Query Auto-Completion for Rare Prefixes

Bhaskar Mitra
Microsoft
bmitra@microsoft.com

Nick Craswell
Microsoft
nickcr@microsoft.com

ABSTRACT

Query auto-completion (QAC) systems typically suggest queries that have previously been observed in search logs. Given a partial user query, the system looks up this query prefix against a precomputed set of candidates, then orders them using ranking signals such as popularity. Such systems can only recommend queries for prefixes that have been previously seen by the search engine with adequate frequency. They fail to recommend if the prefix is sufficiently rare such that it has no matches in the precomputed candidate set.

We propose a design of a QAC system that can suggest completions for rare query prefixes. In particular, we describe a candidate generation approach using frequently observed query suffixes mined from historical search logs. We then describe a supervised model for ranking these synthetic suggestions alongside the traditional full-query candidates. We further explore ranking signals that are appropriate for both types of candidates based on n-gram statistics and a convolutional latent semantic model (CLSM). Within our supervised framework the new features demonstrate significant improvements in performance over the popularity-based baseline. The synthetic query suggestions complement the existing popularity-based approach, helping users formulate rare queries.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval

Keywords: Query auto-completion; Deep learning

1. INTRODUCTION

As users enter their query into the search box, most modern search engines provide a ranked list of query suggestions based on the current prefix already typed by the user. In a typical approach used by many query auto-completion (QAC) systems, candidate queries are identified by doing an exact prefix lookup against a fixed set of popular queries, using a data structure such as a prefix tree [4]. The candidates are then ranked by their expected likelihood, which is typically computed as a function of its past popularity (commonly referred to as the *MostPopularCompletion* (MPC) model [1]). Such a system can only suggest queries with enough historic popularity to make it into the prefix tree.

CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org//10.1145/2806416.2806599.

Table 1: Synthetic QAC candidates generated by the suffix-based approach and ranked using only the CLSM similarity feature. The CLSM model projects both the prefix and the suffix to a common 128-dimensional space allowing us to rank according to prefix-suffix cosine similarity. One of the lower quality synthetic candidates "cheapest flights from seattle to airport" is ranked seventh in the second list.

what to cook with chicken and broccoli and
what to cook with chicken and broccoli *and bacon*
what to cook with chicken and broccoli *and noodles*
what to cook with chicken and broccoli *and brown sugar*
what to cook with chicken and broccoli *and garlic*
what to cook with chicken and broccoli *and orange juice*
what to cook with chicken and broccoli *and beans*
what to cook with chicken and broccoli *and onions*
what to cook with chicken and broccoli *and ham soup*

cheapest flights from seattle to
cheapest flights from seattle *to dc*
cheapest flights from seattle *to washington dc*
cheapest flights from seattle *to bermuda*
cheapest flights from seattle *to bahamas*
cheapest flights from seattle *to aruba*
cheapest flights from seattle *to punta cana*
cheapest flights from seattle *to airport*
cheapest flights from seattle *to miami*

We propose an additional candidate generation strategy for QAC by mining popular query *suffixes*. Candidate suffixes are popular n-grams that appear at the ends of queries. By appending such n-grams suffixes to a user's query prefix we can generate synthetic suggestion candidates that have never been observed in the historical query logs. Table 1 contains examples of such suggestions. We further propose a supervised framework for ranking these synthetic queries alongside the traditional full-query suggestion candidates. We also explore new ranking signals in this framework, based on the query n-gram statistics and a deep *convolutional latent semantic model* (CLSM)[15].

2. RELATED WORK

Language modelling based approaches for *sentence completion* have been studied in the context of e-mail and document authoring[3, 5, 8, 13]. In Web search, White and Marchionini [17] and Fan et al. [7] proposed models for term recommendations to aid users in their query formulation process. Bhatia et al. [2] extracted frequently

Table 2: Most popular query suffixes extracted from the publicly available AOL logs.

Top suffixes	Top 2-word suffixes	Top 3-word suffixes
com	for sale	federal credit union
org	yahoo com	new york city
net	myspace com	in new york
gov	google com	or no deal
pictures	new york	disney channel com
lyrics	real estate	my space com
edu	of america	in new jersey
sale	high school	homes for sale
games	new jersey	department of corrections
florida	space com	chamber of commerce
for sale	aol com	bath and beyond
us	s com	in las vegas

occurring phrases from document corpus and used them to generate suggestion candidates in the absence of a query log. Duan and Hsu [6] have studied the problem of online spelling correction for query auto-completion and Hawking and Griffiths [9] have explored mechanisms for generating query suggestions in the enterprise settings. Our proposed approach generates synthetic query suggestion candidates by combining the input prefix with popular query *suffixes* to augment the regular full-query QAC suggestions.

Within our proposed supervised framework, we explore *convolutional latent semantic model* (CLSM)[10, 15] as a ranking signal. Mitra [11] previously used the CLSM for modelling session context for QAC ranking. Unlike Mitra [11], our focus is on ranking query suffixes and we propose a novel approach for training a CLSM on prefix-suffix pairs for this task.

3. RARE PREFIX AUTO-COMPLETION

We propose two key ideas in this paper. Firstly, we generate synthetic query suggestion candidates for QAC using popular query *suffixes*. We then introduce n-gram and CLSM based features in a supervised learning setting to rank these synthetic suggestions alongside the full-query suggestion candidates.

3.1 Candidate Generation

From every query in the search engine logs we generate all possible n-grams from the end of the query. For example, from the query "bank of america" we generate the suffixes "america", "of america" and "bank of america". By aggregating across all queries we identify the most popular suffixes. Table 2 shows the most frequently observed query suffixes in the publicly available AOL logs [14].

Next, for a given prefix we extract the *end-term* as shown in Figure 1. We match all the suffixes that start with the end-term from our precomputed set. These selected suffixes are appended to the prefix to generate synthetic suggestion candidates. For example, the prefix "cheap flights fro" is matched with the suffix "from seattle" to generate the candidate "cheap flights from seattle". Note that many of these synthetic suggestion candidates are likely to not have been observed by the search engine before.

We merge these synthetic suggestions with the set of candidates selected from the list of historically popular queries. This combined set of candidates is used for ranking as we will describe in Sec 4.

3.2 Ranking Features

For every prefix and suggestion candidate (synthetic or previously observed), we compute a set of common features for the supervised ranking model. We describe these features in this section, focusing on the n-gram and CLSM features that we propose for this setting.

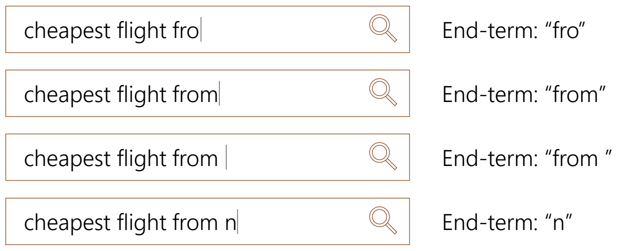

Figure 1: Examples of fully or partially typed end-terms extracted from the input prefixes. The end-term is used for selecting the set of candidate suffixes for the generation of synthetic query suggestions.

N-gram based features. From the set of all n-grams $\mathcal{G}$ in a candidate suggestion we compute the n-gram frequency features $ngramfreq_i$ (for i = 1 to 6).

$$ngramfreq_i = \sum_{g \in \mathcal{G}, len(g)=i} freq(g) \qquad (1)$$

Where $len(g)$ and $freq(g)$ are the number of words in the n-gram g and its observed frequency in the historical query logs, respectively. Intuitively, these n-gram features model the likelihood that the candidate suggestion is generated by the same language model as the queries in the search logs.

CLSM based features. For document retrieval, Shen et al. [15] demonstrated that discriminatively training a deep neural network model with a convolutional-pooling structure on clickthrough data can be effective for modelling query-document relevance. We adopt the CLSM by training on a *prefix-suffix* pairs dataset (instead of query-document titles). The training data for the CLSM is generated by sampling queries from the search logs and splitting each query at every possible word boundary. For example, from the query "breaking bad cast" we generate the two pairs ("breaking", "bad cast") and ("breaking bad", "cast"). The architecture shown in Figure 2 is used on both the prefix and the suffix side of the CLSM model.

Now given a prefix $\mathcal{P}$ and a suggestion candidate $\mathcal{C}$, we extract a normalized prefix $\bar{p}$ by removing the *end-term* from the prefix. Then a normalized suffix $\bar{s}$ is extracted by removing $\bar{p}$ from the query $\mathcal{C}$. Then we use the trained CLSM model to project the normalized prefix and the normalized suffix to a common 128-dimensional space and compute a $clsmsim$ feature.

$$clsmsim(\bar{p}, \bar{s}) = cosine(y_1, y_2) = \frac{y_1^\intercal y_2}{\|y_1\| \|y_2\|} \qquad (2)$$

where y_1 and y_2 are the CLSM vector outputs corresponding to $\bar{p}$ and $\bar{s}$, respectively. Table 1 shows examples of synthetic suggestion candidates ranked by this *clsmsim* feature alone.

Other features. Other features used in our model includes the frequency of the candidate query in the historical logs, length based features (length of the prefix, the suffix and the full suggestion in both characters and words) and a boolean feature that indicates whether the prefix ends with a space character.

4. EXPERIMENTS

Our experiment setup is based on the *learning to rank* framework proposed by Shokouhi [16]. We generate all possible prefixes[1] from each query impression to use for training, validation and testing. For each prefix we identify the set of candidate suggestions as described

[1]Mitra et al. [12] showed that users use QAC more at word boundaries but for simplicity we sample the prefixes with equal probability.

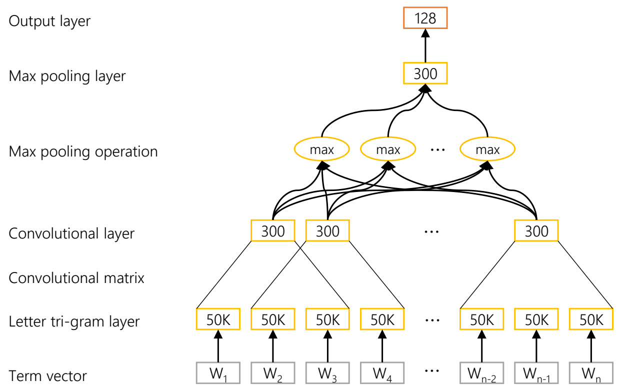

Figure 2: The CLSM model architecture. The model has a convolutional-pooling structure and a 128-dimensional output.

in Section 3.1. We associate a positive relevance judgment with the candidate that matches the original query from which the prefix was extracted. Unlike Mitra [11], to accurately measure the coverage impact of our approach we retain all prefix impressions where the submitted query is not in the list of candidates available for ranking.

We train LambdaMART [18] models for ranking the suggestions using features described in Section 3.2. We limit our ranking task to instances where the prefix contains at least one complete word, since completions with very short prefixes is already well solved by our popularity-based features and we are focusing on rare prefixes. We always train 300 trees (with early stopping using a validation set) and evaluate the model performances on the test set using the *mean reciprocal rank* (MRR) metric.

We conduct all our experiments on the publicly available AOL query logs [14] and reproduce the same results on the large-scale query logs of the Bing search engine. We refer to these two datasets hereafter as the *AOL testbed* and the *Bing testbed*, respectively.

The query impressions on both the testbeds are divided into four temporally separate partitions (background, training, validation and test). On the *AOL testbed* we use all the data from 1 March, 2006 to 30 April, 2006 as the *background* data. We sample queries from the next two weeks for *training*, and from each of the following two weeks for *validation* and *test*, respectively. On the *Bing testbed* we sample data from the logs from April, 2015 and use the first week of data for background, the second week for training, the third for validation and the fourth for testing. We normalize all the queries in each of these datasets by removing any punctuation characters and converting them to lower case.

For candidate generation, both the list of popular queries and suffixes are mined from the *background* portion of the two testbeds. We use 724,340 and 1,040,674 distinct queries on the AOL testbed and the Bing testbed, respectively, as the set of full-query candidates. We evaluate our approach using 10K and 100K most frequent suffixes. We limit the number of full-query candidates per prefix to ten and compute the final reciprocal rank by considering only the top eight ranked suggestions per model. Finally, the CLSM models are trained using 44,558,631 and 212,854,198 prefix-suffix pairs on the AOL and the Bing testbeds, respectively.

5. RESULTS

Table 3 summarizes the experiment results and clearly demonstrates the improvements from the synthetic suggestion over the MPC model. All the LambdaMART models with different feature sets when combined with the suffix-based candidates show an improved MRR over the popularity based baseline. The models however perform no better, and in most cases worse, compared to

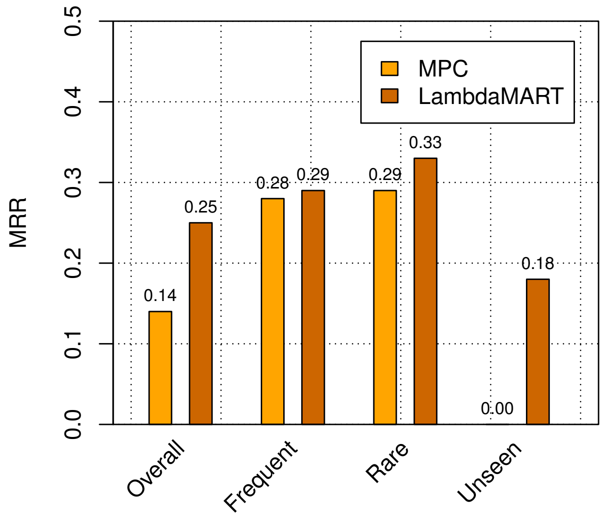

Figure 3: MRR improvements by historical popularity of the input prefix on the AOL testbed. The LambdaMART model uses n-gram and CLSM features and includes suffix-based suggestion candidates. Any prefix in the top 100K most popular prefixes from the background data is considered as *Frequent*. There are 7622, 6917 and 14,135 prefix impressions in the *Frequent*, *Rare* and *Unseen* segments, respectively. All reported differences in MRR with the MPC model are statistically significant by the t-test ($p < 0.01$).

the MPC baseline when only the full-query based candidates are considered. This is expected as the models are trained with the suffix-based candidates in the training data.

Figure 3 analyses the improvements by segmenting the prefixes based on their historical popularity. The improvements from the suffix-based candidates are expectedly higher for the rarer prefixes. Interestingly, the absolute MRR values for both models are higher for rare prefixes than for the frequent ones. One factor in this is that rare prefixes tend to be longer and therefore more specific, giving fewer candidates to rank and making it easier to achieve good MRR.

The models with the *clsmsim* feature perform better than the corresponding models without the feature across all experiments. However, in general the n-gram features seems to be showing higher improvements compared to the CLSM based feature. We hypothesize that the CLSM feature is less precise than the n-gram features. For example, we can see in Table 1 that the CLSM based feature ranks a suffix highly that generates a semantically meaningless query suggestion "cheapest flight from seattle to airport". While "airport" is a location that you can take a flight to, in the context of the given prefix it is clearly an inappropriate suggestion. It is possible that the prefix-suffix pairs based training of the CLSM can be further improved. We believe that this is an important area for future investigations given that the CLSM holds certain other advantages over n-gram models. For example, the CLSM has limited storage requirements[2], and because of the *word hashing* technique the CLSM may be more robust to morphological variations and spelling errors in the input prefix compared to the n-gram based models.

6. CONCLUSION

We proposed a novel candidate generation technique for query auto-completion by mining and ranking popular query *suffixes*. Our empirical study shows that this is an effective strategy for significantly improving MRR for rare and unseen prefixes. The supervised

[2]The CLSM model itself needs to be stored in memory but has no data storage requirements, unlike the n-gram models.

Table 3: Comparison of all models on the AOL and the Bing testbeds. Due to the proprietary nature of the Bing dataset, we only report MRR improvements relative to the MPC model for this testbed. Statistically significant differences by the t-test ($p < 0.01$) are marked with "*". Top three highest MRR values per testbed are bolded.

Models	AOL		Bing
	MRR	% Improv.	% Improv.
Full-query based candidates only			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.1445	-0.1	-1.7*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.1427	-1.4*	-1.2*
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.1445	-0.1	-1.2*
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.1432	-1.0*	-1.5*
Full-query based candidates + Suffix based candidates (Top 10K suffixes)			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.2116	+46.3*	+32.8*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.2326	+60.8*	+42.6*
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.2249	+55.5*	+40.1*
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.2339	**+61.7***	+43.8*
Full-query based candidates + Suffix based candidates (Top 100K suffixes)			
MostPopularCompletion	0.1446	-	-
LambdaMART Model (n-gram features = no, CLSM feature = no)	0.2105	+45.5*	+39.9*
LambdaMART Model (n-gram features = yes, CLSM feature = no)	0.2441	**+68.7***	**+54.2***
LambdaMART Model (n-gram features = no, CLSM feature = yes)	0.2248	+55.4*	**+48.9***
LambdaMART Model (n-gram features = yes, CLSM feature = yes)	0.2453	**+69.6***	**+55.3***

ranking framework proposed in this paper is generic and can be employed in any QAC system that combines multiple sources of candidates. We described features based on n-gram language models and convolutional neural networks with demonstrable improvements.

While we have shown significant improvements in MRR using synthetic candidate generation, we have not measured how often this approach generates semantically meaningless synthetic suggestions and have not quantified the effect of showing synthetic suggestions to search users. A user study on this aspect is left as future work. There is also further scope for exploring other language models (such as recurrent neural networks) in the context of this task.

Acknowledgements. The authors are grateful to Piotr Mirowski and Milad Shokouhi for the valuable discussions and their insightful feedback during the early stages of this work.

References

[1] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proc. WWW*, pages 107–116, 2011.

[2] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *Proc. SIGIR*, pages 795–804, 2011.

[3] S. Bickel, P. Haider, and T. Scheffer. Learning to complete sentences. In *Proc. ECML*, pages 497–504. Springer, 2005.

[4] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *Proc. SIGMOD*, pages 707–718, 2009.

[5] J. J. Darragh, I. H. Witten, and M. L. James. The reactive keyboard: A predictive typing aid. *Computer*, 23:41–49, November 1990.

[6] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*, pages 117–126, 2011.

[7] J. Fan, H. Wu, G. Li, and L. Zhou. Suggesting topic-based query terms as you type. In *Proc. APWEB*, pages 61–67, 2010.

[8] K. Grabski and T. Scheffer. Sentence completion. In *Proc. SIGIR*, pages 433–439, 2004.

[9] D. Hawking and K. Griffiths. An enterprise search paradigm based on extended query auto-completion. do we still need search and navigation? In *Proc. ADCS*, pages 18–25, 2013.

[10] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proc. CIKM*, pages 2333–2338. ACM, 2013.

[11] B. Mitra. Exploring session context using distributed representations of queries and reformulations. In *Proc. SIGIR*, To appear, 2015.

[12] B. Mitra, M. Shokouhi, F. Radlinski, and K. Hofmann. On user interactions with query auto-completion. In *Proc. SIGIR*, pages 1055–1058, 2014.

[13] A. Nandi and H. V. Jagadish. Effective phrase prediction. In *Proc. VLDB*, pages 219–230, Vienna, Austria, 2007.

[14] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proc. InfoScale*. ACM, 2006. ISBN 1-59593-428-6.

[15] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proc. WWW*, pages 373–374, 2014.

[16] M. Shokouhi. Learning to personalize query auto-completion. In *Proc. SIGIR*, pages 103–112, 2013.

[17] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Inf. Process. Manage.*, 43: 685–704, May 2007.

[18] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Journal of Information Retrieval*, 13:254–270, 2009.