Data Mining and Knowledge Discovery (2019) 33:1446–1467
https://doi.org/10.1007/s10618-019-00625-3

Deeply supervised model for click-through rate prediction in sponsored search

Jelena Gligorijevic[1] · Djordje Gligorijevic[1] · Ivan Stojkovic[1] · Xiao Bai[1] · Amit Goyal[2] · Zoran Obradovic[3]

Received: 13 August 2018 / Accepted: 26 March 2019 / Published online: 3 April 2019

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, limiting their 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.

Keywords Deep learning · Click 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 click-through-rate (CTR) in order to maximize the expected

Responsible editor: Po-ling Loh, Evimaria Terzi, Antti Ukkonen, Karsten Borgwardt.

Jelena Gligorijevic and Djordje Gligorijevic have contributed equally to this work.

Amit Goyal: The work was done when the author was with Yahoo Research.

Extended author information available on the last page of the article

revenue while ensuring good user experience. Both overpredicting and underpredicting CTR would result in revenue loss to search engines, as they would either allocate limited search result slots to unattractive ads or miss opportunities to show attractive ads.

Machine learning models have achieved 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 (Cheng and Cantú-Paz 2010). However, models that heavily rely on click features often fail to generalize to new queries and new ads with insufficient history (Richardson et al. 2007). To make predictions in such cases, models resort to syntactic or semantic features extracted from queries, ads, and advertisers (Richardson et al. 2007; Li and Xu 2014). Deep neural networks were also proposed to learn features from traditional models (Jiang 2016) or to learn CTR from existing features (Zhang et al. 2014). In spite of the existing success, designing and selecting appropriate features remains a very challenging problem for CTR prediction (He et al. 2014).

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 (Edizel et al. 2017). However, since such models are learned end-to-end from clicks without explicit supervision for capturing the semantic similarity between a query and an ad, they have not achieved their full potential in CTR prediction, as we show in this work.

A number of recent works (Grbovic et al. 2016; Jaech et al. 2017) used deep neural networks to model the semantic similarity between a query and an ad. These models were shown to be effective in a query-ad relevance matching. However, as they do not directly model clicks, retrieved ads can be 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 CTR prediction and semantic matching, 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)* implicitly learning semantic-rich discriminative representations to enable query-ad matchings more correlated with clicks and expected revenue. The key contributions of this work are as follows:

– 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 effective attempt to simultaneously learn CTR and semantic embeddings using click data. By optimizing two losses specific to CTR prediction and semantic matching instead of using only one CTR specific logistic loss, we were able to

achieve statistically significant increase in the area under the receiver-operating characteristic curve (AUC).
- We propose a novel cohort negative sampling technique to naturally draw information from implicit negative signals in the data.
- We conduct an extensive empirical evaluation of the proposed architecture using about one billion query-ad samples from a major 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 normalized discounted cumulative gain (NDCG) of the best-performing baseline by a statistically significant 0.5%, confirming its ability to learn meaningful semantic embedding.

2 Related work

In this section, we first present problems and challenges in sponsored search and review the most recent advances in deep learning approaches applied to this area. Subsequently, we review other relevant advances in deep learning, which have not previously been applied to sponsored search tasks.

2.1 Related work in sponsored search

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

A large body of work focuses on predicting the probability that an ad would be clicked, if shown as a response to a submitted query (Graepel et al. 2010; McMahan et al. 2013; He et al. 2014). 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 (Richardson et al. 2007). These approaches range from Bayesian (Graepel et al. 2010) to feature selection approaches (He et al. 2014), however, a common challenge for all is creating and maintaining a large number of sparse contextual and semantic features (McMahan et al. 2013). Manually designing and selecting features requires substantial investment of human time and effort, and utility of such generated features is largely dependent on the domain knowledge of human experts curating the features (McMahan et al. 2013). Moreover, since typical applications are nonlinear, considering feature interactions (e.g cross-features) quickly becomes prohibitively expensive due to a combinatorial explosion (Bhamidipati et al. 2017).

More recently, many approaches for CTR prediction stared using deep learning techniques, primarily to alleviate some of the mentioned issues with handcrafting features. Deep learning models automatically learn informative and nonlinear features directly from the "raw" query and ad text data, but at the cost of requiring huge amounts

of data and computational power. In Shan et al. (2016), 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, like DeepMatch (Edizel et al. 2017) and MatchTensor (Jaech et al. 2017), proposed very deep dual network architectures for query and ad embeddings with a matching layer to learn ad impression representations useful for CTR prediction.

Focusing on the matching of queries and ads that have similar semantic meaning is another line of research (Fuxman et al. 2008; Zheng et al. 2008; Robertson and Walker 1994). The task is to retrieve ads that are semantically similar to the query (Grbovic et al. 2016) without exactly matching keywords (i.e. query "running machine" and ad "elliptical trainer"). This task has been commonly addressed by query rewriting models (Jones et al. 2006) or by semantic matching (Grbovic et al. 2016; Fuxman et al. 2008; Huang et al. 2013).

Huang et al. (2013) proposed a deep structured semantic model (DSSM) with dual architecture that embeds a query on the one side and a web search document on the other and learns matching between the two given the click information. In order to improve quality of the learned semantic match and to capture query intent, a word attention mechanism was successfully used for the query and ad representations (Zhai et al. 2016). Pair-wise loss is common in most recent approaches (Guo et al. 2016; Mitra et al. 2017) that deal with query-to-document relevance modeling. For web search, relevance is the most important factor to determine the ranking, therefore the goal is to ensure that a relevant document is ranked better than an irrelevant document through the pairwise loss function. However, for sponsored search, the actual CTR value is also important in addition to ranking by relevance, therefore a point-wise loss function is more appropriate for the CTR prediction. Thus, we focus on point-wise approaches and do not compare experimentally to the aforementioned pairwise ranking models, due to their similarity to other baselines in building blocks and reported underperformance against more traditional baselines (Robertson and Walker 1994), which our baselines largely outperformed (Edizel et al. 2017).

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. Another recent work (Yan et al. 2018) combines click information and semantic similarity matching, but in an in effective way, as may be seen from the achieved results. The approach we propose in this study is a well-rounded and effective framework for ad systems capable of both learning quality semantics of queries and ads as well as accurately predicting click probability.

The two mentioned approaches, DeepMatch (Edizel et al. 2017) and MatchTensor (Jaech et al. 2017) 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 the way they learn representations of words, i.e. DeepMatch primarily uses temporal convolutional layers, while MatchTen-

sor 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 Sect. 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 Rumelhart et al. (1988) and recently successfully applied for learning semantic and syntactic relations among words or tokens (Mikolov et al. 2013b). The idea of using distributed representations of words was further exploited in approaches such 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 (Greff et al. 2017).

bi-RNNs Another successful paradigm is the bidirectional-RNN (bi-RNN) approach, where two RNNs (i.e. LSTM, thus bi-LSTM) independently encode the text sequence in both forward and in backward direction (Schuster and Paliwal 1997) 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. Therefore, we employ this approach in the proposed model.

Attention Network Models Attention models dynamically re-weight 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 (Bahdanau et al. 2015; Gligorijevic et al.

2018b). 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 (Zhai et al. 2016; Gligorijevic et al. 2018b, a). We utilize this mechanism for the summarization of query and ad representations in the proposed model.

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 (Zhang et al. 2015) and the Very Deep CNN (VDCNN) model (Conneau et al. 2017), 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 comparable to the DeepMatch model (except for the matching layer in Deep Match).

Deeply supervised models Recently, several models drew benefits from utilizing deep supervision (Zhang et al. 2016; Lee et al. 2015; Szegedy et al. 2015). The key idea is to use supervision at various layers across the model to enforce discriminativeness of the features (Lee et al. 2015) and potentially resolve the exploding/vanishing gradients problem (Zhang et al. 2016; Szegedy et al. 2015). However, the existing approaches mostly use the same predictive task in lower layers as in the final layer (Lee et al. 2015; Szegedy et al. 2015) and in some cases use reconstruction loss (Zhang et al. 2014). We build upon these advances proposing a novel approach of using deep supervision specifically designed to extract information from data in an explicit way, semantically controlling the discriminativeness of lower layers in deep architectures.

Due to their similarity, it is worth to contrast deep supervision to multi-task learning approaches (Liu et al. 2016). Multi-task learning approaches use supervision on the top of their architecture with the goal to optimize the algorithm for several tasks simultaneously. Deep supervision, on the other hand, operates by addressing the originally ill-posed optimization problem by limiting the space of solutions to ones that satisfy the imposed supervision on deeper layers. As such, deep supervision has a regularization effect (Lee et al. 2015) by imposing a prior on lower layers, although it should not be confused with regularization.

Learning from implicit negative signals Modeling implicit negative signals in search sessions has been a challenging task for a long time. Recently, search2vec model for learning with implicit negative signals from sponsored search sessions was proposed (Grbovic et al. 2015) with improved performance and speed of the algorithm. Furthermore, Chen et al. (2017) have confirmed this approach and applied it on the special case of bipartite graphs. We exploit implicit negatives in our model and consider comparing experimentally to the search2vec algorithm in Sect. 4.2.

3 Proposed model

Graphical representation of the proposed model, which we call the Deeply Supervised Matching (DSM) model is given in Fig. 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 matching loss. The learned matching tensor is then passed through a 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 performed in two separate parts of the network. First, l_q words in the query and l_a words in the ads are embedded into a $d_{qa}^{(1)} = 300$ dimensional common space (*word embeddings layer*). The word embeddings layer is common between queries and ads, as queries and ads come from the same language and share the same vocabulary. Then, a fully connected layer (300×40) is used to learn linear combinations of words in a $d_{qa}^{(2)} = 40$ dimensional space. This layer shares weights for both query and ad word vectors, similar to the embedding layer. This layer is introduced to allow pre-trained embeddings, as well as to enable the size of the embeddings to be varied without relearning the word embeddings from scratch each time. The output of the first projection layer is an ($l_q \times 40$) embedding for query and an ($l_a \times 40$) embedding for ad. The ($l_q \times 40$) query embedding is then passed to the query bi-LSTM, and the ($l_a \times 40$) ad embedding is passed through the ad bi-LSTM layer, 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 (Jaech et al. 2017; Zhai et al. 2016). Finally, fully connected layers are used to reduce representations of all words to 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. 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 (Zhai et al. 2016). Two attention blocks are used, one for query text and one for ad text. These blocks yield word scores, that signify the attentions that the

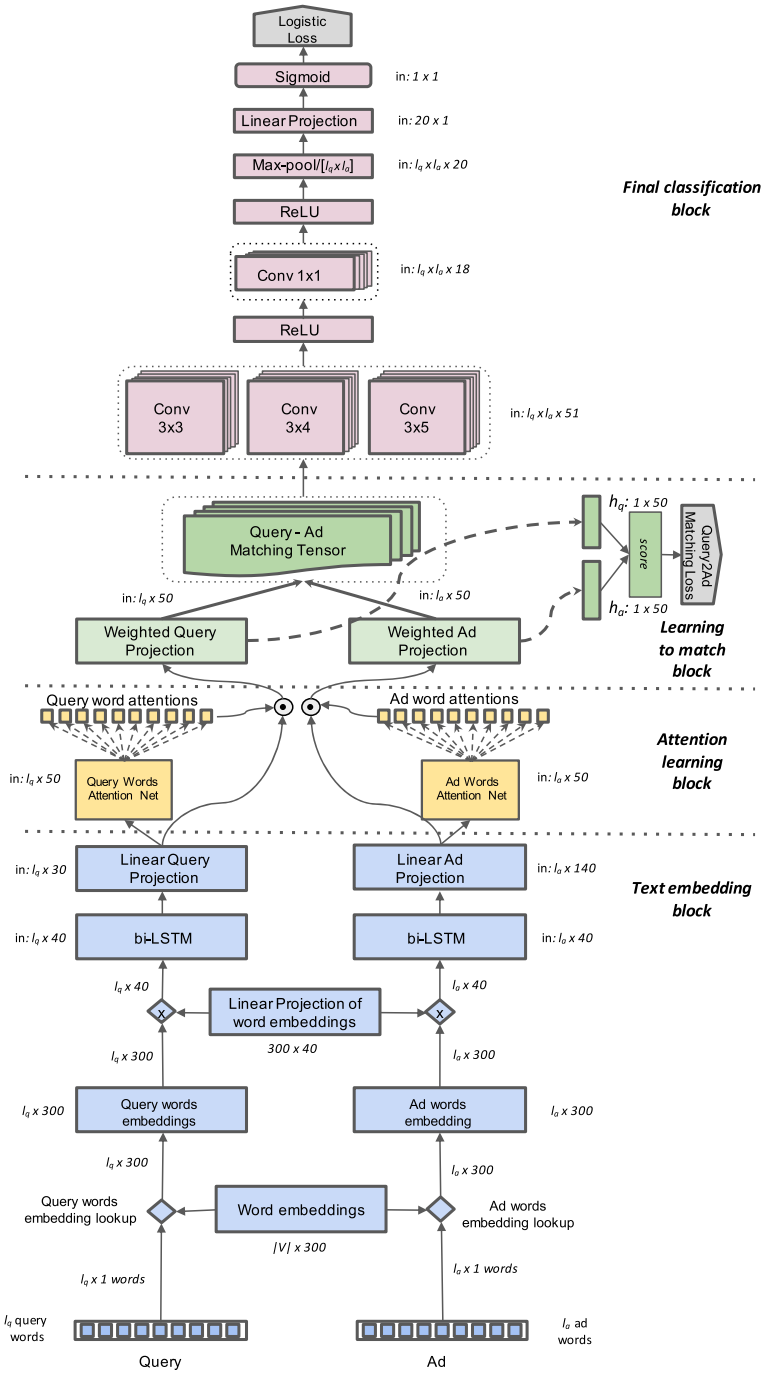


Fig. 1 Proposed DSM model block diagram

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

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

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 a simple sum of element-wise multiplications ($*$): $h_q = \sum_i t_q^{(i)} * v_q^{(i)}$ 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 (Zhai et al. 2016; Gligorijevic et al. 2018b), 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 high-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 (Jaech et al. 2017), 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 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 (Jaech et al. 2017).

Precisely, using the learned higher-order representations of words in a query Q of shape $l_q \times d_{qa}^{(4)}$ and an ad A of shape $l_a \times d_{qa}^{(4)}$, a tensor is built for their implicit matching. Matching tensor H of the shape $l_q \times l_a \times d_{qa}^{(4)}$ is created in the following manner:

$$H(m, n, :) = Q(m, :) \odot A(n, :), \tag{2}$$

where $m, n \in \mathbb{N}$, $1 \leq m \leq l_q$, $1 \leq n \leq l_a$ and $\odot$ represents the element-wise product of $Q_i(m, :)$ and $A_i(n, :)$. 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 (Jaech et al. 2017).

Second, we propose a 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 further away if there was no click, similarly to Grbovic et al. (2016). 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.

Benefits of using deep supervision have recently been recognized (Lee et al. 2015). 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 (rectified linear unit) layer, after which another 1×1 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 (denoted as W in remainder of the text), we have obtained logistic loss $\mathcal{P}$ for the CTR prediction based on logits from the topmost layer:

$$\mathcal{P}(W) = -\frac{1}{N}\sum_{n=1}^{N}\left(y_n \log(\hat{y}_n) + (1 - y_n)\log(1 - \hat{y}_n)\right), \tag{3}$$

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 (Mikolov et al. 2013a):

$$\begin{aligned}\mathcal{Q}(W) &= \sum_{b=1}^{B}\left(\sum_{j\in\mathcal{D}_p^{(b)}}\mathcal{Q}^{+}(W)+\sum_{k\in\mathcal{D}_n^{(b)}}\mathcal{Q}^{-}(W)\right)\\ &= \sum_{b=1}^{B}\left(\sum_{j\in\mathcal{D}_p^{(b)}}-\log\sigma(h_q^{(j)T}h_a^{(j)})+\sum_{k\in\mathcal{D}_n^{(b)}}\log\sigma(-h_q^{(k)T}h_a^{(k)})\right),\end{aligned} \tag{4}$$

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 Mikolov et al. (2013a), 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 Eqs. 3 and 4

$$\mathcal{L}(W) = \mathcal{P}(W) + \mathcal{Q}(W). \tag{5}$$

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

Based on the Lemma 1 in Lee et al. (2015), a good solution for $\mathcal{Q}$ is also a good solution for $\mathcal{P}$. However, the converse 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 (Lee et al. 2015), 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 task and compared it to well-established models designed for that task in Sect. 4.2.

Weights are initialized by a truncated normal initializer. To optimize $\mathcal{L}$, we use Adam (Kingma and Ba 2015) 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 in 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 (Chen and Yan 2012). Up to five ads can be presented at this location, and users may or may not click on any of them. Click/No-click information provides implicit information on query and ad relevance that we can learn from. An example can be query "*things to do in Paris*" for

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

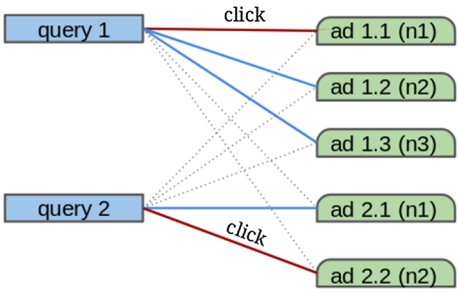


Fig. 2 Cohort negative sampling (an example with queries and served ads in the position "north", n1 up to n5) Red links are ad clicks, blue links are ads displayed but not clicked, and negative pairs we recreated by coupling queries and ads that were not displayed for that query—gray dotted links (Color figure online)

which are shown three ads: "*guided bike tour in Paris*", "*secret food tours Paris*" and "*Louvre tickets and tours*", such that a user clicked on the "Louvre tickets and tours" ad only. 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/batch 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 (Grbovic et al. 2016), while its computational benefits were supported in Chen et al. (2017). In this study, unlike in Chen et al. (2017), implicit negative samples naturally occur as signals from the users. Furthermore, in Chen et al. (2017) a complete ground-truth bipartite graph is needed to obtain a good working model, however, 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 Fig. 2.

Originally, techniques such as negative sampling were proposed for efficient approximation of the costly normalization function in Softmax, while learning to match (Mikolov et al. 2013a). However, to implement negative sampling in DSM, for each of m positive samples (in our case clicked query-ad (q, a) pairs) and n negative samples in the given cohort (not clicked query-ad (q, a) pairs), k new ads need to be randomly sampled from the distribution P_n of negative ads for each given query q (i.e. ads that were never shown for the query, like ad "*hotels in Melbourne*" for query "*things to do in Paris*"), resulting in a total of $m+n+m*k$ embedding operations prior to matching (as in every moment in one cohort we have readily accessible embeddings for queries and ads from that cohort only)

In the case of cohort negative sampling, there is no need for additional sampling of k negative ads, as embeddings of ads displayed for (other) different queries within the given cohort are already calculated and k random ads can be selected for constructing negative pairs within the cohort (dotted gray links in the Fig. 2), thus the computation is decreased by $m*k$ from the standard negative sampling approach. Using this strategy, we can obtain $m*(b-1)-n$ additional negative pairs in the cohort of size b, in addition to existing n implicit negative pairs. The cohort negative sampling is based on the assumption/observation that queries in a batch are mostly unrelated to each other, making the matching ads of other queries "negative" to the query being examined.

4 Experiments

In order to assess the capabilities of the proposed DSM approach, 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 (Sect. 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 (Sect. 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. In the remainder we will refer to this shortly as $p(s)$.

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, split URLs to domain and subdomain (for example, www.cnn.com/specials/space-science becomes "www.cnn.com specials space science"). Furthermore, for query canonization, text processing tools such are typo reduction and stemming are used to obtain canonic forms of queries and match them to their canonic forms if possible, i.e. canon query "*buy iPhone X*" from query "*buying iPhone X*". All example pairs are accompanied with information on whether the ad was clicked or not, which we use as supervised information to train all models. To better characterize the dataset, we comment on its distribution of 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 traditional history-based CTR prediction models, and given the volume of the tail queries, this reaffirms the necessity for predictive, broad-match, 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 Sect. 2.2: A linear logistic regression learned on top

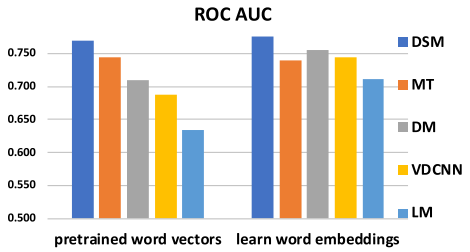


Fig. 3 Models with learned embeddings (on the right) perform better than models with pretrained vectors (on the left)

of the word embedding layer (LM), Very Deep CNN (VDCNN) (Conneau et al. 2017), DeepMatch (DM) (Edizel et al. 2017), and MatchTensor (MT) (Jaech et al. 2017).

All deep learning models were trained in two ways: (i) with the use of pre-trained word embedding vectors (obtained from Pennington et al. (2014)); 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 on Spark[2] framework and were run on a distributed cluster with multiple GPU machines (NVIDIA Tesla K80) due to the size of the data. The number of executors per experiment was 8. Mini-batch size was set to 512, giving about 2 million updates in total (250,000 updates per executor) for 1 iteration through the bigger ($\sim$1 billion records) dataset and models were trained in 4 iterations. Training of one iteration took approximately 1 day for all the models. The initial learning rate of 0.0001 was set for the Adam Optimizer.

4.1.3 Metrics

For assessing the quality of estimated CTR probabilities, we use the area under the ROC curve (AUC) classification performance measure, as well as Accuracy obtained after choosing the appropriate classification threshold. In addition, we study the bias (Baeza-Yates and Ribeiro-Neto 1999) of the predicted probabilities defined as the ratio between sum of sample ($s \in S$) click probabilities $p(s) \in [0..1]$, and sum of click labels $l(s) \in \{0, 1\}$: $Bias = \sum_{s \in S} p(s) / \sum_{s \in S} l(s)$.

Unbiasedness (bias=1) is a desirable property, as higher than 1 bias implies overly-optimistic estimates and waste of resources (bidding where there is a lower chance of click), and lower than 1 bias implies to overly-conservative estimates and missed opportunity (not bidding where there is a higher probability of click).

4.1.4 Results

Prediction performance results, on the holdout test set are presented in Fig. 3. Results shown in Table 1 are the best results obtained by each 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 the word embedding approaches described in Sect. 2. Furthermore, we see that by introducing deep models, such as VDCNN, we are able to

[2] https://github.com/yahoo/TensorFlowOnSpark.

Table 1 Performance of the proposed models versus 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

achieve significant improvement in performance (3% increase 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% increase in AUC). Finally, when a model is capable of capturing discriminative features deep in the architecture, we obtain further improvements (additional 2% of AUC increase). The accuracy measure consistently identified DSM as the best performing model.

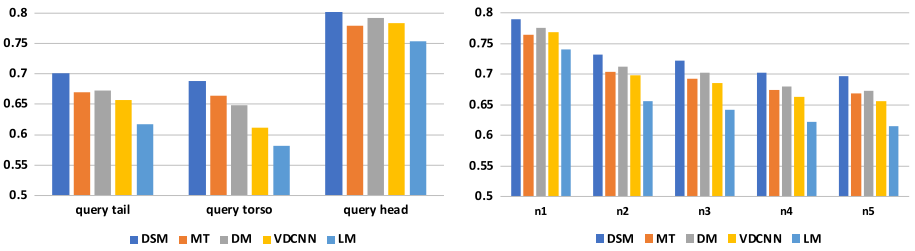
Furthermore, we evaluate the bias of predictions made by different models and observe that the DSM model is the most unbiased model in the experiment (closest to the ratio of 1, as shown in Table 1). This implies that the expected number of clicks deviates the least from the exact number of clicked ads, 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 high volume of served ads.

Learn word embeddings versus 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 Fig. 3 show that the models where the word embeddings are learned directly on the task of CTR prediction, are superior to their counterparts which use pre-trained vectors in a majority of cases. Thus, we argue that it is important for such models to capture word specificities of the domain rather than using an external embedding that may be suboptimal.

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

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.

CTR prediction over different Ad Positions It was previously noted that ad position plays an important role in CTR prediction (Chen and Yan 2012). 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



(a) AUC by query frequency **(b)** AUC by ad impression position

Fig. 4 AUC for CTR decomposed by **a** query frequency and **b** ad position

Table 2 AUC for CTR decomposed by query frequency for the DSM model trained on 1 billion versus 1 million records

	Query frequency		
	Tail	Torso	Head
1B dataset	0.701	0.689	0.805
1M dataset	0.662	0.656	0.778
Improvement	3.9%	3.3%	2.6%

its position is the most favorable (convenient) one. Therefore, in this subsection, 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 (n1 to n5). Results presented in Fig. 4b 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 positions, with even larger improvements over baselines on less attractive ad locations, demonstrating its potential to capture wider semantic relations between a query and an ad.

CTR prediction - training set scale impact We also studied model training on datasets of different scales, including small-scale data with million (1M) of examples, and large-scale data with billion (1B) of examples. Dataset scale is an important aspect of model performance, as a model trained on small data would underperform on larger sets of data, failing to observe enough words in right contexts. More data is particularly helpful for tail queries whose words may not be observed in small training data. If we compare the DSM model trained on 1B dataset with the DSM model trained on the smaller 1M dataset on tail,torso and head queries, we see in Table 2 that the largest improvement ($\sim 4\%$) is obtained exactly for the tail queries.

As shown in Fig. 5, scale matters when trying to characterize models for ad impression data. For example, models that use pre-trained word vectors perform 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 the algorithms that use a batch normalization approach perform much worse on smaller datasets than their non-batch-normalizing alternatives, which is not the case on the larger dataset, suggesting that algorithms that use batch normalization need more data to learn good representations. It is important to note the difference

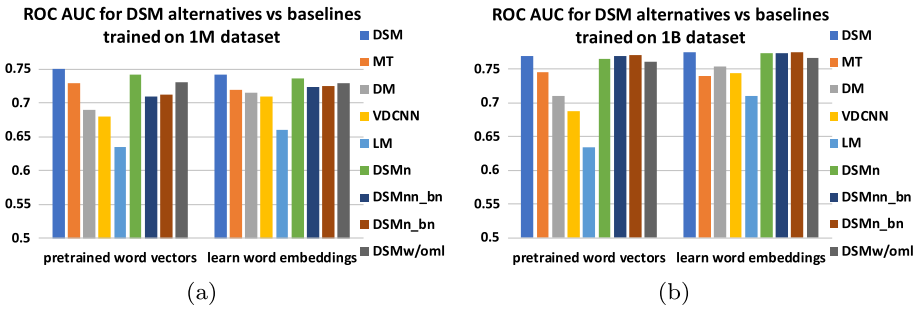


Fig. 5 Effect of Data set scale on models' CTR prediction performance

in the order of baselines with the change in scale of the data. This clearly suggests a need to evaluate novel deep architectures on very large datasets. Variants of the proposed DSM model are top-performing on both smaller and larger scale-data, obtaining statistically significant improvements over all baselines.

Robustness of the DSM model optimization For the proposed methodology ablation analysis, we first analyze how batch normalization affects the performance, as out-of-the-distribution "tail" queries can cause loss fluctuations during the optimization. Second, as the two losses are used simultaneously on different depths, the deep supervision can reach zero much faster than the main loss, therefore, in order to analyze performance when the deep supervision loss is prevented from easily reaching zero, we normalized the two losses to the same scale. 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. This provides evidence for the stability of optimization of the DSM model under various discussed situations.

DSM without matching loss Finally, we removed the matching loss from the DSM model to evaluate 1) the performance gain obtained by using the attention layer compared to the MT model we built upon and 2) the performance loss when using logistic loss only (without using the matching loss).

From Fig. 5 we see a larger drop from 0.775 AUC for DSM to 0.7671 AUC for $DSM_{w/oml}$ when *removing the matching loss* (the Wilcoxon signed-rank test p–value $8.63e^{-05}$), thus validating that the matching loss benefits the quality of the CTR prediction. We also notice the increase of 2% in AUC that *the attention layer* brings when compared to the TM model AUC of 0.745 (the Wilcoxon signed-rank test p–value $7.35e^{-05}$), confirming that highlighting important terms in queries and ads by weighted attentions brings performance improvements.

In further analysis, we noticed that the matching loss drops much faster than the logistic loss, even after losses normalization. That confirms that the matching loss

served as a regularization surrogate (Lee et al. 2015) that forces lower layer of query and ad representations to be semantically discriminative thus yielding higher quality CTR predictions and enabling the model to excel on matching tasks.

4.2 Query-to-ad matching

Finally, we assess the quality of the learned representations. The proposed DSM learns 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 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, which reflect query-ad semantics as well as the click probability. Using scores from the matching loss is avoided, as it leads to suboptimal performance because it represents a surrogate loss.

4.2.1 Relevance data

To evaluate the quality of query and ad embeddings, we used an in-house dataset consisting of editorially graded query-ad pairs. The editors were instructed to grade 65446 query-ad pairs as either Perfectly Relevant, Highly Relevant, Relevant, Somewhat Relevant, Barely Relevant, or Irrelevant as in Aiello et al. (2016). For each ad, the editors had access to ad title, description, and display URL to help them reach their judgment. For each query (8315 unique queries) there was on average $\sim$ 7 graded ads, allowing us to evaluate ranking of ads in addition to relevance.

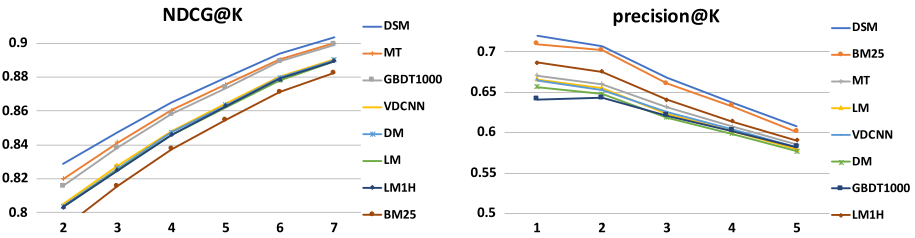
4.2.2 Baselines

We compared our method to traditional relevance models: Gradient boosted decision trees (with 1000 trees) (Zheng et al. 2008) ($GBDT_{1000}$), with 185 text-based features (Aiello et al. 2016) (trained on 700,000 editorial query-ad pairs) and the BM_{25} (Robertson and Walker 1994). We also use other CTR prediction task baselines (described in Sect. 4.1.2), where, as for the DSM, logits of the models were used as matching scores. Finally, we evaluated the search2vec (Grbovic et al. 2016) 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 (2167 unique queries coming from 8725 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 Fig. 6a.

For matching quality, we measure $precision@K$ and Normalized Discounted Cumulative Gain $NDCG@K$ (Wang et al. 2013) averaged across all queries.

4.2.3 Matching results

NDCG Relevance was assessed using $NDCG@K$ (Wang et al. 2013), and the results are given in Fig. 6a. We observe that the DSM approach improves over the alternatives (higher values of NDCG). Even though the difference is not obvious because



(a) NDCG@K **(b)** precision@K

Fig. 6 NDCG@K and precision@K for editorially judged 65 K query-ad pairs

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 NDCG@7 of the GBDT model by 2% and the best deep learning baseline MT by 0.5%.

Precision We also measure Precision@K to further characterize models, as shown in Fig. 6b. The DSM model is still the best performing model. However, for this metric the 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.

5 Final remarks

In this study we proposed a novel deep architecture that jointly learns CTR and semantic embeddings using click data. In addition, we proposed a novel cohort negative sampling technique for efficient learning of implicit negative signals. 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 learned on click data also outperformed other competitive algorithms on a query to ad matching task, as measured by the NDCG on the editorial dataset. Ablation study confirmed that the dual loss architecture statistically significantly enhanced 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. The above results suggest that joint training of two complementary tasks through deep supervision, query to ad matching and CTR prediction, yields high quality, versatile models.

However, the proposed method also has its limitations. Some of its disadvantages are directly inherited from the general framework of deep neural networks, like the need for large amounts of data, computation power. and training time. Although automatic learning of features comes with much less effort compared to manually crafted ones, it also comes at cost of lost interpretability. This would substantially increase required effort and time for debugging the model and explaining the results (good or bad) from model deployment. Other limitations stem from problem modeling. Namely, advertisements are served on multiple positions, and some positions are more likely

to be clicked due to sole convenience, and as we show in Fig. 4b, models tend to predict better at higher positions. In the proposed DSM model, we haven't explicitly accounted for such position bias, and in the future, we will aim to introduce such perspective into the prediction model, perhaps by using the "deep and wide" network architecture to inject position as an extra feature to the network.

Advances presented in this study, such as deeply supervised matching architecture and cohort-negative sampling optimization for large scale data, are applicable beyond the case of sponsored search. Essentially, the proposed methods can be utilized in any task where one needs to find a good match among the instances from two distinct sources of free text data. Prominent examples of such tasks are online recommender systems, where best match of product description and user's query should be found; professional networking services where one needs to match appropriate job opportunities and prospective employees based on requirements and skills in textual form; or online dating sites where users should be matched based on the textual descriptions of themselves.

Acknowledgements The authors gratefully thank to Lee Yang for his invaluable help in deploying our models on distributed GPU clusters, as well as Aleksandar Obradovic and Stefan Obradovic for proofreading and editing the language of the manuscript. The authors would like to thank the anonymous referees for their valuable comments and suggestions.

References

Aiello L, Arapakis I, Baeza-Yates R, Bai X, Barbieri N, Mantrach A, Silvestri F (2016) The role of relevance in sponsored search. In: 25th ACM international conference on information and knowledge management. ACM, pp 185–194

Baeza-Yates RA, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley Longman Publishing Co. Inc, Boston

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: International conference on learning representations

Bhamidipati N, Kant R, Mishra S (2017) A large scale prediction engine for app install clicks and conversions. In: Conference on information and knowledge management. ACM, pp 167–175

Cheng H, Cantú-Paz E (2010) Personalized click prediction in sponsored search. In: 3rd ACM international conference on web search and data mining. ACM, pp 351–360

Chen Y, Yan TW (2012) Position-normalized click prediction in search advertising. In: 18th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 795–803

Chen T, Sun Y, Shi Y, Hong L (2017) On sampling strategies for neural network-based collaborative filtering. In: 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 767–776

Conneau A, Schwenk H, Barrault L, Lecun Y (2017) Very deep convolutional networks for text classification. In: 15th Conference of the European chapter of the association for computational linguistics, pp 1107–1116

Edizel B, Mantrach A, Bai X (2017) Deep character-level click-through rate prediction for sponsored search. In: 40th ACM SIGIR international conference on research and development in information retrieval, pp 305–314

Fuxman A, Tsaparas P, Achan K, Agrawal R (2008) Using the wisdom of the crowds for keyword generation. In: 17th international conference on world wide web. ACM, pp 61–70

Gligorijevic D, Gligorijevic J, Raghuveer A, Grbovic M, Obradovic Z (2018a) Modeling mobile user actions for purchase recommendation using deep memory networks. In: The 41st international ACM SIGIR conference on research and development in information retrieval, pp 1021–1024

Gligorijevic D, Stojanovic J, Satz W, Stojkovic I, Schreyer K, Del Portal D, Obradovic Z (2018b) Deep attention model for triage of emergency department patients. In: SIAM international conference on data mining, pp 297–305

Graepel T, Candela JQ, Borchert T, Herbrich R (2010) Web-scale bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In: 27th international conference on machine learning, pp 13–20

Grbovic M, Djuric N, Radosavljevic V, Silvestri F, Bhamidipati N (2015) Context- and content-aware embeddings for query rewriting in sponsored search. In: International ACM SIGIR conference on research and development in information retrieval, pp 383–392

Grbovic M, Djuric N, Radosavljevic V, Silvestri F, Baeza-Yates R, Feng A, Ordentlich E, Yang L, Owens L (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, pp 375–384

Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2017) LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst 28(10):2222–2232

Guo J, Fan Y, Ai Q, Croft WB (2016) A deep relevance matching model for ad-hoc retrieval. In: 25th ACM international conference on information and knowledge management. ACM, pp 55–64

He X, Pan J, Jin O, Xu T, Liu B, Xu T, Shi Y, Atallah A, Herbrich R, Bowers S, et al (2014) Practical lessons from predicting clicks on ads at Facebook. In: 8th international workshop on data mining for online advertising. ACM, pp 1–9

Huang PS, He X, Gao J, Deng L, Acero A, Heck L (2013) Learning deep structured semantic models for web search using clickthrough data. In: 22nd ACM international conference on information and knowledge management. ACM, pp 2333–2338

Jaech A, Kamisetty H, Ringger E, Clarke C (2017) Match-tensor: a deep relevance model for search. arXiv preprint arXiv:1701.07795

Jiang Z (2016) Research on CTR prediction for contextual advertising based on deep architecture model. J Control Eng Appl Inform 18(1):11–19

Jones R, Rey B, Madani O, Greiner W (2006) Generating query substitutions. In: 15th international conference on world wide web. ACM, pp 387–396

Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd international conference on learning representations

Lee CY, Xie S, Gallagher P, Zhang Z, Tu Z (2015) Deeply-supervised nets. In: Artificial intelligence and statistics, pp 562–570

Li H, Xu J et al (2014) Semantic matching in search. Found Trends Inf Retr 7(5):343–469

Liu P, Qiu X, Huang X (2016) Deep multi-task learning with shared memory. In: Conference on empirical methods in natural language processing, pp 118–127

McMahan HB, Holt G, Sculley D, Young M, Ebner D, Grady J, Nie L, Phillips T, Davydov E, Golovin D, et al (2013) Ad click prediction: a view from the trenches. In: 19th ACM SIGKDD international conference on knowledge discovery and data mining

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013a) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 26:3111–3119

Mikolov T, Chen K, Corrado G, Dean J (2013b) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781

Mitra B, Diaz F, Craswell N (2017) Learning to match using local and distributed representations of text for web search. In: 26th international conference on world wide web. International World Wide Web Conferences Steering Committee, pp 1291–1299

Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Empirical methods in natural language processing, pp 1532–1543

Richardson M, Dominowska E, Ragno R (2007) Predicting clicks: estimating the click-through rate for new ads. In: 16th international conference on world wide web. ACM, pp 521–530

Robertson SE, Walker S (1994) Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In: International ACM SIGIR conference on research and development in information retrieval. Springer, New York, pp 232–241

Rumelhart DE, Hinton GE, Williams RJ et al (1988) Learning representations by back-propagating errors. Cognit Model 5(3):1

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Shan Y, Hoens TR, Jiao J, Wang H, Yu D, Mao J (2016) Deep crossing: web-scale modeling without manually crafted combinatorial features. In: 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 255–262

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A, et al (2015) Going deeper with convolutions. In: International conference on learning representations, pp 1–9

Wang Y, Wang L, Li Y, He D, Chen W, Liu TY (2013) A theoretical analysis of NDCG ranking measures, vol. 8. In: 26th annual conference on learning theory

Yan S, Lin W, Wu T, Xiao D, Zheng X, Wu B, Liu K (2018) Beyond keywords and relevance: a personalized ad retrieval framework in e-commerce sponsored search. In: 27th international conference on world wide web, pp 1919–1928

Zhai S, Chang Kh, Zhang R, Zhang ZM (2016) Deepintent: learning attentions for online advertising with recurrent neural networks. In: 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1295–1304

Zhang Y, Dai H, Xu C, Feng J, Wang T, Bian J, Wang B, Liu TY (2014) Sequential click prediction for sponsored search with recurrent neural networks. In: AAAI conference on artificial intelligence, pp 1369–1375

Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. In: Advances in neural information processing systems, pp 649–657

Zhang Y, Lee K, Lee H (2016) Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In: International conference on machine learning, pp 612–621

Zheng Z, Zha H, Zhang T, Chapelle O, Chen K, Sun G (2008) A general boosting method and its application to learning ranking functions for web search. In: Advances in neural information processing systems, pp 1697–1704

Affiliations

Jelena Gligorijevic[1] · Djordje Gligorijevic[1] · Ivan Stojkovic[1] · Xiao Bai[1] · Amit Goyal[2] · Zoran Obradovic[3]

✉ Zoran Obradovic
zoran.obradovic@temple.edu

Jelena Gligorijevic
jelenas@oath.com

Djordje Gligorijevic
djordje@oath.com

Ivan Stojkovic
ivans@oath.com

Xiao Bai
xbai@oath.com

Amit Goyal
a.goyal@criteo.com

1 Yahoo! Research, 701 First Ave, Sunnyvale, CA 94089, USA

2 Criteo, 325 Lytton Avenue, Palo Alto, CA 94301, USA

3 Computer and Information Sciences Department, Temple University, 1925 N 12th St, Philadelphia, PA 19122, USA