# Improving Search Query Suggestion With User Feedback

Michaeel Kazi
LinkedIn Corporation
mikazi@linkedin.com

Weiwei Guo
LinkedIn Corporation
wguo@linkedin.com

Huiji Gao
LinkedIn Corporation
hgao@linkedin.com

Bo Long
LinkedIn Corporation
blong@linkedin.com

## ABSTRACT

At LinkedIn, users rely on meaningful search queries to find specific jobs, qualified candidates, and much more. When a user lacks broad knowledge of industry jargon, we can assist them by suggesting alternate but related queries. To learn these, we can learn from past actions of different users which query reformulations were successful. We can also learn directly by analyzing which suggested queries were clicked. In this paper, we propose a novel method for incorporating both user reformulation data and user suggestion click data to train a deep sequence to sequence model for query generation. Our method involves adding a ranking loss term to the standard log likelihood loss of machine translation. Compared to several alternate approaches for joining the two datasets together, our method significantly outperforms the control model both offline and with respect to key business metrics.

## KEYWORDS

Ranking models, Deep Neural Networks, Sequence to sequence, Query Suggestion, Machine Translation

## 1 INTRODUCTION

Searching within a site like LinkedIn is different than a general web search. The domain is for a specific application, namely finding jobs, people, or industry news. Industry veterans will have more knowledge of key terms and better phrasings that will lead them to their desired results. Novice job seekers may not know the right terminology to find what they want. This is where query suggestion comes in [3, 7].

With query suggestion, we can provide the user with potentially related searches. In Figure 1, we show a typical search at LinkedIn, where suggested queries are listed at the bottom of the page. Users can click on the suggested queries to issue a new search. We can offer suggestions that could guide them to make the most of the search experience. The current state of the art approach is to treat query suggestion as a machine translation problem, and solve it with sequence to sequence modeling [6, 15].

*SIGIR, 2020, Xi'An*
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

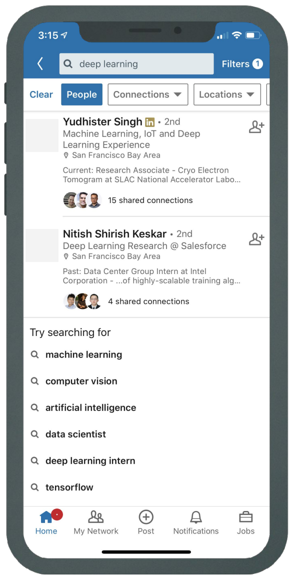

**Figure 1: The "Related Search" product on LinkedIn mobile. Given a query "deep learning", the suggested queries are "machine learning", "computer vision", etc.**

The primary source of training data for this task is search logs [10]. We denote this as **reformulation data**. Search logs are mined for instances where a user made an initial unsuccessful search, changed their search slightly, and then performed a successful search soon afterwards. This can be combined with other heuristics, such as requiring words in common between the query pairs, or requiring a minimum number of users to also have searched for the same pairs. Once the model is trained and deployed online, we can see which queries were chosen, if any. We denote this data as **user feedback data**. The focus of this paper is in using both of these sources of data.

The query reformulation data is valuable as it provides human generated examples of real queries that were rewritten and turned into successful searches. For example, a user may have searched for "relevance programmer", and found few job postings, then searched for "ai engineer" and clicked on a result. User feedback data, on the other hand, can tell us whether a suggested query catches the user's attention. For example, "ethical ai" and "500k ai engineer" may be chosen over "ai engineer intern". However, the second option may not return results. This motivates us to combine the two sources of data and produce a model that provides both useful and attractive suggestions.

We can treat the clicked query pairs from the feedback data as additional machine translation data. However, this misses a key observation: user feedback is a different kind of data. The users did not type them in from scratch; instead, they simply selected them from a list. This is not the same as knowing gold standard suggestions. Instead of inserting them into the training data, a more principled approach would be to give this dataset an entirely different interpretation: clicked versus unclicked labels in our data induce a basic ranking on the suggestions. The clicked suggestion is the better one, and it should receive a higher likelihood score than the others.

Using this intuition, we build a model training procedure that simultaneously trains with the reformulation data with the standard log-likelihood loss, and adds a penalty term for the user feedback data to guide the model into ranking clicked suggestions higher than non-clicked ones. We test this method against several simpler methods: (1) using only reformulation data, (2) using only clicked pairs, (3) model fine-tuning on clicked pairs, and (4) combining the reformulation and clicked pairs into a single dataset.

The advantages of our method are as follows:

- It is a straightforward and powerful way to incorporates both human generated examples and feedback data in sequence to sequence modeling. In many web applications, both data types are available, thus there are many possible applications of this work.
- This method naturally provides a way to insert "negative examples" (i.e. poor quality suggestions) to train a robust model.
- This method is easy to deploy in production. Compared to previous work that explicitly incorporates query history[11], it does not change the underlying architecture of the seq2seq model, only the training procedure. Consequently, the model latency and the online infrastructure are unchanged relative to the baseline.

## 2 RELATED WORK

Query suggestion with machine translation [13], and specifically sequence to sequence modeling, has been successfully deployed in industry [8]. The neural machine translation approach is highly customizable and can easily incorporate many more features, such as search session data [6, 15].

Click Feedback Aware Networks for query suggestion [11] use feedback data in providing high quality query suggestions. CFAN is a ranking model for a separate generation pipeline. Using query session information, it builds a hierarchical, siamese network. Our model, on the other hand, acts directly on the sequence to sequence framework, and does not use any session information during inference. It does generation and ranking in the same model.

Within machine translation, the subproblem of domain adaptation focuses on the of use two separate datasets: an *in-domain* corpus, and a larger *out-of-domain* corpus, with the goal of producing a functioning domain-specific MT system. We highlight two main approaches. The first approach is to combine the in-domain and out-of-domain data into a single corpus [5]. Since the out-of-domain corpus is larger, the machine translation system will be biased towards it. Typically, biases manifest as the system preferring interpretations from the out-of-domain corpus whenever there is overlap. The second approach is fine-tuning the model [12, 14]: first, train on the general domain; then, continue training for some a smaller number of epochs on the specialized domain. This approach emphasizes the in-domain corpus by focusing on it exclusively in the second phase. However, after finetuning, too much knowledge of the original domain can be lost, and advanced approaches can be used to mitigate catastrophic forgetting [17].

All of the above methods work on the assumption that the problem formulation is the same between different datasets. In our setting, however, the user feedback data is not simply more machine translation sentence pairs. It is user preference data on machine generated pairs. In this work, we propose a model format with two separate loss terms for each dataset to handle these separate data types.

## 3 APPROACH

Given a source query $s = s_1, \ldots, s_m$ consisting of a series of tokens, the goal is to directly translate these tokens into a query suggestion $t = t_1, \ldots, t_n$. Our goal is to find a probability function $p(t|s, \Theta)$, where $\Theta$ are model parameters, that is maximal on useful query suggestions.

### 3.1 Sequence to Sequence Modeling

Using the sequence to sequence modeling framework [16] (seq2seq), we can produce a target sequence $t$ of any length from a source sequence $s$. In seq2seq, models have an *encoder* and *decoder*. The encoder, typically an RNN [9, 19] or Transformer [18], takes the input sequence and produces a fixed dimensional vector $h_0$ and a vector for each input token, collectively denoted $\mathbf{e}$. The decoder is a function intended to represent the i-th target word, given all previous target words, and the entire source sentence, using an intermediate hidden state $h_i$:

$$p(t_i|t_{i-1}, \ldots, t_0, s_0, s_1, \ldots, s_m) := p(t_i|h_i, \mathbf{e}; \Theta)$$

The decoder is another RNN/Transformer network, coupled with a softmax layer on each $h_i$ with attention [1]. If $L = L(t)$ is the number of words in $t$, then the log probability of an entire sentence is the sum over log probabilities for each word in the sentence:

$$\log p(t|s) = \sum_{i=0}^{L(t)} \log p_w(t_i|t_{i-1}, \ldots, t_0; s) \tag{1}$$

### 3.2 User Feedback Seq2Seq Model

To use both datasets to train our model, we keep the same negative log likelihood loss term for our reformulation data, but we add a separate term for the feedback data. We start with the simple desire that clicked query suggestions should score higher than non-clicked ones. In our case, the scores are given by Equation 1. We add a new term to the loss term, which is similar to the **pairwise** loss[2, 4]:

$$L_p = -(\log(t_p|s_f) - \log(t_n|s_f)) \tag{2}$$

where $\langle s_f, t_p, t_n \rangle$ is a triple from the feedback data, which we denote $C$, consisting of a source query, a clicked query, and an unclicked query, respectively. For our full loss function, we let $\langle s, t \rangle \in R$ be source and target queries from the reformulation data $R$. We add to equation 2 three parts: (1) a max switch that can disable the penalty, if the positive query is already better; (2) a margin parameter $\epsilon$, which determines how much slack we allow before penalty occurs; and (3) a relative weight $\lambda$ which controls the strength of reformulation vs feedback data contribution to the loss. Putting it all together, we have

$$L = \sum_{\langle s,t \rangle \in R} -\log p(t|s) + \lambda \sum_{\langle s_f, t_p, t_n \rangle \in C} \max(0, \log p(t_n|s_f) - \log(t_p|s_f) + \epsilon)$$

In other words, within a batch, our loss is the total cross entropy between our expectation and prediction, plus a weighted penalty proportional to how much an unclicked example outperforms a clicked one. If the positive example outscores the negative by at least a certain margin $\epsilon$, then no penalty is incurred.

## 3.3 Robustness via Data Augmentation

This formulation easily allows us to augment our data with negative samples. If we know that the model wrongly promotes certain results, such as ungrammatical, fragmented, or repetitive queries, we can insert these as non clicked queries to teach the model to avoid these patterns. Indeed, we have noticed our baseline models sometimes tend to output fragmented queries (for example, queries ending with 'and'), or repeating some words (e.g. 'software engineer and software'). We also experiment with data augmentation in this work. To generate poor examples, we used the following simple algorithm. For a given source query, we generate a 'bad' query by appending on to the source query one of the following at random:

(1) Any random word from the source. E.g. "remote software engineer" becomes "remote software engineer remote"
(2) Any random joiner word from 'and', 'in', 'the', 'of', 'or'. E.g. "software engineer" becomes "software engineer in"
(3) Any random joiner plus a word from the source, e.g. "software engineer", becomes "software engineer and software"

For each example $\langle s_f, t_p, t_n \rangle$ in the feedback data, we add $\langle s_f, t_p, t_{\text{bad}} \rangle$ to the data with probability $p_{\text{bad}}$. This ad-hoc data augmentation technique is merely one possible way of creating negative samples, but it proves useful in our experiments (see Section 4.1.4).

## 4 EXPERIMENTS

In this section we describe our experimental setup. We perform offline experiments to demonstrate the model's performance in optimizing metrics against both reformulation data and feedback data. Afterwards, we perform online experiments to verify our offline metrics correspond to measurable value for users.

**Table 1: Offline Perplexities on Reformulation Data and MRR@6 on feedback data, Different Model Types**

| Model | Perplexity | MRR click |
|---|---|---|
| Reformulation only (baseline) | 13.44 () | 0.5536 |
| Feedback data only | 86.79 (76.29, 99.66) | **0.6128** |
| Reformulation train + finetune click | 23.94 (21.97, 26.12) | 0.5947 |
| Combined data reformulation + click | 13.21 (12.40,14.07) | 0.5760 |
| Seq2Seq Click $\lambda = 0.75$ augmented | 13.41 (12.62,14.30) | 0.5945 |

## 4.1 Offline Experiments

*4.1.1 Datasets.* Our reformulation query pairs are collected over a one years period of data and consist of 180 million query pairs for training, and 2,000 query pairs for each of the validation and test sets[1]. Our click examples are collected over a 6 month period, and are filtered by suggestions that resulted in at least one click. The final count is approximately 17.8 million queries, each with approximately 6 suggestions, at least one of which is clicked. This yields 107 million total feedback triples $\langle s_f, t_p, t_n \rangle$ for our model. Our feedback test set contains 20,000 searches, each containing 5-6 suggestions with at least one clicked.

*4.1.2 Architecture.* For all experiments, we maintain the same architecture. We use an LSTM [9] based seq2seqs model with attention [12]. We use a 100-hidden dimension, 2-layer network with a 60K vocabulary. We train with SGD, learning rate 1.0, for 2 epochs (only 2 since our data is quite large), then begin decaying by $\frac{1}{2}$ ten times over the final 2 epochs.

*4.1.3 Variants and Offline Metrics.* We evaluate several different baselines for incorporating user feedback into query suggestions. The first four models are trained on query pairs under seq2seq framework (equation 1).

(1) **Reformulation only**. This is our baseline model, and it is trained only on query reformulation data $\langle s, t \rangle \in R$.
(2) **feedback data only**. This model is trained only on clicked suggested queries $\langle s_f, t_p \rangle \in C$.
(3) **Reformulation train + Finetune Click**. The model is initialized to our control model, then fine-tuned it for one additional epoch on the clicked suggested queries only.
(4) **Mixed reformulation + click**. This model added the clicked queries directly into the reformulation training data to train a new model.
(5) **Seq2Seq Click**. This is our proposed penalty model evaluated against the same metrics.

We measure (1) *perplexity* on the reformulation data test set, and (2) *mean reciprocal rank* (MRR) on the user feedback test set.

Separately, we also evaluate our proposed method with several of its parameters varied, in order to choose the best parameter settings to choose for comparison. For all data augmentation experiments, we use $p_{\text{bad}} = 1/3$ (Section 3.3) to yield a final feedback data size of approximately 150M examples.

*4.1.4 Results and analysis.* Our offline results are presented in two tables. In Table 2, we tweak various parameters of our click model to arrive at our highest performing model. We found that $\epsilon > 0$

[1]Test and validation sets consisting of only a few thousand examples are common in machine translation settings, see http://statmt.org/wmt19

**Table 2: Offline Perplexities on Reformulation Data and MRR@6 on feedback data, Different Parameters of Seq2Seq Click Model**

| Model | Perplexity | MRR click |
|---|---|---|
| Seq2Seq Click $\lambda = 0.5$ | 13.62 | 0.5896 |
| Seq2Seq Click $\lambda = 0.5$, $\epsilon = 0.25$ | 13.57 | 0.5861 |
| Seq2Seq Click $\lambda = 0.5$, $\epsilon = 0.5$ | 13.58 | 0.5815 |
| Seq2Seq Click $\lambda = 0.75$ | 13.60 | 0.5918 |
| Seq2Seq Click $\lambda = 0.75$, $\epsilon = 0.05$ augmented | 13.56 | 0.5912 |
| Seq2Seq Click $\lambda = 0.75$ augmented | 13.41 | 0.5945 |
| Seq2Seq Click $\lambda = 1$ augmented | 13.57 | 0.5955 |

**Table 3: Online improvement observed relative to the baseline (control). Bold indicates statistically significant at p<0.01**

| Model | Suggestion CTR | Search CTR |
|---|---|---|
| Reformulation only (baseline) | - | - |
| Click-only | **+10.26%** | +0.17% |
| Fine-tune | **+11.50%** | +0.23% |
| Click $\lambda = 0.75$ aug | +5.14% | **+0.48%** |

seemed to hurt performance. For $\lambda$, we tried 0.5, 0.75, and 1.0, and settled on a value of 0.75.

In Table 1, we evaluate our best performing model against several alternatives. The goal of the baseline model is to minimize perplexity of the reformulation dataset, therefore, we expect it to have the lowest perplexity. However, combining the data without duplication gave a slightly lower perplexity, perhaps by focusing the model on its best outputs. The feedback data only model achieves the highest MRR. However, not surprisingly, its perplexity on reformulation data is an order of magnitude worse (86.79 ppl in Table 1 versus mid 13's for the better models). As expected, the baseline and click only models perform very well on their individual metrics, but not on both.

The finetuned model uses both sources of data, though it nearly doubles perplexity (from 13.21 to 23.94). Mixed reformulation + click achieves the best perplexity, and outperforms the baseline on MRR, but falls short of the fine-tuning and feedback data only models. Finally, we see that User Feedback Seq2Seq with augmented negative samples, can close the gap completely with the baseline and finetuned models (within 0.03 ppl and 0.0002 MRR, respectively).

Optimizing for two objectives allows room for qualitative assessment; our model is not the best on either metric, but can be qualitatively considered the "best overall". Instead of introducing an ad-hoc metric based on weighted sums of the perplexity and MRR, we defer to the next section, where we make this claim concrete with the results of online experiments.

## 4.2 Online Experiments

We deployed the baseline model, finetuned model, click-only, and the new seq2seq-click $\lambda = 0.75$ augmented model to a randomly sampled 10% of LinkedIn traffic for two weeks. In Table 3, we report performance relative to our control (Reformulation only) model. We display two online metrics: Suggestion CTR, which measures the click-through rate on *suggested queries only*, and Search CTR, which measures CTR@5 on *documents* (e.g. jobs and people profiles) shown to users site-wide. The second is a holistic measure of the general user experience regarding the whole search system at LinkedIn. Providing better suggested queries can improve on search CTR if the suggestions ultimately guide the user towards what they want. The feedback data only and finetune models generate significantly higher amounts of user clicks on suggested queries, but are not significantly useful to improving user experiences. Seq2Seq-Click fares better. Its 0.48% lift is significant ($p < 0.01$), despite its suggestion CTR even while its lift is 5% instead of 10+%. It is interesting to note the interpretation of the two data sources. Optimizing MRR over feedback data leads directly to a boost in suggestion CTR. Optimizing the perplexity of the query reformulation leads to predicting queries that users find useful. Therefore, optimizing both leads to prioritizing queries that are both useful and likely to be clicked.

## 5 CONCLUSION

In this paper, we have proposed a new method for jointly training a query suggestion seq2seq model on both reformulation data and user feedback. By measuring online performance, we see that this enables us to offer suggestions that are both interesting and useful.

We have focused on query suggestion. In this task, a user's click cannot be treated as the "right" or "only" suggestion. With respect to query length, there are exponentially many possibilities for suggestions, and only a few are evaluated. Because of this, the pairwise penalty is a natural addition to our loss function. Looking beyond, this framework of loss plus pairwise rank term can be applied anywhere one has access to ground truth labels and user preferences on machine generated results.

## REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11 (2010).

[3] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *KDD*.

[4] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*.

[5] Chenhui Chu and Rui Wang. 2018. A Survey of Domain Adaptation for Neural Machine Translation. *CoRR* abs/1806.00258 (2018). arXiv:1806.00258 http://arxiv.org/abs/1806.00258

[6] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *CIKM*.

[7] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *CIKM*.

[8] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *CIKM*.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

[10] Chien kang Huang, Lee feng Chien, and Yen jen Oyang. 2003. Relevant term suggestion in interactive Web search based on contextual information in query session logs. *JASIST* (2003).

[11] Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. 2019. Click Feedback-Aware Query Recommendation Using Adversarial Examples. In *WWW*.

[12] Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *EMNLP*.

[13] Stefan Riezler and Yi Liu. 2010. Query Rewriting Using Monolingual Statistical Machine Translation. *Computational Linguistics* (2010).

[14] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *ACL*.

[15] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion. *CoRR* abs/1507.02221 (2015).
[16] I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *NIPS* (2014).
[17] Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation. In *ACL*.
[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
[19] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* (1989).