Deeply Supervised Semantic Model for Click-Through Rate Prediction in Sponsored Search

Jelena Gligorijevic*
Temple University
Philadelphia, Pennsylvania, US
jelena.stojanovic@temple.edu

Djordje Gligorijevic*
Temple University
Philadelphia, Pennsylvania, US
gligorijevic@temple.edu

Ivan Stojkovic
Temple University
Philadelphia, Pennsylvania, US
ivan.stojkovic@temple.edu

Xiao Bai
Yahoo! Research
Sunnyvale, California, US
xbai@oath.com

Amit Goyal
Criteo
Palo Alto, California, US
a.goyal@criteo.com

Zoran Obradovic
Temple University
Philadelphia, Pennsylvania, US
zoran.obradovic@temple.edu

ABSTRACT

In sponsored search it is critical to match ads that are relevant to a query and to accurately predict their likelihood of being clicked. Commercial search engines typically use machine learning models for both query-ad relevance matching and click-through-rate (CTR) prediction. However, matching models are based on the similarity between a query and an ad, ignoring the fact that a retrieved ad may not attract clicks, while click models rely on click history, being of limited use for new queries and ads. We propose a deeply supervised architecture that jointly learns the semantic embeddings of a query and an ad as well as their corresponding CTR. We also propose a novel cohort negative sampling technique for learning implicit negative signals. We trained the proposed architecture using one billion query-ad pairs from a major commercial web search engine. This architecture improves the best-performing baseline deep neural architectures by 2% of AUC for CTR prediction and by statistically significant 0.5% of NDCG for query-ad matching.

CCS CONCEPTS

• **Information systems** → **Sponsored search advertising**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Deep Learning, CTR Prediction, Query to Ad Matching

1 INTRODUCTION

Sponsored search has been a major monetization model for commercial web search engines, contributing a significant portion to the multi-billion dollar industry of online advertising. Given a query, it is critical for search engines to retrieve relevant ads and to accurately predict their CTR in order to maximize the expected revenue while ensuring good user experience. Both overpredicting and underpredicting CTR would result in revenue loss.

Machine learning models made great success in predicting CTR for sponsored search. Most of the models adopted in the industry rely on a large set of well-designed features to predict CTR. Features extracted from click history have been proved very effective [5]. However, models that heavily rely on click features often fail to generalize to new queries and new ads with insufficient history [27]. To make predictions in such cases, models resort to syntactic or semantic features extracted from queries, ads, and advertisers [21, 27]. Deep neural networks were also proposed to learn features from traditional models [17] or to learn CTR from existing features [36]. In spite of the existing success, designing and selecting appropriate features remains a very challenging problem for CTR prediction [14].

Following the progress of deep learning in natural language processing, recent efforts rely on deep neural networks to capture semantic similarities between queries and ads to predict CTR without any feature engineering [7]. Such models are learned end-to-end from clicks without explicit supervision for capturing the semantic similarity between a query and an ad, and as we show in this work, they have not achieved their full potential in CTR prediction.

A number of recent works [11, 16] used deep neural networks to model the semantic similarity between a query and an ad. These models were shown effective in a query to ad relevance matching. However, as they do not directly model clicks, retrieved ads are only weakly correlated to the ads presented to users based on expected revenue (which highly depends on the predicted CTR).

In this work, we propose a deeply supervised end-to-end architecture for CTR prediction in sponsored search. This architecture jointly learns CTR and discriminative representations of queries and ads such that clicked query-ad pairs are also mapped closer in the embedded space. Specifically, this architecture takes the texts of a query and an ad as input to bi-directional recurrent neural networks (bi-RNNs) and attention networks to learn discriminative distributed embeddings. Query and ad embeddings are then matched together and fed into convolutional neural networks (CNNs) to predict CTR. Two losses, specific to semantic matching and CTR prediction, are jointly optimized at different levels of the architecture to provide a deep supervision for both tasks. This architecture has the advantages of *(i)* not relying on any feature engineering; *(ii)* directly optimizing CTR prediction; *(iii)* directly learning semantic representations to enable query-ad matchings more correlated with clicks and expected revenue. The key contributions are:

*Co-first author

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- We propose a novel deep architecture that jointly learns CTR and discriminative representations of queries and ads. To the best of our knowledge, this is the first attempt to simultaneously learn CTR and semantic embeddings using click data. By optimizing two logistic losses specific to CTR prediction and semantic matching instead of using only one CTR specific logistic loss, we were able to achieve statistically significant lift in AUC.
- We propose a novel cohort negative sampling technique that naturally draws information from implicit negative signals in the data. We assess the impact of this technique in terms of performance and prove the convergence of our method through theoretical analysis.
- We conduct an extensive empirical evaluation of the proposed architecture using about one billion query-ad samples from the Yahoo! web search engine. Comparison with state-of-the-art CTR prediction models shows that our model improves the AUC of the best-performing baseline model by 2%.
- We evaluate the quality of the query and ad embeddings learned by our model through a query-ad matching task using a large-scale editorially labeled dataset. Comparison with state-of-the-art matching models shows that our model improves the NDCG of the best-performing baseline by statistically significant 0.5%, confirming its ability to learn meaningful semantic embedding.

2 RELATED WORK

We first present problems and challenges in sponsored search and review most recent advances in deep learning approaches. Subsequently, we review other relevant advances in deep learning, which have previously been applied only on tasks different than ours.

2.1 Related Work in Sponsored Search

The frequently tackled problems of improving the sponsored search include CTR prediction, query rewriting and query to ad matching.

A large body of work focused on predicting probability that an ad would be clicked, if shown as a response to a submitted query [10, 14, 22]. State-of-the-art approaches have mainly used handcrafted features of ad impressions obtained from historical impressions (i.e. ad and query CTR's, users' historical features, etc.) and semantic similarities of queries and ads [27]. These approaches range from Bayesian [10] to feature selection approaches [14], however, a common challenge for all is creating and maintaining a large number of sparse contextual and semantic features [22].

Focusing on the broad matching of queries and ads that have similar semantic meaning is another line of research [8]. The task is to retrieve ads that are semantically similar to the query [11] without exactly matching keywords (i.e. query "running machine" and ad "elliptical trainer"). This task has been commonly addressed by query rewriting models [18] or by semantic matching [8, 11, 15].

More recently, many approaches for CTR prediction utilize various deep learning techniques. Deep learning primarily alleviates issues of creating and maintaining handcrafted features by learning them automatically from the "raw" query and ad text data.

It is common to learn query and ad semantics from ad impressions for a given query with click information. In [15] authors proposed a deep structured semantic model (DSSM) with dual architecture that embedded a query on the one side and an ad on the other and learned matching between the two given the click information. In order to improve quality of the learned semantic match and capturing query intent, a word attention mechanism was successfully used for the query and ad representations [34].

Some of the approaches are defined as a CTR prediction task rather than as a matching task. In [31], features of an impression (query text, ad text, ad landing page, campaign ID, keywords, etc.) are learned automatically from the impression, in a deep architecture, to predict click probability. Other models, DeepMatch [7] and MatchTensor [16] proposed very deep dual network architectures for query and ad embeddings with a matching layer to learn ad impression representations useful for CTR prediction.

Both groups of approaches, learning semantics of queries and ads and learning to predict CTR are widely used in systems for serving ads. However, they pose a trade-off, while semantic learning learns relations between queries and ads, it has no direct click probability notion, CTR prediction models, on the other hand, may suffer from not capturing the semantics of queries and ads implicitly thus affecting their prediction quality. The approach we propose in this study is a well-rounded framework for ad systems capable of both learning quality semantics of queries and ads as well as being able to accurately predict click probability.

The two mentioned approaches, DeepMatch and MatchTensor have shown great results in practice and will, thus, be the main baselines and building blocks for the model proposed in this study. The two approaches are conceptually very similar as both learn independent representations of a query and an ad, and use a matching layer to associate their words, and finally learn to predict CTR. However, the difference between them is in way they learn representations of words, i.e. DeepMatch primarily uses temporal convolutional layers, while MatchTensor uses bi-RNNs. Also, they propose slightly different matching layers, DeepMatch proposes a cross-feature matrix, while MatchTensor proposes cross-feature tensor. As both models perform exceptionally well, we present a detailed analysis of performance of both models experimentally in Section 4.

The model proposed in this study further extends on the advances described above by addressing their shortcomings by introducing novel ways of learning semantically rich representations. As such, the proposed model demonstrates the state-of-the-art results on both CTR prediction and query2ad matching tasks, traditionally modeled by different families of models. This is achieved by means of (i) learning new blocks in the deep architectures to improve modeling capacity, (ii) adding deep supervision to improve quality of learned representations deep in the model and (iii) learning parameters in an efficient and information-rich way to capture more of the available semantics in the dataset.

2.2 Related Work in Deep Learning

Many approaches for mathematical characterization of language, that model sequence data, were proposed to advance the field of natural language processing. Initially, distributed low-dimensional representations of words were introduced in [29] and recently successfully applied for learning semantic and syntactic relations among words [23]. The idea of using distributed representations of words was further exploited in approaches as RNNs, capable of learning an embedded high-dimensional representation of sequences.

Recurrent Neural Networks. RNNs are a popular family of models for sequential problems. While previous approaches have often modeled word sequence as an order-oblivious sum, RNNs learn representations of word sequences by maintaining internal states, which are updated sequentially and are used as a proxy for predicting the target. The ability to stack multiple layers allows building deeper representations that result in great improvements on many tasks. In particular, an architecture of RNNs called long short-term memory (LSTM) cell achieved the biggest success [13].

Attention Network Models. Attention models dynamically reweight the importance of various elements (words, phrases or characters) in the text during the decoding process, thus altering the learned representation. Use of attention demonstrated considerable improvements in performance [2]. An attention mechanism was developed as a separate neural network that takes a sequence of word embeddings and learns attention scores for each word, where more "important" words in the document have higher attention leading to a more focused higher-order representation of the sequence. Attention models were recently adapted for the general setting of learning compact representations of documents [34].

bi-RNNs. Another successful paradigm is the bi-RNN, where two RNNs (i.e. LSTM, thus bi-LSTM) independently encode the text sequence in both forward and in backward direction [30] computing representation that captures complex relations between words in the text. Final sentence representation is obtained by aggregating representations of the two single-directional LSTMs, and it was observed that bi-LSTM's perform well on datasets where there is no strict order in the sequences, such as the case with Web queries.

Convolutional Text Models. Recently, architectures for sequence modeling increasingly include temporal convolutions as building blocks. Temporal convolutions are capable of learning representations of sequences which proved as a good building block for several deep architectures. Good examples being ConvNet for text classification [35] and the Very Deep CNN (VDCNN) model [6], both of which use temporal convolutions to model a sequence of words/characters with aim to perform classification. These models successfully outperformed RNN based models. In this study, we use word-level VDCNN as one of the baselines, as it consists of equivalent blocks as the DeepMatch model, save the matching layer.

Deeply supervised models. Recently, several models drew benefits from utilizing deep supervision [20, 32, 37]. The key idea is to use supervision at various layers across the model to enforce discriminativeness of the features [20] and potentially resolve exploding/vanishing gradients [32, 37]. However, existing approaches mostly use the same predictive task in deeper layers as in the final layer [20, 32] and in some cases use reconstruction loss [36]. We build upon these advances proposing a novel approach of using deep supervision specifically designed to extract information from the data in an explicit way, which would not be possible otherwise.

Learning from implicit negative signals. This has for a long time been a challenging task for domains with implicit negative signals. Recently, search2vec model for learning with implicit negative signals from sponsored search sessions was proposed [12] with improved performance and speed of the algorithm. Furthermore, [3] have confirmed this approach and applied it on the special case of bipartite graphs. We exploit implicit negatives in our model and consider comparing to search2vec algorithm in Section 4.2.

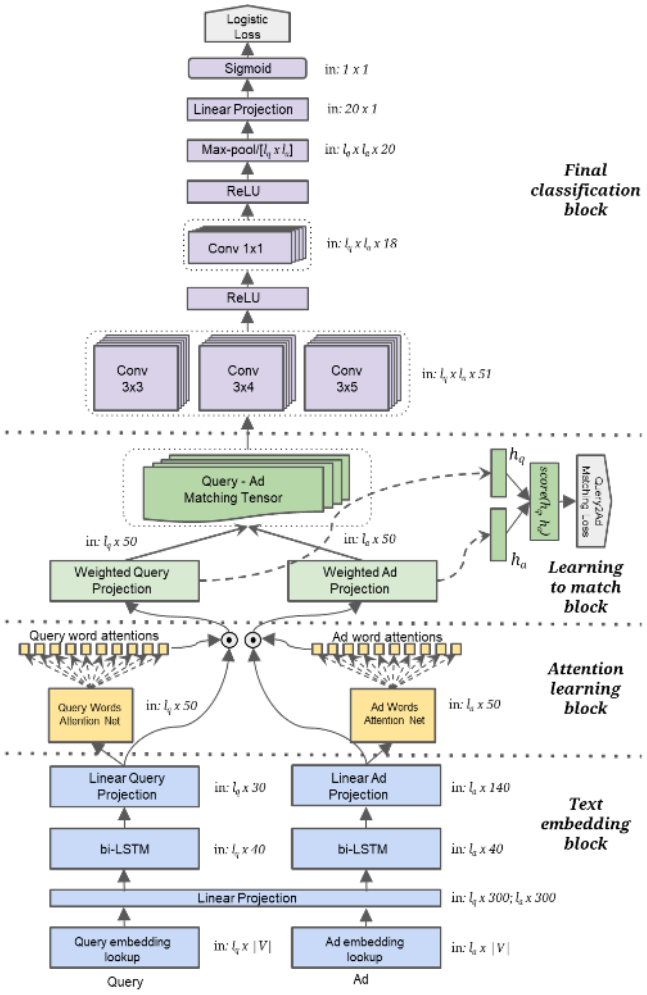

Figure 1: Proposed DSM model block diagram

3 PROPOSED MODEL

Graphical representation of the proposed model, which we call the Deeply Supervised Matching (DSM) model is given in Figure 1.

The model takes query text and ad text as inputs, and it learns their separate embeddings through a series of layers, including bi-direction LSTM and attention layers. Learned embeddings are then used in two-fold matching: (1) embeddings of query and ad words are used in an elementwise product to construct a matching tensor, and (2) matching of dense representations of query and ad is learned using a novel matching loss designed for sponsored search. Learned matching tensor is then passed through series of convolutional and pooling blocks to learn CTR prediction.

3.1 Blocks of the proposed model

3.1.1 Query and Ad text embedding. Embeddings of query and ad texts are done in two networks. First l_q words in the query and l_a words in the ads are embedded into a $d_{qa}^{(1)} = 300$ dimensional space. Then, a fully connected layer is used to learn linear combinations of words in a $d_{qa}^{(2)} = 40$ dimensional space. These two

layers share weights for both queries and ads. Embeddings of query and ad are passed to the respective bi-LSTM layers such that the model learns complex relations between words, which is in particular important for queries that may have a different order of words but the same meaning (i.e. "best restaurants in Boston" vs. "Boston best restaurants"). Due to different lengths of query and ad text embedding sizes are now $d_q^{(3)} = 30$ and $d_a^{(3)} = 140$, as suggested in the literature [16, 34]. Finally, fully connected layers are used to reduce representations of all words in the same, reduced, dimensional space $d_{qa}^{(4)} = 50$, resulting in representations $v_q = l_q \times d_{qa}^{(4)}$ and $v_a = l_a \times d_{qa}^{(4)}$, for query and ad, respectively.

3.1.2 Attention learning. In order to learn rich representations of queries and ads, it is imperative to focus on words that carry the most information. In order to learn representations that focus on important parts of queries and ads we employ the attention models from machine translation and adapt them to a more general case of using word scores for learning compact (vector) representations [34]. Two attention blocks are used, one for query text and one for ad text. These blocks yield word scores, that signify attentions the model will give to different words. Both attention models are implemented as two-layered individual neural networks $s_q(v_q; \theta_q)$ and $s_a(v_a; \theta_a)$ with softmax at their final layer

$$t_q^{(i)} = \frac{exp(s_q(v_q^{(i)}; \theta_q))}{\sum_{i=1}^{l_n} exp(s_q(v_q^{(i)}; \theta_q))}. \quad (1)$$

Neural networks $s_q(v_q^{(i)}; \theta_q)$ and $s_a(v_a^{(i)}; \theta_a)$ learn real valued scores for each i^{th} word in a given query and ad, respectively. Attentions learning in DSM is coupled with the entire network (end-to-end).

Attentions $t_q^{(i)}$ for a query word, and $t_a^{(i)}$ for an ad word, are then used to re-weight their input representations v_q and v_a to obtain compact representations of query and ad used for learning to match as $h_q = \sum_i t_q^{(i)} * v_q^{(i)}$ and $h_a = \sum_i t_a^{(i)} * v_a^{(i)}$. There are other ways of obtaining compact representations h_q and h_a, such as sum, average or max of individual word vectors. However, our experiments, as well as available literature [34], demonstrate that such strategies are inferior to using attention.

3.1.3 Query and Ad matching. Many models for sponsored search advertising have either the capability to learn good quality semantic representations of queries and ads, or the capability to perform CTR prediction well without explicitly modeling semantics, thus (over-)specializing in only one of the tasks. To address this, we have two matching processes in our framework.

First, similarly to MatchTensor [16], we build a tensor for implicitly matching words in a query and an ad. l_q words in a query and l_a words in an ad, with $d_{qa}^{(4)}$-dimensional embeddings, are matched in a cross product tensor of shape $l_q \times l_a \times d_{qa}^{(4)}$. Each word in a query will be matched to each word in an ad, and the element-wise product of their vectors will be a thread in the matching tensor. Finally, an exact-match $l_q \times l_a$ slice is added to the tensor, with all zeros except for words that co-occur in a query and an ad, where we put ones. This slice serves as a bias and yields slight improvement as opposed to the model that does not use exact matches [16].

Second, we propose explicit matching to capture semantic similarity between a query and an ad. We propose a way to match the vectors h_q and h_a, where we aim to embed them such that they are closer in the embedded space if there was a click and farther away if there was no click, similarly to [11]. To achieve this, we optimize scores between h_q and h_a vectors, where scores are posed as an inner product of the vectors. To avoid introducing the computational complexity of negative sampling, we introduce a cohort negative sampling approach to optimize the matching function. The detailed description, as well as convergence analysis of the proposed optimization strategy, are given in Sections 3.2.1 and 3.2.2.

Benefits of using multiple learning tasks for the same model have recently been recognized [20]. Deep models benefit from enforcing the middle layers to be discriminative, which is beneficial for the final predictive task, as discriminative classifiers trained on highly discriminative features will perform better than a discriminative classifier trained on less discriminative features. In our case, representations of query and ad should be close for semantically similar pairs and distant for dissimilar ones. Such representations benefit the classification task as the semantic relations have been well captured deep in the model. Due to adding such deep supervision, our model is named the Deeply Supervised Matching (DSM) model.

3.1.4 Learning to predict from matched representation. The matching tensor from the previous block is then convolved through the entire depth $d_{qa}^{(4)} + 1$ by three convolutional blocks with different filter sizes: 3 for query words; and 3, 4, and 5 words for ad filters. The number of filters is fixed to 6 for the first set of convolution blocks and 20 for the final convolutional layer. Complex representations between a query and ad words are learned here, and they are passed through the ReLU layer, after which another 1 convolution with ReLU was used before the two-dimensional max-pool layer that embeds the whole query-ad impression in a single vector. Finally, the vector is fed to a fully connected layer and passed through a sigmoid layer $\sigma(\cdot)$ to obtain the logits of the model.

3.2 Logistic and Matching Losses

Finally, to optimize the parameters of DSM, we have logistic loss $\mathcal{P}$ for the CTR prediction based on logits from the topmost layer:

$$\mathcal{P}(W) = -\frac{1}{N}\sum_{n=1}^{N}(y_n log(\hat{y}_n) + (1 - y_n) log(1 - \hat{y}_n)), \quad (2)$$

where $\hat{y}_n$ are obtained logits after final sigmoid layers and y_n is click label for the n^{th} ad impression. The matching loss $\mathcal{Q}$ for query and ad vectors, as a negative sampling approximation, can be generalized as a composition of positive and negative pairs [24]:

$$\begin{aligned}\mathcal{Q}(W) &= \sum_{b=1}^{B}\Big(\sum_{j\in\mathcal{D}_p^{(b)}} \mathcal{Q}^+(W) + \sum_{k\in\mathcal{D}_n^{(b)}} \mathcal{Q}^-(W)\Big) \\ &= \sum_{b=1}^{B}\Big(\sum_{j\in\mathcal{D}_p^{(b)}} -\log\sigma(h_q^{(j)T} h_a^{(j)}) + \sum_{k\in\mathcal{D}_n^{(b)}} \log\sigma(-h_q^{(k)T} h_a^{(k)})\Big),\end{aligned} \quad (3)$$

where B is the total number of batches, while $\mathcal{D}_p$ and $\mathcal{D}_n$ are positive and negative impressions within each batch, respectively. In our implementation, we use a variant of the negative sampling

loss for learning to match query and ad vectors, called cohort[1] negative sampling. As will be discussed later in the paper, this loss differs from the negative sampling loss proposed in [24], as negative samples are used within the cohort but not sampled ad-hoc, thus saving computational time.
The final loss function becomes the sum of Eq. 2 and Eq. 3

$$\mathcal{L}(W) = \mathcal{P}(W) + \mathcal{Q}(W). \tag{4}$$

We use W to annotate the set of all parameters in the DSM.

Based on the Lemma 1 in [20], a good solution for $\mathcal{Q}$ is also a good solution for $\mathcal{P}$. However, conversely is not necessarily true. This clearly states that features learned for $\mathcal{P}$ may not be optimal for $\mathcal{Q}$. In the case of our application, features learned for the classification task may not capture semantic similarities between queries and ads that may carry considerable amounts of information. Another interesting aspect of using multiple optimization functions is that it is reasonable to assume that $\mathcal{L}$ and $\mathcal{P}$ share the same optimum [20], while $\mathcal{Q}$ can be observed as a regularizer.

Therefore, it is important to notice that $\mathcal{Q}$ is not used for learning to match explicitly, but as stated before, to enforce discriminative embeddings of the lower layers such that final logits reflect semantic information found in the data. To demonstrate this, we used the DSM model for query to ad matching and compared it to well-established models for the task in Section 4.2.

Weights are initialized by a truncated normal initializer. To optimize $\mathcal{L}$, we use Adam [19] with a decaying gradient step.

3.2.1 Cohort Negative Sampling for Matching Loss. The nature of ad serving in sponsored advertising is that for each query, the publisher (search engine in this case) can provide a set of ads on different positions on the search result page. The most impactful position is called "north" (ads placed above organic links) and it yields the largest click-through rate for ads [4]. Up to five ads can be presented at this location (n1 – n5), and users may or may not click on any of them. Click/No-click information provides an implicit information on a query and ad relevance that we can learn from. Thus, to learn matching we need to focus on a group of query–ad pairs that were served to the user for a given search, and we can pull several such searches in the cohort we use for training. Such data allows us to learn a semantic match of a query and an ad implicitly, based on users' feedback. In the past, learning such implicit relations between queries and ads has shown great benefit in sponsored search ad recommendations [11], while its computational benefits were supported in [3]. In this study, unlike in [3], implicit negative samples naturally occur as signals from the users, furthermore they do not consider that complete ground-truth bipartite graph is needed to obtain the good working model, as artificial negative samples can be harmful if a pair is semantically related. The later issue is leveraged with matching tensor layer, while matching loss merely plays a role of discriminativeness enforcing regularizer. An example of a cohort of users' search query impressions used for training our models is given in Figure 2.

Traditionally, techniques such as negative sampling [24] were proposed as a speedup for costly partition functions while learning to match. However, in negative sampling for each positive sample (in our case query-ad (q, a) pair with click) m there needs to be k sampled ads from some distribution P_n that provide negative pairs for a given query, thus ending up with a total of $m + m * k$ embedding operations prior to matching. In our case, we do not sample k negative ads, thus the computation is decreased by $m * k$ in addition to capturing implicit signals from users.

In cohorts with insufficient negative pairs for the partition function to provide satisfactory approximation of true distribution, we resort to negative cross-referencing queries with ads that are found in cohort and were not served for those queries (dotted gray links in the Figure 2), obtaining up to $< m * (m-1)$ negative pairs.

For further analysis, it is useful to characterize the matching loss function in terms of expected values over query q, and ad a pairs as positive (click) and negative (no click) examples drawn from their respective distributions P_d and P_n:

$$\begin{aligned}\mathcal{Q}(W) &= \mathbb{E}_{(q,a)\sim P_p(q,a)}[\mathcal{Q}^+(q,a;W)] + \mathbb{E}_{(q,\grave{a})\sim P_n(q,\grave{a})}[\mathcal{Q}^-(q,\grave{a};W)] \\ &= \mathbb{E}_{q\sim P_p(q)}\left[\mathbb{E}_{a\sim P_p(a|q)}\mathcal{Q}^+(q,a;W)] + \mathbb{E}_{\grave{a}\sim P_n(\grave{a})}[\mathcal{Q}^-(q,\grave{a};W)]\right],\end{aligned} \tag{5}$$

Due to the nature of the data, we assume that the query is observed first and then ads are provided (conditioned on the query). Obviously, we do not assume independence between two random variables. However, for sampling negative samples $(q, \grave{a})$, we assume that the distribution $P_n(q, \grave{a}) = P_p(q)P_p(\grave{a})$ is approximately scaled by $\frac{P_n(\grave{a})}{P_p(\grave{a})}$ to allow the factorization $P_n(q, \grave{a}) = P_p(q)P_n(\grave{a})$.

It is important to formalize the sampling strategy $\mathbb{E}_B[\mathcal{Q}_B(W^t)]$ that is different from the simple i.i.d. sampling:

$$\begin{aligned}&\mathbb{E}_B[\mathcal{Q}_B(W^t)] = \\ &\mathbb{E}_{(q,a)\sim P_p(q,a)}[\mathcal{Q}^+(q,a;W)] + \mathbb{E}_{(q,a)\sim P_p(q)P_p(\grave{a})}[\frac{P_n(\grave{a})}{P_p(\grave{a})}\mathcal{Q}^-(q,\grave{a};W)] \\ &+ \frac{1}{m(m-1)}\sum_{j=1}^{m(m-1)}\mathbb{E}_{(q,a)\sim P_p(q,\grave{a})}[\frac{P_n(\grave{a})}{P_p(\grave{a})}\mathcal{Q}^-(q,\grave{a};W)] = \\ &\mathbb{E}_{q\sim P_p(q)}[\mathbb{E}_{a\sim P_p(a|q)}[\mathcal{Q}^+(q,a;W)] + \mathbb{E}_{\grave{a}\sim P_n(\grave{a})}[\mathcal{Q}^-(q,\grave{a};W)] \\ &+ \mathbb{E}_{\grave{a}\sim P_p(\grave{a})}[\frac{P_n(\grave{a})}{P_p(\grave{a})}\mathcal{Q}^-(q,\grave{a};W)]] = \\ &\mathbb{E}_{q\sim P_p(q)}\left[\mathbb{E}_{a\sim P_p(a|q)}[\mathcal{Q}^+(q,a;W)] + 2\mathbb{E}_{\grave{a}\sim P_n(\grave{a})}[\mathcal{Q}^-(q,\grave{a};W)]\right] = \\ &\mathcal{Q}(W^t).\end{aligned} \tag{6}$$

We can define samples in the cohort as taking positive and negative samples from the $P_p(q, a)$ and $P_n(q, \grave{a})$ distributions, respectively. As shown in Figure 2, the first expectation is for positive, clicked pairs, the second is for ads not clicked, and the third is for negative query–ad pairs created within the cohort. Due to properties of expectation and joint distribution, we are allowed to factorize the expectation to obtain the result equivalent to Eq. 5. Using the gradient property of expectation, we obtain the following lemma:

Lemma 3.1 (Cohort negative sampling is an unbiased stochastic gradient estimator). *Given samples in B generated by the cohort negative sampling algorithm, the stochastic gradient is*

[1]We use word cohort to disambiguate our sampling strategy from the traditional mini-batch i.i.d. sampling.

unbiased as the expected cohort gradient equals the true gradient: $\mathbb{E}_B[\nabla \mathcal{L}_B(W^t)] \equiv \nabla \mathcal{L}(W^t)$

3.2.2 Stochastic Gradient Descent View. To optimize our objective we use the stochastic gradient descent (SGD) algorithm. The update at the t^{th} iteration of SGD is in the form:

$$W^{t+1} = W^t - \eta_t \nabla \mathcal{L}_t(W). \quad (7)$$

For examples chosen randomly in iteration t, SGD provides an unbiased estimate of the gradient: $\mathbb{E}_B[\nabla \mathcal{L}_B(W^t)] \equiv \nabla \mathcal{L}(W^t)$.

Analyzing the convergence of the SGD algorithm for non-convex problems has been a big research question. However, it was shown in [9, 26] that SGD follows a local convergence bound. Furthermore, [9] provide the following theorem showing that SGD will converge within T steps, based on the assumptions that $\nabla \mathcal{L}(W^t)$ is an unbiased estimator and that the expected variance of the gradient $\nabla \mathcal{L}(W^t)$ is upper-bounded by σ^2.

Theorem 3.2 (Local convergence of non-convex SGD [9, 26]). *Suppose* $\mathcal{L}$ *has a* σ*-bounded gradient; let* $\eta_t = \eta = c\sqrt{T}$ *where* $c = \sqrt{\frac{2(\mathcal{L}(W^0) - \mathcal{L}(W^*))}{L\sigma^2}}$*, and* W^* *is an optimal solution to (4). Then, the iterates of SGD satisfy*

$$\min_{0<t<T-1} \mathbb{E}[\| \nabla \mathcal{L}(W^t) \|^2] \leq \sqrt{\frac{2(\mathcal{L}(W^0) - \mathcal{L}(W^*))L}{T}}\sigma. \quad (8)$$

Although $\mathcal{L}$ is a composite of $\mathcal{P}$ and $\mathcal{Q}$, the theorem still applies according to Lemma 2 from [20]. Thus, we only need to show that the proposed sampling strategy in Section 3.2.1 yields an unbiased minimizer $\mathcal{Q}$, which follows from the Lemma 3.1. Satisfied assumptions conclude the convergence of our algorithm.

4 EXPERIMENTS

We conducted an extensive empirical evaluation for the CTR prediction task on a large dataset (about one billion query-ad samples) from a major commercial search engine (Section 4.1). We also evaluate the quality of the query and ad embeddings learned by our model through a query-ad matching task using a large-scale editorial labeled dataset (Section 4.2). The data and the experimental set-up used for both tasks are described in each of these sections.

4.1 CTR prediction

For the CTR prediction task, the aim is to estimate, as accurately as possible, the probability $P(click|ad, query)$ that a user would click on an ad displayed after submitting a query.

4.1.1 Click-through rate data. To train and test the proposed model and baselines for this task, we collected a random sample of logged query-ad pairs served by a popular commercial search engine. The sample comprises of 987,734,146 query-ad pairs for training and 16,881,864 for testing, containing only advertisements placed at the top (north) of search result page (ads that are served above organic search links). The data consists of a query text on one side, and ad title, ad description and ad display URL on the other side. The query and ad texts are processed and normalized using an in-house tool to remove special characters and punctuations, make letters lower case, fix common typos, split URLs, etc. All example pairs are accompanied with information whether the ad was clicked or not, which we use as supervised information to train all models.

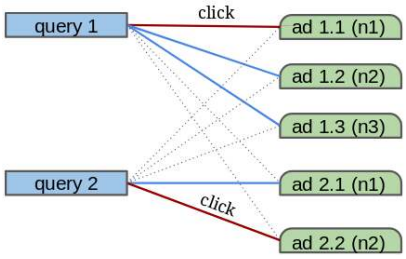

Figure 2: Cohort negative sampling (an example with queries and served ads in the position "north", n1 up to n5) Red links are ad clicks, blue links are ads displayed but not clicked, and negative pairs we create by coupling queries and ads that were not displayed for that ad - dotted links.

To better characterize the dataset, we comment on its distribution of the queries. A majority (75%) of queries are infrequent (tail queries), i.e. appearing less than five times overall, and if measured in the test set only there are more than 90% them. As discussed before, this is a major limitation of most of the traditional CTR prediction models, and given the volume of the tail queries, this reaffirms the necessity for predictive models that can generalize when insufficient or no click history is available. For a subset of queries that are seen often (appear more than 20 times, called head queries) we expect all the models to perform better, even though they make up only about 3% of the training set and less than 1% of the testing dataset.

4.1.2 Baselines. We compare our proposed Deeply Supervised Matching (DSM) approach against several alternatives described in Section 2.2: A linear logistic regression learned on top of the word embedding layer (LM), Very Deep CNN (VDCNN) [6], DeepMatch (DM) [7], and MatchTensor (MT) [16].

All deep learning models were trained in two ways: (i) with the use of pre-trained word embedding vectors (obtained from [25]); and (ii) when the word embeddings are learned specifically for the task, directly from the training dataset. All the models were implemented in the Tensorflow framework, and were run on a distributed cluster with multiple GPU machines (Nvidia p80) due to the size of the data. The initial learning rate of 0.0001 was set for the Adam Optimizer, while the mini-batch size was set to 512.

4.1.3 Metrics. For assessing the quality of the estimated CTR probabilities, we use a common classification performance measure of area under the ROC curve (AUC), as well as Accuracy obtained after choosing the appropriate classification threshold. In addition, we study the bias of the predicted probabilities. Unbiasedness is a desirable property, as positive bias leads to overly-optimistic estimates and a waste of resources, and negative bias leads to overly-conservative estimates and a waste of opportunity.

4.1.4 Results. Prediction performance results, on the holdout testing dataset, are presented in Fig. 3. Results in the Table1 are the best results obtained by the respective model. The DSM approach outperforms all the alternatives with the highest AUC of 0.775.

We first evaluate the simplest way (LM) of learning to predict CTR from combined text data of query and ad, and we observe a decent performance of such an approach, which resonates well with

Table 1: Performance of the proposed models vs baselines

Model	DSM	MT	DM	VDCNN	LM
AUC	**0.775**	0.745	0.755	0.744	0.711
Bias	**0.991**	1.046	1.033	0.974	0.965
Accuracy	**0.742**	0.703	0.719	0.734	0.711

the word embedding approaches described in Section 2. Furthermore, we see that by introducing deep models, such as VDCNN, we are able to achieve significant lifts in performance (3% lift in AUC). However, by introducing individual embeddings of query and ad to capture specificities of both, and learning to match the two, such as in the case of the DM or MT models, we see that the results are further improved (1% lift in AUC). Finally, when a model is capable of capturing discriminative features deep in the architecture, we obtain further improvements (additional 2% of AUC lift). Accuracy measure consistently sets DSM as the best performing model.

Furthermore, we evaluate the bias of predictions made by different models, and observe that the DSM model is the most unbiased model in the experiment (closest to the ratio of 1). This implies that the expected number of clicks deviates the least from the exact number of clicked ads, thus achieving better monetization. The results show that the *DSM* model's click expectation would on average be wrong for 9 clicks, out of 1000, which is 17 clicks better compared to the next best *VDCNN* model, with 26 out of 1000. This significantly impacts revenue due to a volume of served ads.

Learn word embeddings vs. use pre-trained word vectors. As all baselines suggest using pretrained word embeddings in their original approaches, we examined the effect of learning embeddings in an end-to-end manner, rather than using pretrained ones. Results in Figure 3 show that the models where the word embeddings are learned directly on the task of CTR prediction, in a majority of cases are superior to their counterparts which use pre-trained vectors. Thus, we argue that it is important for such models to capture word specificities of the domain rather than using external embedding.

The following two experiments show results obtained by the best version (using pretrained word vectors vs. learning word embedding) of the respective model.

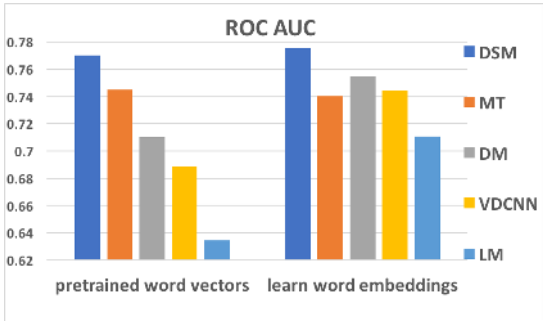

Figure 3: Models with learned embeddings (on the right) perform better than models with pretrained vectors (on the left)

CTR prediction for Head, Torso and Tail Queries. It is expected that predictability of CTR depends on the query frequency. For example, for less frequent queries there may not be enough data to generalize properly. Therefore, in this subsection, we analyze the influence of the query frequency on the model predictive performance. For that purpose, examples were divided into three categories: the most frequent "head" (">20" occurrences), least frequent "tail" ("<5" occurrences), and "torso" in-between.

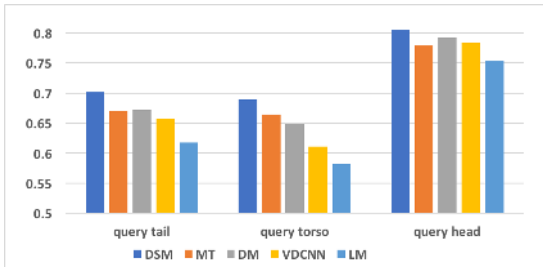

Figure 4: AUC for CTR decomposed by query frequency

Results presented in Figure 4 align with the common sense expectation that the most frequent queries ("query head") will be more predictable. The less frequent "torso" and "tail" queries have expectedly lower AUC (more than 10% less than "head"), where the least frequent queries (from "tail") seems to have slightly higher predictive performance, compared to the "torso".

CTR prediction over different Ad Positions. It was acknowledged that Ad position plays an important role in CTR prediction [4]. For example, ads placed in the north section are more likely to be clicked than those in the south or east sections, both because it was considered the most relevant (by algorithm), and because its position is the most favorable (convenient) one. Therefore, we also analyze the influence of the ad position on the model predictive performance. For that purpose, we segregated the examples into 5 groups based on their positions in the north section (top one is no. 1, and up to 5, as it goes down). Results presented in Fig. 5 convey that predictability decays with the rise in the position number. From the first to the second position it displays the sharpest decrease in the AUC, and from-then-on it goes more gradually until the last, fifth position. Still, the proposed model is the best on all sections.

CTR prediction - training set scale impact. We also studied models training on datasets of different scales, small with millions of examples, and large with billion of examples. As shown in Fig. 6, scale matters when trying to characterize models for ad impression data. For example, models that use pretrained word vectors perform

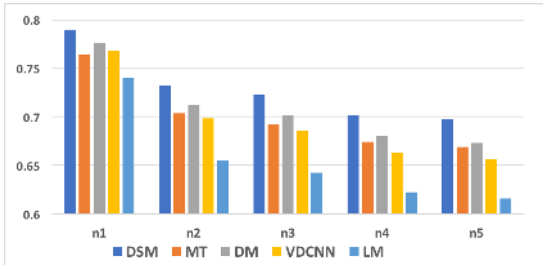

Figure 5: AUC for CTR decomposed by impression position

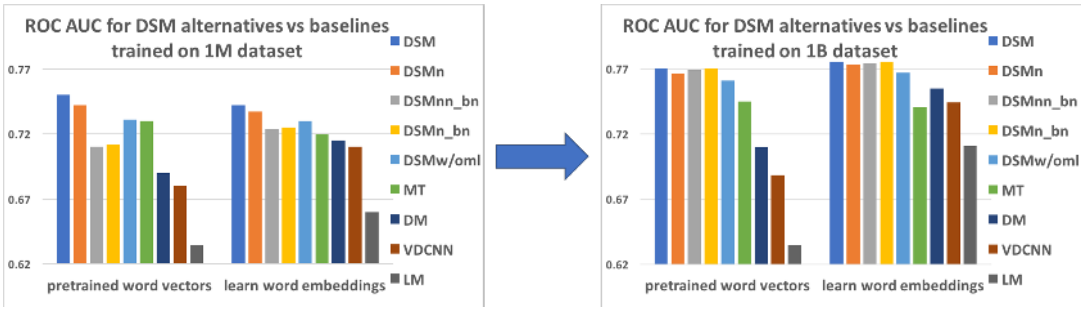

Figure 6: Effect of Data set scale on models' CTR prediction performance

better on smaller dataset than their learn-embeddings alternatives, as the models that learn embeddings require more data to learn meaningful representations of words. We also note that batch normalization algorithms on smaller datasets perform much worse than their non-batch-normalizing alternatives, which is not the case on the larger dataset, suggesting that algorithms that are using batch normalization need more data to learn good representations.

Robustness of the DSM model. Additionally, for the proposed methodology ablation analysis, we study the effect of batch normalization, loss normalization, and attention pooling when we remove the deep supervision from the DSM model. Hence, we had four varieties of our model: plain *DSM*, *DSM* with normalization of the two losses (trying to prevent one of the losses dropping too fast) DSM_n, DSM with batch normalization on the fully connected layers DSM_{bn} to prevent large fluctuations of the logistic loss and DSM with both batch normalization and normalized losses DSM_{n_bn}.

Results of all exploited normalization strategies yield comparable prediction performance with 0.7754, 0.7734, 0.7743 and 0.7727 AUC for *DSM*, DSM_n, DSM_{bn} and DSM_{n_bn}, respectively.

Logistic loss vs. matching loss. Finally, we removed the matching loss from the model to evaluate the gain obtained by it. Furthermore, we noticed that the matching loss drops much faster than the logistic loss, even after losses normalization. That confirms that the surrogate loss served as a form of regularization [20] that forces the hidden layer of a query and ad representations to be semantically discriminative thus yielding higher quality CTR predictions and enabling the model to excel on matching tasks. We see a larger drop when removing the matching loss with 0.7671 AUC (the Wilcoxon signed-rank test p–value $8.63e^{-05}$), thus validating that the matching loss benefits the quality of the CTR prediction.

4.2 Query2Ad Matching

Finally, we assess the quality of the learned representations. The proposed DSM learn semantic matching of a query-ad pair as an effect of the matching layer and deep supervision. To validate this, we evaluate our model on the query to ad (query2ad) matching task, traditionally used for performance assessment. Note that this is not the primary task of the DSM, however, due to the nature of the proposed matching, it has the ability to perform it well. The scores between query and ad used for matching are the final layer's logits, that reflect query-ad semantics as well as the click probability.

4.2.1 Relevance data. To evaluate the quality of query and ad embeddings, we used an in-house dataset consisting of a query-ad pair that was graded editorially. The editors were instructed to grade 65, 446 query-ad pairs as either Perfectly Relevant, Highly Relevant, Relevant, Somewhat Relevant, Barely Relevant, or Irrelevant as in [1]. For each ad, the editors had access to ad title, description, and display URL to help them reach their judgment. For each query (8, 315 unique queries) there was on average ~ 7 graded ads, allowing us to evaluate ranking of ads in addition to relevance.

4.2.2 Baselines. We compared our method to traditional relevance models: Gradient boosted decision trees (with 1000 trees) [38] ($GBDT_{1000}$), with 185 text-based features [1] (trained on 700, 000 editorial query-ad pairs) and the BM_{25} [28]. We also use other CTR prediction task baselines (described in Section 4.1.2), where, as for the DSM, logits of the models were used as matching scores. Finally, we evaluated the search2vec [11] for the matching task. Since the model is only trained for known queries and clicked ads, the coverage of the model on our editorial dataset was small (2,167 unique queries coming from 8,725 query ad pairs out of 65,446 records) and as such model yielded only fairish results ([0.7, 0.8] for NDCG@2 to NDCG@7), so we do not show them in Figure 7.

For matching quality, we use *precision@K* and Normalized Discounted Cumulative Gain *NDCG@K* [33] averaged across all queries.

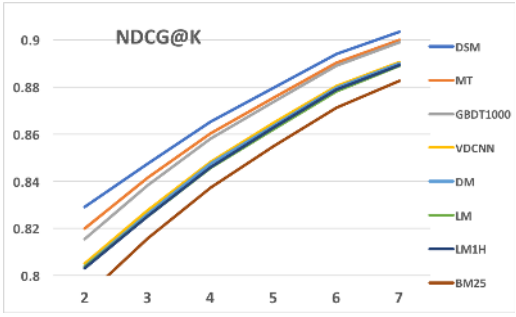

Figure 7: NDCG@K on editorial 65K query-ad pairs

4.2.3 Matching Results. ***NDCG.*** Relevance was assessed using the *NDCG@K* [33], and the results are given in Figure 7. We observe that the DSM approach improves over the alternatives (higher values of NDCG). Even though the difference is not obvious because

of the $NDCG@2$ to $NDCG@7$ scores' scale, Wilcoxon signed-rank test p-value of $2.69e^{-05}$ measured on $NDCG@1$ to $NDCG@100$ shows that the improvement of the DSM model over alternatives is statistically significant. DSM improves the NDCG@7 of the GBDT model by 2% and the best deep learning baseline MT by 0.5%. In addition, we measure Precision@K for all the models, but for the lack of space we report here statistically significant average improvement of 1.5% over the next best alternative.

Precision. We also measure Precision@K to further characterize models, as shown in Figure 8. The *DSM* model is still the best performing model. However for this metric, traditional BM_{25} model performs as the second best model. Statistical significance test of the improvement of the *DSM* over the BM_{25} model returns p-value of $8.85e^{-05}$, confirming that observed improvements are indeed statistically significant.

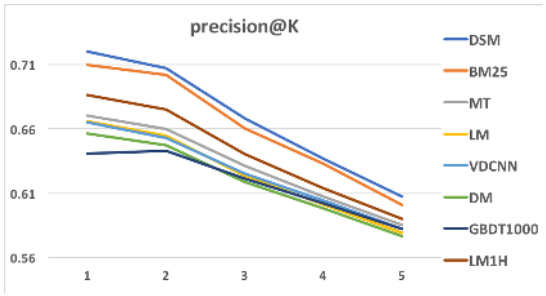

Figure 8: precision@K measured on editorial judgments of 65K query-ad pairs

5 FINAL REMARKS

The results of our extensive experiments demonstrate that the proposed DSM model outperforms state-of-the-art approaches on CTR prediction tasks, as measured by multiple metrics. It was the most accurate, and had the least bias of all the approaches. Our model also outperformed other competitive algorithms on a query to ad matching task, as measured by the NDCG. Ablation study confirmed that the dual loss architecture (statistically significant) enhances the model performance. Moreover, our DSM model was the best performer over different scales of data, frequencies of the queries, ad positions and embedding choices. Above mentioned suggests that joint training of two complementary tasks, as query to ad matching and CTR prediction are, through deep supervision, yields high quality, versatile models.

REFERENCES

[1] Luca Aiello, Ioannis Arapakis, Ricardo 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 International on Conference on Information and Knowledge Management*. ACM, 185–194.

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

[3] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 767–776.

[4] Ye Chen and Tak W Yan. 2012. Position-normalized click prediction in search advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 795–803.

[5] 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*. ACM, 351–360.

[6] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781* (2016).

[7] Bora Edizel, Amin Mantrach, and Xiao Bai. 2017. Deep Character-Level Click-Through Rate Prediction for Sponsored Search. *arXiv preprint arXiv:1707.02158* (2017).

[8] Ariel Fuxman, Panayiotis Tsaparas, Kannan Achan, and Rakesh Agrawal. 2008. Using the wisdom of the crowds for keyword generation. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 61–70.

[9] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.

[10] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 13–20.

[11] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, R. Baeza-Yates, A. Feng, E. Ordentlich, L. Yang, and L. Owens. 2016. Scalable Semantic Matching of Search Queries to Ads in Sponsored Search Advertising. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[12] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, and N. Bhamidipati. 2015. Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[13] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* (2017).

[14] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2333–2338.

[16] Aaron Jaech, Hetunandan Kamisetty, Eric Ringger, and Charlie Clarke. 2017. Match-Tensor: a Deep Relevance Model for Search. *arXiv preprint arXiv:1701.07795* (2017).

[17] Zilong Jiang. 2016. Research on ctr prediction for contextual advertising based on deep architecture model. *Journal of Control Engineering and Applied Informatics* 18, 1 (2016), 11–19.

[18] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[19] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

[20] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *Artificial Intelligence and Statistics*. 562–570.

[21] Hang Li, Jun Xu, et al. 2014. Semantic matching in search. *Foundations and Trends® in Information Retrieval* 7, 5 (2014), 343–469.

[22] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*.

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

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*. 3111–3119.

[25] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[26] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. 2016. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*. 314–323.

[27] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 521–530.

[28] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc., 232–241.

[29] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.
[30] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[31] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 255–262.
[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. 2015. Going deeper with convolutions. Cvpr.
[33] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*.
[34] Shuangfei Zhai, Keng-hao Chang, Ruofei Zhang, and Zhongfei Mark Zhang. 2016. Deepintent: Learning attentions for online advertising with recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1295–1304.
[35] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.
[36] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks.. In *AAAI*. 1369–1375.
[37] Yuting Zhang, Kibok Lee, and Honglak Lee. 2016. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *International Conference on Machine Learning*. 612–621.
[38] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. 2008. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*. 1697–1704.