Similarity Models for Ad Relevance Measures

Wen-tau Yih
Microsoft Research
One Microsoft Way
Redmond, WA, USA
scottyih@microsoft.com

Ning Jiang
Microsoft AdCenter
One Microsoft Way
Redmond, WA, USA
ninjian@microsoft.com

Abstract

Measuring the relevance between the query and paid search ads is an important problem to ensure the overall positive search experience. In this paper, we study experimentally the effectiveness of various document similarity models based solely on the content analysis of the query and ad landing page. Our approaches focus on two different aspects that aim to improving the document representation: one produces a better term-weighting function and the other projects the raw term-vectors to the concept space. Both models are discriminatively trained and significantly outperform the baseline approach. When used for filtering irrelevant ads, combining these two models gives the most gain, where the uncaught bad ads rate has reduced 28.5% when the false-positive rate is 0.1.

1 Introduction

Paid search advertising is the main revenue source that supports modern commercial search engines. When a user issues a query to a search engine, the search result page consists the organic links, as well as short textual ads on the mainline and sidebar. Although from the user's perspective, both organic results and paid search ads should respond to the search intent and provide relevant information, their generation processes are very different. While presenting relevant Web pages should be the only objective of the retrieval algorithm behind organic search, ad selection is heavily influenced by the market behaviors of advertisers. Generally, advertisers create short textual ads with a list of bid keywords and specified matching schemes. Only ads with keywords matching the query have the chance to enter the auction process, which decides the final ads to show.

Although advertisers should select keywords that are highly relevant to their ads and landing pages, naively trusting the auction and matching mechanism could allow showing irrelevant ads due to different reasons. For example, inexact matching schemes (e.g., *phrase* and *broad* match) only partially match the bid keyword to the query, which may be semantically distant. Another example is that adversarial advertisers may game the system by bidding on lots of cheap but irrelevant keywords to increase the traffic to their sites with relatively low cost. To ensure satisfactory user experience, it is thus important to have an ad relevance judgment component in the ad selection pipeline. Such component can be used simply as an ad filter, which only allows the ads with high relevance scores entering the auction process. More ambitiously, a good ad relevance measure can be used to replace the cumbersome keyword–query matching scheme and increase the ad coverage by selecting more ads to participate in the auction.

Previous work on ad relevance follows the vector space model paradigm in information retrieval and focuses on constructing suitable term-vectors representing the query and ad-text. For example, Broder et al. [2] leverage the search engine and use the pseudo relevance feedback technique to expand queries. Choi et al. [3] further enhance such approach by incorporating content from the ad landing page when creating term-vectors for the ads. The relevance function of a pair of query and ad is simply the cosine similarity score of the two corresponding vectors. The main advantage

of this vector space approach is its computational efficiency when handling large-scale data in real-time. For instance, ad selection beyond keyword matching can be done via inverted index, pre-built offline on query and ad vectors. The content-based similarity score can also be combined later with other signals (e.g., advertiser reputation or user click features) to evaluate the relevance of ads from a pre-selected and much smaller candidate set.

In this paper, we aim to *learn* a better vector representation of queries and ads from simple content analysis so that a pre-selected similarity function (e.g., *cosine*) can become a reliable relevance measure. Instead of exploring various unsupervised heuristics of term weighting and selection in previous work, we exploit the annotated pairs of queries and ads and adapt two recently proposed vector learning approaches. The first approach, TWEAK [10], provides a simple means to incorporate various term-level features and results in a much better term-weighting function compared to using fixed formulas like TFIDF. The second approach, *S2Net* [11], maps the original term-vector representation to a much smaller but dense concept vector space, which allows matching of semantically related words, and improves the relevance measure even after dimensionality reduction. While both approaches significantly outperform the TFIDF cosine baseline in our experiments on real-world data, the best result comes from the combination of these two, which reduces the false-negative rate by 28.5% when the false-positive rate is 0.1 in an ad filtering scenario.

The rest of this paper is organized as follows. Sec. 2 gives the problem definition and describes our approaches in detail. Our experimental validation is provided in Sec. 3. Finally, Sec. 4 presents some related work and Sec. 5 concludes the paper.

2 Problem & Approaches

Informally, the problem we are trying to solve can be described as follows. Assume that we are given a set of query–ad pairs with human relevance judgement as training data. The goal here is to learn a function that maps the query and ad to vectors, so that their similarity score (*cosine* in this paper) can be used as a good relevance measure – e.g., the score of an "excellent" query–ad pair will be higher than the "bad" ones. Because both queries and ads contain only a few words and may not provide enough content, we expand them first to "documents" as the raw input. On the query side, we applied the same query expansion method described in [8, 2]. Each query in our data set was first issued to the Bing search engine. The top 100 search result snippets are concatenated to form a corresponding "pseudo-document". On the ad side, we used its landing page. Taking advantage of the human judgement labels, we experiment with two new discriminative approaches to learn the vector representation from documents. One can be viewed as an enhanced term-vector generation process and the other produces a low-rank representation.

2.1 Learning Enhanced Term Vectors

The most widely used document representation for similarity measures is arguably the bag-of-words term-vectors. Suppose $\mathcal{V} = \{t_1, t_2, \cdots, t_n\}$ is the pre-defined vocabulary that consists of a set of all possible terms (e.g., tokens, words) that may occur in each document. The vector that represents a document d is then $[s_1, s_2, \cdots, s_n]^T$, where s_i is the weight of term t_i. Such vectors are typically very sparse as s_i is set to 0 when t_i does not occur in d. Otherwise, s_i is often determined by some fixed weighting formula such as TFIDF (e.g., $s_i = tf(t_i, d) \cdot \log(N/df(t_i))$, where N is the number of documents in the corpus). Because the term weights dictate the quality of the similarity function operating on the term-vectors, here we adapt TWEAK [10] to learn a better weighting function by incorporating more term-level information using our labeled data.

Suppose that each term t_i from document d is associated with a short feature vector, $(\phi_1(t_i, d), \phi_2(t_i, d), \cdots, \phi_m(t_i, d))$, where ϕ_j is the function that captures some term-level information, such as its position in the document or whether it is capitalized. The new term-weighting function is a linear combination of these features, namely $s'_i = tw(t_i, d) \equiv \sum_{j=1}^{m} \lambda_j \phi_j(t_i, d)$, where λ's are the model parameters. Because the human relevance judgement labels are defined on pairs of queries and documents, we cannot use the non-existent "correct" term-weighting scores to train the model. Instead, the difference between the label and similarity score based on the current model will be back-propagated to tune the parameters in each training iteration. More detail on the loss function and training procedure will be described in Sec. 2.3.

2.2 Learning Concept Vectors

When applying an inner-product like similarity functions such as cosine or the Jaccard coefficient, one major weakness of the term-vector representation is that different terms, regardless of how semantically related they are, will not be matched. As an illustrative example, two semantically very close term vectors {buy:0.3, pre-owned:0.5, car: 0.4} and {purchase:0.4, used:0.3, automobile:0.2} will have 0 cosine score because they do not contain any identical terms. Obviously, having a better term-weighting function will not fix this issue as long as the choice of *active* terms remains the same.

A common solution to this problem is to map the original term-vectors to some "concept space", so that semantically close words will be captured by the same concept [6, 7, 1]. Instead of using a generative model, we aim to learn the projection matrix discriminatively by adapting a newly proposed Siamese neural network model, S2Net [11]. Formally, given a set of term-vectors representing queries or ads, we would like to learn a matrix $\mathbf{A}_{n\times k}$ so that a term-vector $\mathbf{s} = [s_1, s_2, \cdots, s_n]^T$ will be mapped to a k-dimensional vector $\mathbf{A}^T\mathbf{s}$, where $k \ll n$. In this low-rank representation, the association between each concept element and the original terms is described in the corresponding column of $\mathbf{A}$. Although the functional form of this model, the matrix $\mathbf{A}$, is identical to the projection matrix used in latent semantic analysis (LSA) [6], the model generation process is completely different. By tuning the model (i.e., all the $k \cdot n$ entries in $\mathbf{A}$) to have a better similarity score from the concept vectors, such dimensionality reduction technique can in fact increase the performance significantly.

2.3 Model Training

Suppose $\mathbf{s}_q$ and $\mathbf{s}_a$ are the concept or term vectors of the query–ad pair (q, a). Then the cosine similarity score is $sim_{\mathbf{\Lambda}}(q, a) \equiv \frac{\mathbf{s}_q \cdot \mathbf{s}_a}{||\mathbf{s}_q|| \cdot ||\mathbf{s}_a||}$, where $\mathbf{\Lambda}$ denotes the model parameters (i.e., λ's in TWEAK and $\mathbf{A}$ in S2Net). Assume that we map the human judgement label of (q, a) to $y \in [-1, 1]$, then one simple loss function we can use is the mean-squared error, defined as $L_{\mathrm{MSE}}(y, sim_{\mathbf{\Lambda}}(q, a)) = \frac{1}{2}(y - sim_{\mathbf{\Lambda}}(q, a))^2$.

However, in either the ad filtering or ranking scenarios, the ranking order of the ads or query–ad pairs is usually more important than the absolute score. We therefore use a pairwise loss function for training the models instead. Given two pairs (q_1, a_1) and (q_2, a_2) where the former is annotated as more relevant than the latter, let Δ be the difference of their similarity scores. Namely, $\Delta = sim_{\mathbf{\Lambda}}(q_1, a_1) - sim_{\mathbf{\Lambda}}(q_2, a_2)$. The loss function L we use is: $L(\Delta) = \log(1 + \exp(-\gamma\Delta))$, where γ is the scaling factor that magnifies Δ from $[-2, 2]$ to a larger range. This loss function can be shown to upper bound the pairwise accuracy. It can also be regularized further by adding a term like $\frac{\alpha}{2}\sum_j \lambda_j^2$ or $\frac{\alpha}{2}||\mathbf{A}||^2$. In the experiments in this paper, γ is set to 10 and α is 0.01 for TWEAK. S2Net is regularized using a simple early stop scheme tuned based on the validation set. Optimizing the model parameters can be done using gradient based methods, such as stochastic gradient decent or L-BFGS.

3 Experiments

In this section, we present our experiments of applying the above similarity learning models in the ad relevance problem, including the data collection process, tasks and evaluation metrics, as well as the detailed results.

3.1 Data & Tasks

The similarity models discussed above are evaluated on a proprietary dataset made available by Microsoft AdCenter. The dataset consists of 12,481 unique queries that were randomly sampled from the Bing search engine logs. For each query, a number of top ads ranked according to their monetization values or cost per impression (CPI) are selected. Ads with higher CPI have higher chance to be shown to search users, and thus are more sensitive to classification error. Ads with invalid or unreachable landing pages are removed from the dataset. This results in a total number of 681,100 query-ad pairs in the dataset, of which 446,781 unique pairs of queries and landing pages are found. Note that duplicates exist because multiple ads may point to the same landing page.

Feature	Remark
TF	term-frequency
DF	document-frequency
Loc	the position of the first occurrence of the term
Len	the length of the document
QF	query log frequency
IsCap	whether any occurrence of the term is capitalized
InQry	whether the term is part of the query
InUrl	whether the term is part of the URL
InAnchorText	whether the term is in an anchor text
InHtmlTitle	whether the term occurs in the title
InMetaDescription	whether the term appears in the meta-description section
InMetaKeywords	whether the term appears in the meta-keywords section
Emphasized	whether the term is emphasized in some special fonts

Table 1: Term-level features used in the TWEAK model. The QF feature is the number of times the term is seen as a query in search logs during an 18-month period. The logarithmic values of TF, DF, Len and QF are also used as features. InQry is only used for the query side. InUrl, InAnchorText, InHtmlTitle, InMetaDescription, InMetaKeywords and Emphasized are only used on ad landing pages.

Each query-ad pair is then manually labeled using a scheme that describes the relationship between concepts (or sets). In this scheme, each document is regarded as a concept, and the relationship between two concepts is one of the five relations, namely *same*, *subset*, *superset*, *overlap*, or *disjoint*. In our experiment, when the task is a binary classification problem, pairs labeled as same, subset, or superset (23% of the dataset) are considered relevant, pairs labeled as disjoint (60% of the dataset) are considered irrelevant, and others (17% of the dataset) are ignored. When pairwise comparisons are needed in either training or evaluation, the relevance order is *same* $>$ *subset* $=$ *superset* $>$ *disjoint*.

The dataset is split into training, validation and test sets by queries. Ads selected for a particular query always appear in only one of these three sets. This avoids the same query-landing page pair being used in both training and validation/test sets, which would happen if one randomly splits query-ad pairs in the dataset. In our experiments, 40% of the queries are reserved for training, 30% for validation and the remaining 30% for testing. The validation set is used to tune some hyper-parameters such as the number of training iterations and the weight of the regularization term.

As mentioned in Sec. 2, we use the search snippets to form "pseudo-documents" for queries and the landing pages for ads as the raw input documents. Our vocabulary set contains 29,854 words and is determined using a document frequency table derived from a large web corpus. Only words with counts larger than a pre-selected threshold are retained. When applying the TWEAK model to learn the term-weighting function, the features we used are summarized in Tab. 1.

We test our models in two different application scenarios. The first is to use the ad relevance measure as an *ad filter*. When the relevance score of an ad is below a pre-selected decision threshold, this ad is considered not relevant to the query and will be filtered before going to the auction process. As this setting is close to the typical anomaly detection problem, we present the Receiver Operating Characteristic (ROC) curves of tested models to show the trade-off between false-positive (i.e., mistakenly removed good ads) and true-positive (i.e., filtered bad ads), as well as the corresponding AUC scores as the evaluation metrics. The second one is the commonly ranking scenario as used in organic search, where the ads with keywords that match the query are purely selected and ranked by their relevance score. In this scenario, we use the standard NDCG scores as the evaluation metric.

3.2 Results

We compare four different configurations in our experiments. Served as our baseline, *TFIDF* is the basic term-vector representation with the TFIDF weighting ($tf \cdot \log(N/df)$). TWEAK has exactly the same terms in each TFIDF vector, but the weights are determined by the linear function of features in Tab. 1 with model parameters learned from the training data. Taking these two different term-

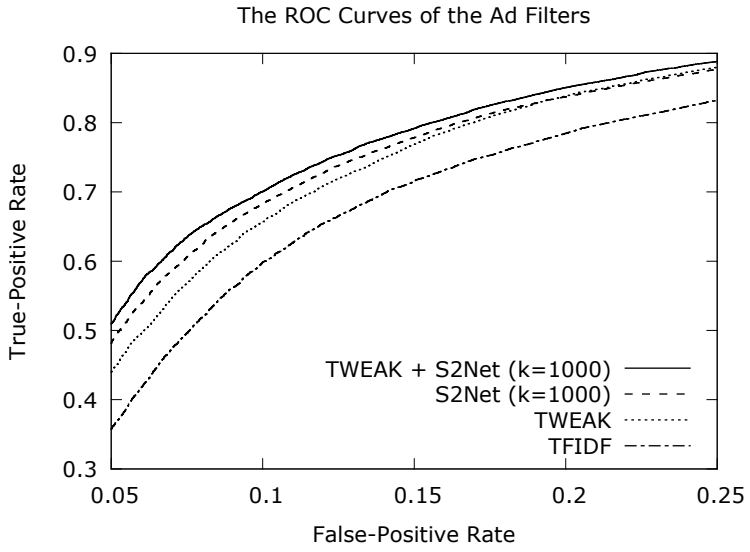

Figure 1: The ROC curves of four different vector representations when the corresponding cosine scores are used as ad filters. False-positive rate indicates the percentage of mistakenly filtered relevant ads and true-positive rate is the percentage of successfully filtered irrelevant ads.

	TFIDF	TWEAK	S2Net	TWEAK + S2Net
AUC	0.861	0.888	0.892	0.898
NDCG@1	0.825	0.859	0.855	0.870
NDCG@3	0.854	0.883	0.883	0.893
NDCG@5	0.876	0.899	0.901	0.909

Table 2: The AUC and NDCG scores of the cosine similarity scores on different vector representations. The dimensionality parameter k is 1000 for S2Net. Except for the NDCG scores between TWEAK and S2Net, the differences between any two methods in various metrics are statistically significant.

vectors as input, we applied S2Net to learn the projection matrices to map them to a k-dimensional space, denoted as S2Net and TWEAK+S2Net, respectively.

When the cosine scores of these vector representations are used as ad filters, their ROC curves (focusing on the low false-positive region) are shown in Fig. 1. It can be clearly observed that the similarity scores computed based on the learned vectors indeed have better quality, compared to the raw TFIDF representation. Among them, TWEAK and S2Net perform quite similarly, where S2Net has slight advantage when the false-positive rate is below 0.15. Not surprisingly, our best result comes from the combination of these two models. At the 0.1 false-positive rate point, TWEAK+S2Net can filter 28.5% more irrelevant ads compared with TFIDF.

Similar trend is also reflected on the AUC scores and NDCG numbers, presented in Tab. 2. S2Net has a higher AUC score compared to TWEAK, but is inferior in NDCG@1 and NDCG@3. TWEAK+S2Net is a clear winning approach, and has higher scores in both AUC and NDCG. Again, all the learning models result in stronger similarity scores than simply using TFIDF term vectors. All comparisons except for the NDCG scores between TWEAK and S2Net are statistically significant. For AUC, we randomly split the data into 50 subsets and ran a paired-t test between the corresponding AUC scores of two methods. For NDCG scores, we compared the DCG scores per query of the compared models using the paired-t test. The difference is considered statistically significant when the p-value is less than 0.01 after the Bonferroni correction.

Although for efficiency reason, ideally we would like the dimensionality of the projected concept vectors as small as possible. However, the quality of such representation usually degrades as well.

	TFIDF	PCA_{1000}	S2Net_{100}	S2Net_{300}	S2Net_{500}	S2Net_{750}	S2Net_{1000}
AUC	0.861	0.848	0.855	0.879	0.880	0.888	0.892
NDCG@1	0.825	0.815	0.843	0.852	0.856	0.860	0.855
NDCG@3	0.854	0.847	0.871	0.879	0.881	0.884	0.883
NDCG@5	0.876	0.870	0.890	0.897	0.899	0.902	0.901

Table 3: The AUC and NDCG scores of S2Net at different k (dimensionality) values. TFIDF and PCA ($k = 1000$) are used as baselines. The differences in AUC for any two methods, except for S2Net_{300} and S2Net_{500}, are statistically significant. For the NDCG scores, all S2Net models outperform TFIDF and PCA statistically significantly. The differences among S2Net_{300}, S2Net_{500}, S2Net_{750} and S2Net_{1000} are not statistically significant.

It is thus interesting to know the best trade-off point between these two variables. We conduct this study by using the raw TFIDF term-vectors as input for the S2Net model with various k values, and the results in terms of AUC and NDCG are shown in Tab. 3. In addition, we also compared the results with the commonly used dimensionality reduction technique, PCA. As can be found in the table, the performance of S2Net easily surpasses TFIDF when $k = 300$. As k increases, the performance improves quite consistently as well. Notice that even with $k = 1000$, PCA is not doing better than S2Net ($k = 100$), which uses only one tenth of the space, and is still inferior to TFIDF.

4 Related Work

Our approach on learning vector representation for similarity measures is very related to the work of distance metric learning [5]. As the computational complexity of learning a complete Mahalanobis matrix is at least $O(n^2)$, where n is the vocabulary size, directly applying them to the problems in the text domain is not practical. Although learning a low-rank matrix has been suggested [4, 9], our previous study has shown that the TWEAK/S2Net approach can perform better [11]. On the content analysis side, Choi et al. [3] used the cosine score to judge the ad relevance, but applied document summarization techniques to identify important portions in the ad landing page to construct the vector. Such information can easily be incorporated in our model and could potentially improve the performance.

5 Conclusions

In this paper, we explore the effectiveness of two recently proposed similarity models, TWEAK and S2Net, for measuring paid-search ad relevance. Both approaches aim to learn new vector representations of documents to improve the quality of the target similarity score (e.g., cosine) operating on the vectors. When used in the scenarios of ad filtering and ranking as relevance measures, the learned vector representations lead to significantly better results compared to the typical TFIDF term-vector construction. As the two approaches focus on different aspects and are complementary to each other, we found that combining these two methods produces the most performance gain.

The promising results from this initial experimental study trigger several interesting research direction for the future work. For example, the current combination approach treats the TWEAK and S2Net models separately and chains them in a sequential fashion. Training these two sets of model parameters could be a more natural approach to further enhance the overall model performance. On the feature side, improving the relevance measure by incorporating more information in the model, such as ad-text, advertiser reputation and deeper query and landing page analysis is also on our agenda.

References

[1] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Andrei Z. Broder, Peter Ciccolo, Marcus Fontoura, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. Search advertising using web relevance feedback. In *CIKM*, pages 1013–1022, 2008.

[3] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. Mediano, and B. Pang. Using landing pages for sponsored search ad selection. In *Proceedings of the 19th World Wide Web Conference*, 2010.

[4] Jason V. Davis and Inderjit S. Dhillon. Structured metric learning for high dimensional problems. In *KDD*, pages 195–203, 2008.

[5] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 209–216, New York, NY, USA, 2007. ACM.

[6] Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99*, pages 50–57, 1999.

[8] Mehran Sahami and Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th World Wide Web Conference*, 2006.

[9] L. Wu, R. Jin, S. Hoi, J. Zhu, and N. Yu. Learning bregman distance functions and its application for semi-supervised clustering. In *Advances in Neural Information Processing Systems 22*, pages 2089–2097. 2009.

[10] W. Yih. Learning term-weighting functions for similarity measures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, 2009.

[11] W. Yih and C. Meek. Learning vector representations for similarity measures. Technical Report MSR-TR-2010-139, Microsoft Research, October 2010.