *logloss* of each impression i with regard to W:

$$\begin{aligned} CB_i(W) = &- \log p(\mathbf{w}_i, \boldsymbol{\mu}_C) \\ &- \sum_{v \in N(w_i)} \log \left(1 - p(\mathbf{v}, \boldsymbol{\mu}_C)\right), \end{aligned} \tag{4}$$

where the context C of a word w_i comes from a window of size k around the word in a sentence of n words $w_1, \ldots, w_n$: $C = w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}$. $\boldsymbol{\mu}_C$ is the averaged context vector of w_i; $\boldsymbol{\mu}_C = \frac{1}{|C|} \sum_{v \in C} \mathbf{v}$. $N(w_i)$ is the set of negative examples which is drawn according to the unigram distribution of the corpus raised to the 3/4th power. Similarly to (Mikolov et al., 2013b), we adopt a dynamic window size – for each word the actual window size is sampled uniformly from $1, \ldots, k$.

Similarly the Skip-gram algorithm minimizes the *logloss* of each impression i with regard to W:

$$\begin{aligned} SK_i(W) = &- \sum_{v \in C} \log p(\mathbf{w}_i, \mathbf{v}) \\ &- \sum_{v \in N(w_i)} \log \left(1 - p(\mathbf{w}_i, \mathbf{v})\right). \end{aligned} \tag{5}$$

In our first word embedding model that incorporates click feedback, we construct a corpus by taking only clicked impressions from $\mathcal{D}$ and then mixing (q, t, d) of each impression into a sentence. Then we simply train CBOW and Skip-gram models on the corpus.

[1] We use only a single vector for a word unlike (Mikolov et al., 2013b) where two vectors ("input" and "output") for a word are used. This halves the required space to store vectors without performance loss.

4.2 Joint word embedding using click feedback

Although the CBOW and Skip-gram models trained on a specially constructed corpus can capture signals from both click feedback and word co-occurrence, they have a couple of drawbacks. First, by ingesting only clicked impressions we "waste" the large amount of negative signals contained in the non-clicked impressions. Second, the incorporation of indirect signals such as word co-occurrences can be rather harmful for achieving accurate click prediction as these are very noisy compared to direct signals such as click feedback.

For our second word embedding model that incorporates click feedback, we define a joint word embedding model that minimizes the following *weighted logloss*:

$$JW_i(W) = \eta(y_i)\Big\{l(y_i, q_i, t_i) + l(y_i, q_i, d_i)\Big\}, \quad (6)$$

where the component loss function $l(\cdot,\cdot,\cdot)$ is defined as follows:

$$\begin{aligned} l(y, a, b) = \sum_k^{|a|} \sum_l^{|b|} & - y \log p(\mathbf{a}_k, \mathbf{b}_l) \\ & - (1-y) \log\left(1 - p(\mathbf{a}_k, \mathbf{b}_l)\right). \end{aligned} \quad (7)$$

Here $\eta(y_i)$ is a function that returns a small weight η only to negative examples:

$$\eta(y_i) = \begin{cases} \eta, & \text{if } y_i = 0, \\ 1, & \text{otherwise.} \end{cases} \quad (8)$$

The fact that the user did not click on an ad does not necessarily mean that the ad is not what the user wanted; it often means that the ad is just less favored than the clicked ads (Rendle et al., 2009). Since the scope of this work is restricted to estimating click probability on a per-impression basis, we adopt a weighting scheme rather than optimizing rank-based loss over a set of related impressions.

4.3 Joint mean vector optimization

The joint word embedding models defined in the previous sections do not define how to aggregate a variable length sequence of word vectors into a sentence (or paragraph) vector to facilitate the computation of sentence-level similarity scores. One approach to this aggregation task is *mean vector*: simply average the word-level embeddings across the sentence or paragraph. As noted by (Tai et al., 2015), this approach is a strong competitor to more complex models such as RNNs or LSTMs despite its simple composition method. However, this method may generate suboptimal sentence vectors. With *weight logloss*, we aim to optimize sentence vectors instead of individual word vectors:

$$JM_i(W) = \eta(y_i)\Big\{\bar{l}(y_i, q_i, t_i) + \bar{l}(y_i, q_i, d_i)\Big\}, \quad (9)$$

where the component loss function $\bar{l}(\cdot,\cdot,\cdot)$ is defined as follows:

$$\begin{aligned} \bar{l}(y, a, b) = & - y \log p(\boldsymbol{\mu}_a, \boldsymbol{\mu}_b) \\ & - (1-y) \log\left(1 - p(\boldsymbol{\mu}_a, \boldsymbol{\mu}_b)\right), \end{aligned} \quad (10)$$

where $\boldsymbol{\mu}_s$ returns the average vector for the multiword string s, i.e. $\boldsymbol{\mu}_s = \frac{1}{|s|}\sum_k^{|s|} \mathbf{s}_k$.

5 Experiments

Data The data used in our experiments were collected from a random bucket of the Yahoo! sponsored search traffic logs for a period of 4 weeks in October 2014. In the data there are approximately 65 million unique users, 150 million unique queries and 12 million unique ads. There are approximately 985 million ad impression events in total. We split the data into 3 partitions with respect to time: the first 14 days' data are used for training word embedding models, the next 7 days' data for training click prediction models, and the last 7 days' data for testing. Table 1 presents more detailed statistics for the data.

Models We are interested in evaluating the usefulness of different word embedding models as features for click prediction, and already have a very good baseline system for this task. Consequently, in all experiments below, we used the same personalized historical CTRs, contextual and lexical features as the baseline system described in Section 3. We tested models with the following additional features derived from word embeddings:

1. cosine similarity between the mean vectors of the query and ad title
2. cosine similarity between the mean vectors of the query and ad description
3. sum of 1 and 2
4. sigmoid function value for the dot product of the mean vectors of the query and ad title
5. sigmoid function value for the dot product of the query and ad description
6. sum of 4 and 5

All these continuous features are quantized into 50 bins. We compared five different word embedding algorithms:

1. Skip-gram trained on Wikipedia (SK-WIKI)
2. Skip-gram trained on clicked impressions (SK-CI), see Section 4.1
3. CBOW trained on clicked impressions (CB-CI), see Section 4.1
4. Joint individual word vector embedding (JIWV), see Section 4.2

	Embedding	Train	Test	Total
Number of impressions	489M	252M	244M	985M
Number of unique queries	82M	46M	45M	150M
Number of unique ads	8M	6M	6M	12M
Number of unique users	40M	25M	25M	65M
Number of unique words in query	6757K	6039K	5774K	16028K
Number of unique words in ad title	3371K	2586K	2557K	4598K
Number of unique words in ad description	1993K	1583K	1589K	2628K
Number of words in query	1600M	830M	800M	3230M
Number of words in ad title	3260M	1690M	1635M	6585M
Number of words in ad description	4311M	2206M	2131M	8650M
Average number of words in query per impression	3.275	3.284	3.289	3.281
Average number of words in ad title per impression	6.667	6.708	6.706	6.687
Average number of words in ad description per impression	8.818	8.759	8.741	8.784

Table 1: Data description

5. Joint mean vector embedding (JMV), see Section 4.3

We used the skip-gram mode of word2vec [2] with a window size of 5 and negative sampling to train the SK-WIKI model. For all other models we used in-house implementation which employs *AdagradRDA* (Duchi et al., 2011) to minimize the loss functions introduced in Section 4. The dimension of a word vector is set to 100 for all algorithms[3]. We removed a set of prefixed stop-words and all words occurring fewer than 100 times; the resulting vocabulary comprised 126K unique words. To process our web-scale data, we implemented a multithread program where each thread randomly traverses over a partition of the data $\mathcal{D}$ to compute gradients and update the matrix W stored in a shared memory. For computational efficiency, the hogwild lock-free approach (Recht et al., 2011) is used. We set η in Eq. 8 and Eq. 9 to 0.2 through grid search based on two-fold cross-validation. This small value indeed verifies the idea of weak negative feedback for unclicked impressions.

In order to circumvent the severe influence of ad position on the click prediction model, only the impressions that are placed at the top position were used for training the click prediction model. Note that CTR at other positions can be derived from that of the ad at the top position through scaling. Dembczynski et al. (2008) showed that CTR can be decomposed as a product of the probability of an ad getting clicked given its being seen and the probability of an ad being seen at a particular position.

Results We ordered query-ad pairs by the predicted score to compute *AucLoss* (i.e. 1 - AUC where AUC is the area under the *Receiver Operating Characteristic* (ROC) curve). The ROC AUC is known to have a correlation with the quality of ranking by the predicted score (Fawcett, 2006); thus is one of the most important metrics for click prediction (McMahan et al., 2013).

	AucLoss Reduction (%)
Baseline + SK-WIKI	0.530
Baseline + SK-CI	0.595
Baseline + CB-CI	0.656
Baseline + JIWV	2.276
Baseline + JMV	**4.114**

Table 2: Comparative evaluation results in AucLoss reduction from the baseline system

Our experimental results (Table 2) show that the use of features derived from our proposed word embedding models significantly reduce AucLoss by up to 4.1%. For commercial search engines which have a very strong baseline *AucLoss*, a reduction of 1% can be considered large (McMahan et al., 2013). Moreover, as expected, the more issues (as identified in Section 4) an algorithm addresses, the better performance it achieves. From the comparison between the SK-WIKI model and the rest, we can recognize the importance of word embedding models specialized to domain text and supervision signals. Also the difference between the CB-CI (SK-CI) model and the JIWV model indicates the noisiness of indirect signals such as word co-occurrence compared to direct signals like click feedback. Finally the gap between the JIWV and JWV models highlights the significance of considering compositionality in the word embedding training process.

Eyeballing the most similar words to several queries in the vector space is often helpful for getting some sense about how different methods influence the resulting vector spaces. Table 3 lists the top 20 most similar words to the query "metal watch." The top words for the SK-WIKI model are not semantically interesting in terms of capturing the intent or desires. This clearly shows the limitation of word embedding methods that only rely on indirect signals (i.e. word co-occurrences) from a generic text corpus (e.g. Wikipedia) for sponsored search click prediction. Noticeably the methods

[2]Available at https://code.google.com/p/word2vec/

[3]Higher vector dimensions such as 200 were also tried but did not give a significant improvement.

SK-WIKI	CB-CI	JIWV	JMV
watch (0.733)	metal (0.718)	wwhl (0.711)	tacticalwatch.com (0.701)
grind (0.687)	previews (0.652)	cbs (0.704)	watchrepairsusa.com (0.695)
grease (0.682)	yidio.com (0.648)	station (0.668)	omegas (0.689)
kites (0.676)	episodes (0.633)	putlocker.com (0.662)	watchco.com (0.682)
hammer (0.675)	steel (0.626)	iwatch (0.659)	wach (0.670)
spinning (0.672)	whatch (0.615)	kidizoom (0.658)	station (0.656)
flashing (0.671)	bobcometal.com (0.586)	freesports360.com (0.658)	shockwarehouse.com (0.628)
trash (0.670)	ridiculousness (0.582)	tedtalks (0.655)	freesports360.com (0.618)
flame (0.669)	www.abc.com (0.574)	11/10c (0.650)	akribos (0.616)
flaming (0.665)	a&e (0.573)	movie2k (0.640)	18mm (0.616)
cigar (0.664)	utube (0.570)	nfl.com/now (0.637)	narutoget.com (0.614)
home-made (0.663)	instantly (0.565)	criminalsgonewilddvd (0.631)	watchstation.com (0.611)
glow (0.662)	khoobsurat (0.562)	espnnfllive.com (0.631)	authenticwatches.com (0.610)
bouncing (0.662)	outnumbered (0.561)	foxsports1 (0.623)	interrupted (0.603)
filler (0.662)	itv (0.559)	viooz (0.623)	criminalsgonewilddvd (0.601)
smoke (0.660)	premiere (0.556)	bubble (0.623)	whatch (0.600)
shoot (0.658)	films (0.555)	wewood (0.621)	$109.99 (0.597)
scoop (0.653)	stainless (0.554)	westclox (0.617)	tirebuyer.com (0.596)
noises (0.652)	fabricators (0.550)	potlocker (0.613)	skagen.com (0.588)
rocking (0.651)	fabrication (0.550)	nickelodeon (0.613)	wewood (0.584)

Table 3: Top 20 most similar words to “metal watch”

SK-WIKI	CB-CI	JIWV	JMV
costumes (0.762)	princess (0.833)	costumes (0.831)	new-costumes.com (0.815)
bride (0.723)	costumed (0.814)	wonder (0.784)	coustume (0.808)
princesses (0.722)	customes (0.808)	sweetiegames.com (0.782)	namefully.com (0.800)
serene (0.708)	costume (0.806)	cleopatra (0.753)	$35.90 (0.789)
highness (0.676)	costomes (0.806)	new-costumes.com (0.752)	2-days (0.781)
bess (0.674)	costimes (0.803)	girls.simple (0.749)	$36.90 (0.776)
princess]] (0.671)	m.buycostumes.com (0.800)	werewolf (0.749)	costume (0.766)
attire (0.670)	custumes (0.796)	yoshi (0.747)	$28.90 (0.764)
princess (0.662)	officialprincesscostumes.com (0.792)	leia (0.747)	princess (0.758)
dresses (0.662)	custume (0.789)	merida (0.742)	cistumes (0.756)
highness (0.658)	coustome (0.789)	$49.90 (0.735)	spider-woman (0.756)
jewels (0.652)	cosyumes (0.787)	low-budget (0.727)	the-wristband-factory.com (0.750)
wedding (0.651)	coustume (0.786)	babies (0.727)	$17.90 (0.750)
robes (0.648)	coustums (0.786)	fembot (0.726)	sugarsmascotcostumesċom (0.748)
prince (0.644)	buycostumes.com (0.785)	costums (0.722)	cotumes (0.748)
clothes (0.644)	codtume (0.784)	$3.90 (0.721)	coneheads (0.747)
dancing (0.644)	leia (0.784)	hermione (0.718)	$23.90 (0.739)
sophie (0.640)	coustumes (0.784)	supergirl (0.715)	$7.90 (0.739)
consorts (0.639)	costums (0.783)	toothless (0.713)	costomes (0.738)
glamorous (0.639)	coatumes (0.782)	starlord (0.709)	$19.90 (0.737)

Table 4: Top 20 most similar words to “princess costumes”

that incorporate click feedback find more words related to products, services or websites instead of just conceptuatlly related words. Given that real products or services can be regarded as the best possible surrogates to user intents and desires, this demonstrates the effectiveness of our methods. This tendency gets stronger as a method takes into account both positive and negative click feedback.

Another very interesting observation comes from the fact that none of the methods except JMV successfully captures the composite meaning of “metal watch”; they tend to either find related words separately for each query word (e.g. “watch” is strongly associated to the sense of watching something like movie or other types of video) or find totally unrelated words (particularly SK-WIKI). This demonstrates that it is crucial to address compositionality in the very process of learning word vectors.

Table 4 shows the top 20 most similar words to the query “princess costumes.” In this example we can spot another surprising result. The JMV model pushes a lot of price related expressions to the top[4]. This may imply that many parents search for lower cost costumes, clearly showing a clear psychological desire in the financial dimension. This observation confirms the findings in (Wang et al., 2013) about the significant role of certain ad expressions in triggering users’ psychological desires. We also note that the CB-CI model returns a lot of misspells for “costume(s)”, which would not be possible with simple lexical features of the baseline system. A close look at this example generally confirms the observations we made for the previous ex-

[4]We have not tried any normalization for numbers but it might be worth doing given the important role they play.

SK-WIKI	CB-CI	JIWV	JMV
kids (0.724)	game (0.629)	kids (0.795)	program (0.764)
pac-man (0.708)	gams (0.615)	kidsfootlocker.com (0.788)	scorekeepers (0.757)
games (0.703)	games (0.613)	flight: (0.754)	gamingchairs.hayneedle.com (0.754)
pinball (0.687)	petrasplanet.com (0.601)	edit: (0.744)	teepees (0.752)
boy (0.680)	for (0.590)	raz (0.737)	nn2 (0.750)
adventure (0.664)	gamse (0.584)	program (0.733)	game (0.749)
roleplaying (0.664)	ganes (0.574)	abercrombiekids.com (0.729)	girls (0.746)
quests (0.658)	collectors (0.574)	sumdog (0.721)	cranium (0.746)
d&d (0.658)	awsome (0.551)	take20 (0.702)	ggg (0.738)
genie (0.655)	kid (0.550)	freegamesdownload.com (0.699)	pac-man (0.732)
solitaire (0.654)	game.com (0.537)	program; (0.695)	kidsfootlocker.com (0.730)
gamers (0.653)	g2u.com (0.536)	gromsocial.com (0.694)	equestriagirls.hasbro.com (0.721)
games=== (0.651)	bouncing (0.533)	softschools (0.693)	kids (0.718)
game; (0.649)	for.kids (0.532)	gapkids (0.691)	mahjong (0.718)
consoles]] (0.645)	catching (0.528)	downnload (0.682)	tvo (0.717)
in-game (0.645)	gamesfreak (0.519)	edheads (0.681)	sumdog (0.701)
rpg (0.644)	minecraft (0.516)	ruum.com (0.674)	cpm.wargaming.net (0.699)
wargames (0.644)	firetruck (0.514)	gamestop (0.672)	osgood-schlatter (0.698)
game]] (0.643)	games: (0.513)	a&f (0.668)	gamestop (0.697)
fast-paced (0.641)	todlers (0.512)	pac-man (0.663)	$4.01 (0.695)

Table 5: Top 20 most similar words to "game for kids"

ample. Finally Table 5 shows top the 20 most similar words for the query "game for kids." Once again we found the same analysis holds for this case.

6 Conclusions

In this paper we explored the use of word embedding techniques to overcome the shortcomings of traditional lexical features for ad click prediction in sponsored search. We identified several potential weaknesses of the plain application of conventional word embedding methodologies: the lack of the right machinery to harness both positive and negative click feedback, the limited utility of pure word co-occurrence signals, and no consideration of vector composition in the word embedding training process. We proposed a set of new implicit feedback-based joint word embedding methods to address those issues. We evaluated the new word embedding methods in the context of a very good baseline click prediction system, on a large scale data set collected from Yahoo! search engine logs. Our experimental results clearly demonstrate the effectiveness of the proposed methods. We also presented several examples for qualitative analysis to advance our understanding on how each algorithm really contributes to the improved performance. To the best of our knowledge this work is the first successful application of word embedding techniques for the sponsored search task.

There are multiple interesting research directions for future work. One of these directions is to extend the vocabulary by identifying significant phrases (as well as words) before training word vectors. Hillard et al. (2011) employed *Conditional Random Fields* to divide queries with multiple words into segments and collected historical CTR on the segment level. We also like to investigate more structured embedding methods such as RNNs (probably for ad descriptions). In case the computational cost of such methods are too high to be practical for sponsored search, we can employ them only for a small fraction of ads filtered by faster methods.

It may be possible to deal with the implicit negative feedback of unclicked ad impressions in a more principled way by adopting ranking-based loss functions. However, this is only possible with the extra cost of identifying and aggregating related ads into a single transaction.

Though not directly related to NLP, yet another promising direction is to jointly embed not only text data but also a variety of user activities (e.g., organic search results, mobile app usages, other daily activities) all together in the same vector space. Since many of the different sources contain their own unique information, we might be able to obtain a much better understanding about the user state and intent through this rich joint embedding space. Joint embedding with rich information can also help us to perform automatic clustering of users, eventually leading to powerful smoothing methods for personalized historical CTR statistics.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Haibin Cheng and Erick Cantú-Paz. 2010. Personalized click prediction in sponsored search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 351–360, New York, NY, USA. ACM.

K. Dembczynski, W.Kotlowski, and D.Weiss. 2008. Predicting ads click-through rate with decision rules. In *Proceedings of WWW 08*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

David Easley and Jon Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press.

Benjamin Edelman, Michael Ostrovsky, Michael Schwarz, Thank Drew Fudenberg, Louis Kaplow, Robin Lee, Paul Milgrom, Muriel Niederle, and Ariel Pakes. 2005. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129.

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, pages 383–392, New York, NY, USA. ACM.

Dustin Hillard, Eren Manavoglu, Hema Raghavan, Chris Leggetter, Erick Cantú-Paz, and Rukmini Iyer. 2011. The sum of its parts: reducing sparsity in click estimation with query segments. *Inf. Retr.*, 14(3):315–336.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Andrej Karpathy and Li Fei-Fei. 2014. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

H. Brendan McMahan, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, Jeremy Kubica, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, and Eugene Davydov. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 1222, New York, New York, USA, August. ACM Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 693–701.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Matthew Richardson. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *In Proceedings of the 16th International World Wide Web Conference (WWW-07*, pages 521–530. ACM Press.

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012a. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, ACL '12, pages 5–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012b. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. 2012. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, ADKDD '12, pages 2:1–2:6, New York, NY, USA. ACM.

Taifeng Wang, Jiang Bian, Shusen Liu, Yuyu Zhang, and Tie-Yan Liu. 2013. Psychological advertising: exploring user psychology for click prediction in sponsored search. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–571. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650.