Incorporating User Feedback into Sequence to Sequence Model Training

Michaeel Kazi, Weiwei Guo, Huiji Gao, Bo Long
LinkedIn, Mountain View, California
{mikazi,wguo,hgao,blong}@linkedin.com

ABSTRACT

As the largest professional network, LinkedIn hosts millions of user profiles and job postings. Users effectively find what they need by entering search queries. However, finding what they are looking for can be a challenge, especially if they are unfamiliar with specific keywords from their industry. Query Suggestion is a popular feature where a search engine can suggest alternate, related queries. At LinkedIn, we have productionized a deep learning Seq2Seq model to transform an input query into several alternatives. This model is trained by examining search history directly typed by users. Once online, we can determine whether or not users clicked on suggested queries. This new feedback data indicates which suggestions caught the user's attention. In this work, we propose training a model with both the search history and user feedback datasets. We examine several ways to incorporate feedback without any architectural change, including adding a novel pairwise ranking loss term during training. The proposed new training technique produces the best combined score out of several alternatives in offline metrics. Deployed in the LinkedIn search engine, it significantly outperforms the control model with respect to key business metrics.

KEYWORDS

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

ACM Reference Format:
Michaeel Kazi, Weiwei Guo, Huiji Gao, Bo Long. 2020. Incorporating User Feedback into Sequence to Sequence Model Training. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412714

1 INTRODUCTION

Searching within a site like LinkedIn is different than a general web search. The domain is professional communication, including finding jobs, people, or industry news. Industry veterans will have more knowledge of key terms and phrases that will lead them to produce better searches with better results. Novice job seekers may not know the right terminology to find what they want. This is where query suggestion comes in [4, 8].

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412714

Table 1: Examples of query reformulation data. Users first typed in the first query, reformulated it to the second query, and then clicked a document.

First query	Second query (document clicked)
ai	ai engineer
relevance	ai
patent law	ip law
charles smith	chuck smith
charles smith mo	charles smith missouri

With query suggestion, we can provide the user with potentially related searches. We can offer suggestions that could guide them to make the most of the search experience. State of the art approaches typically begin with by treating query suggestion as a machine translation problem, and solve it with sequence to sequence modeling [7, 18]. Within this framework, one can also add in previous search history and other member personal features.

The primary source of training data for this task is search logs [11]. We denote this as **reformulation data**, with examples in Table 1. 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. Success is defined by whether a user performed any additional action on the search results, i.e. interacting with a profile or job listing. 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** (Table 2). 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. In other words, we learn as the users learn. 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" are more interesting or glamorous than "ai engineer intern".

One may be tempted to use only the feedback data to train the query reformulation model. However, this poses two drawbacks. Firstly, the feedback data is machine generated. We are limited to the suggestions provided by whichever machine learning model is deployed during the data collection period. Secondly, the ability to attract the user's eye may lead to 'clickbait' searches. In our example above, the second option ("500k ai engineer") may not return results,

Table 2: Examples of user feedback data. For the query "patent law", 6 related queries were suggested by the related search model. The user clicked on the "intellectual property law".

Query	Suggestion	Clicked
patent law	patent lawyer	0
	intellectual property law	1
	ip law	0
	patent counsel	0
	patent attorney	0
	patent legal	0
mechanical designer	mechanical engineer	0
	solidworks	0
	mechanical design engineer	1
	mechanical drafter	0
	industrial designer	0
	junior mechanical designer	0

or results of low quality. 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 so that it "ranks" higher.

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 incorporate 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, since it only modifies the training procedure. In online serving, the model latency and the online infrastructure are unchanged relative to the baseline.
- Finally, this is orthogonal to approaches that explicitly incorporate query history [12]; this does not change the underlying architecture of the seq2seq model, only the training procedure, and can theoretically be combined with any sequence to sequence model.

2 RELATED WORK

Query suggestion with machine translation [16], and specifically sequence to sequence modeling, has been successfully deployed in industry [9]. The neural machine translation approach is highly customizable and can easily incorporate many more features, such as search session data [7, 18].

Click Feedback Aware Networks for query suggestion [12] 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. 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 [6]. 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 [14, 17]: 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 [20].

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 QUERY SUGGESTION AT LINKEDIN

At LinkedIn, query suggestion is used in the product called Related Search. 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. In the past we have relied on a lookup-table based method whose suggestions are precomputed by collaborative filtering with search sessions on one dimension and queries on the other.

The primary benefit of using machine translation instead of collaborative filtering is that we can produce a suggestion for any query, even unseen ones. In our production systems, it outperforms collaborative filtering by a large margin. We tested the NMT approach relative to collaborative filtering with an offline evaluation

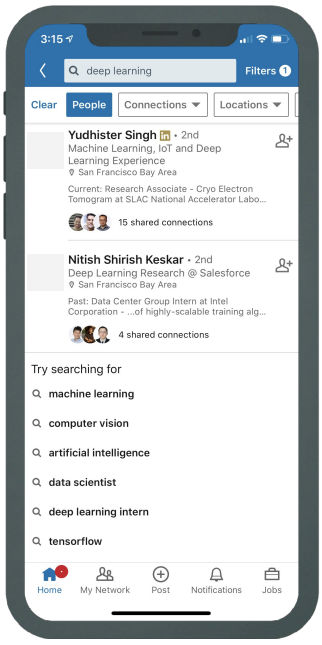

Figure 1: The "Related Search" product on LinkedIn mobile. Given a query "deep learning", the suggested queries are "machine learning", "computer vision", etc.

Table 3: Offline and online results by updating from collaborative filtering to sequence to sequence modeling. All results significant at $p < 0.05$.

Metric	Lift
Coverage (offline)	+48%
MRR (offline)	+11.1%
Job Applications (online)	0.8%
Search Sessions (online)	0.6%

of 50k queries. The coverage (proportion of queries with any suggestion at all) increases by 50%, and the mean reciprocal of clicked suggestions had a relative boost of 11%. Online, the first machine translation based approaches boosted job applications by 0.8% sitewide, and successful search sessions by 0.6%. We define a successful search session as one that led to a click on an entity, and that the user did not immediately close or click back. See Table 3 for a summary of these results.

4 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 Θ are model parameters, that is maximal on useful query suggestions.

4.1 Sequence to Sequence Modeling

In this subsection, we briefly review sequence to sequence modeling. Using the sequence to sequence (Seq2Seq) modeling framework [19], 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 [10, 23] or Transformer [21], 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 p_w 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:

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

The decoder is another RNN/Transformer network, coupled with a softmax layer on each h_i with attention [1, 15]. 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:

$$\log p(t|s) = \sum_{i=0}^{L(t)} \log p_w(t_i|h_{i-1}, \mathbf{e}; \Theta) \quad (1)$$

See Figure 2 for an graphical overview of this model.

4.2 User Feedback Seq2Seq Model

The form of the reformulation data R is identical sequence to sequence modeling inputs: it consists of query pairs $\langle s, t \rangle$. The form of user feedback data C is triples $\langle s_f, t_p, t_n \rangle$, where s_f is a source query, t_p a positive clicked example, and t_n a non-clicked example. The source query does not need to line up with the reformulation source data; this data can be collected independently. 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, 5]:

$$L_p = -(\log(t_p|s_f) - \log(t_n|s_f)) \quad (2)$$

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 ϵ, which determines how much slack we allow before penalty occurs; and (3) a relative weight λ 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

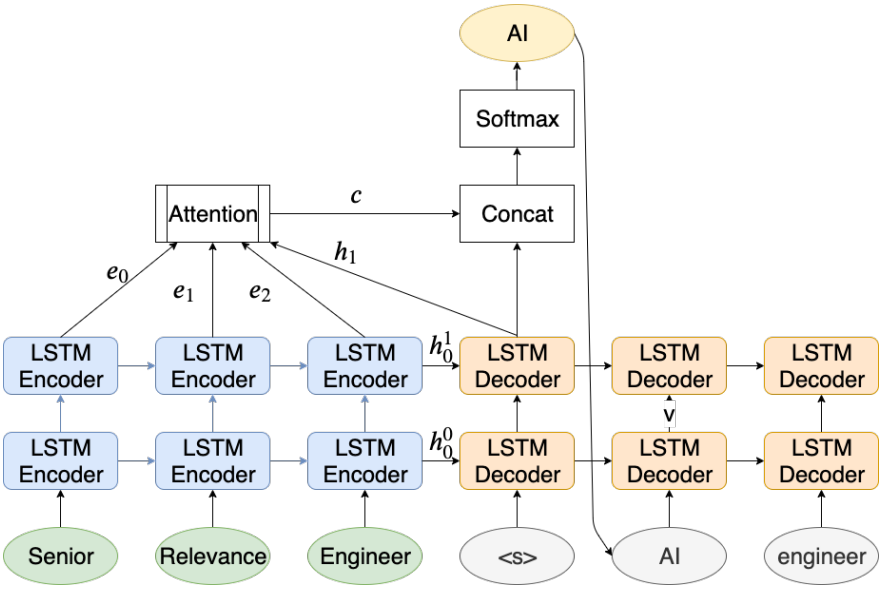

Figure 2: Sequence to Sequence Model

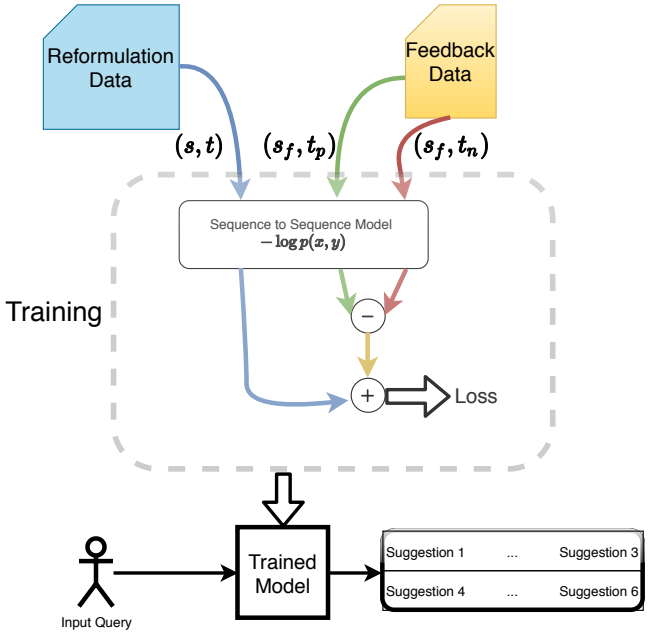

Figure 3: Overview of the User Feedback Seq2Seq Training routine.

least a certain margin ϵ, then no penalty is incurred. For an overview of this model, see Figure 3.

In our experiments, we use an example from R and C together. To simplify experiments, we simply over-sample the smaller of R or C so they line up for each epoch.

4.3 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_{bad}. This ad-hoc data augmentation technique is merely one possible way of creating negative samples, and more analysis is needed to understand its effects, but it does show some qualitative impact on a subset of queries. See Table 4.

5 ONLINE DEPLOYMENT

Our current related search online deployment serves the machine translation model in real-time. The related search module and document retrieval & ranking module are called in parallel; this allows us more time to serve the model, relaxing latency requirements, which are often a problem with deep learning models. Our system also includes the ability to select different related search models,

Table 4: Qualitative analysis of selected queries using data augmentation. Italicized suggestions are of poor quality and can be avoided with data augmentation.

Query	Suggestions - With Data Augmentation	Suggestions - Without
gls quality	gls quality manager, gls qa, gls quality director, gls quality management, gls quality assurance, gls quality engineer	gls quality manager, gls quality director, gls quality engineer, gls safety, gls quality management, *gls quality and quality*
ted	ted talks, ted conferences, ted talk, tedx, *ted at*, theodore	ted talks, ted conferences, ted talk, tedx, *ted ted*, ted tan
random house	random house london, random house uk, penguin random house, random house magazine, random house group, random house uk ltd	penguin random house, random house magazine, random house publishing, the random house, *random house random house*, random house uk
wsl	world surf, world surfing, united nations, the surf, surf, surf league	world surf, world football, world cup, wsl world, world surfing, *world world*
python developer	django developer, java developer, junior python developer, django, python developer looking for, python developer looking	django developer, django, junior python developer, *python developer python*, java developer, *python developer and python*

both for experiment and internationalization purposes, and the ability to retrieve personal user features if the model uses any. We also include a query suggestion caching mechanism, which can retrieve popular queries without performing inference, though this enjoys fewer hits when serving personalized or session aware models. See Figure 4 for an overview of our online serving architecture.

Despite calling the suggestion module in parallel, its latency can still be an issue for longer queries. We have carefully balanced relevance and latency for our baseline models. Thankfully, with our User Feedback Sequence to Sequence approach, the resulting model has the same structure as the baseline model, and does not increase the model's latency.

6 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.

6.1 Offline Experiments

6.1.1 Datasets. Our reformulation query pairs are collected over a one year period and consist of 180 million query pairs for training, and 100,000 query pairs for each of the validation and test sets. 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.

6.1.2 Architecture. For all experiments, we maintain the same architecture. We use an LSTM [10] based seq2seqs model with attention [14]. 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 two additional epochs.

6.1.3 Metrics. On all offline experiments, we measure (1) *perplexity* on the reformulation data test set, and (2) *mean reciprocal rank* (MRR) on the user feedback test set.

Perplexity is a standard measurement of the amount of 'surprise' per word. It is computed as $\exp\left\{\frac{1}{N}\sum_{j=1}^{N} -\log p_j\right\}$. This metric is directly related to the log likelihood loss, and ensures the model remembers the patterns in the data.

Mean Reciprocal Rank (MRR) [22] is evaluated by computing the model probability score on each of k ($k = 6$ in our dataset, since we serve 6 suggestions to the user in production) suggestions for a given source query. We sort the suggestions from high probability to low, and find the rank of the clicked query. The final score is the average of 1/rank. This measurement correlates with assigning higher scores to clicked rather than unclicked suggestions, as we desire. A low perplexity model may cluster several bad suggestions as highly as the clicked one, but a high MRR model is less likely to make this mistake.

6.1.4 Variants. To evaluate the impact of our method of incorporating user feedback into query suggestion, we fix the sequence to sequence architecture to architecture described above, and vary only the training technique. We evaluate several different baselines for incorporating user feedback into query suggestions. The first four models are trained on query pairs under seq2seq framework with the standard loss (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 the reformulation only model, then fine-tuned for one additional epoch[1] on the clicked suggested queries only.

[1] Additional epochs worsened perplexity with no noticeable gain in MRR.

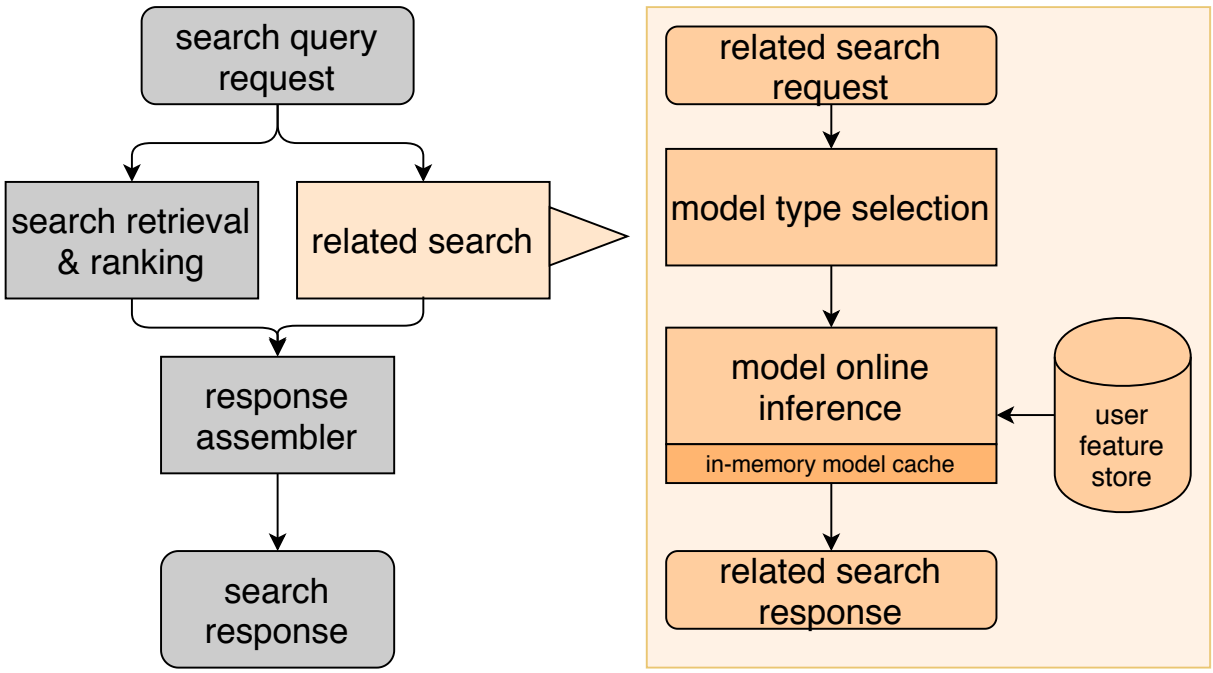

Figure 4: Online serving architecture for query suggestion

Table 5: Offline Perplexities on Reformulation Data, Different Model Types. Clusters of statistical significance with 95% confidence are partitioned from each other by horizontal lines. Systems within the lines are considered tied.

Model	Perplexity
Combined data reformulation + click	13.32
Reformulation only (baseline)	13.32
User Feedback Seq2Seq $\lambda = 0.75$ aug	13.70
Reformulation train + finetune click	23.17
Feedback data only	69.21

Table 6: Offline MRR@6 on feedback data, Different Model Types. Clusters of statistical significance with 95% confidence are partitioned from each other by horizontal lines. Systems within the lines are considered tied.

Model	MRR click
Feedback data only	**0.6127**
User Feedback Seq2Seq $\lambda = 0.75$ aug	0.5979
Reformulation train + finetune click	0.5953
Combined data reformulation + click	0.5744
Reformulation only (baseline)	0.5604

(4) **Mixed reformulation + click**. This model added the clicked queries directly into the reformulation training data to train a new model.
(5) **User Feedback Seq2Seq**. This is our proposed penalty model evaluated against the same metrics.

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 experiments, we supersampled from the feedback data to turn 107 million triples into 180 million triples, to match up with the size of the reformulation data. For all data augmentation experiments, we use $p_{\text{bad}} = 1/3$ (Section 4.3) to yield a final feedback data size of approximately 150M examples.

6.1.5 Results and analysis. Our offline results are presented in two tables. We do a grid search on our click model to arrive at our highest performing model on validation set. For this task, we take a sample of 40 million query pairs and 40 million user feedback triples. We found that $\epsilon < 0$ seemed to improve MRR. For λ, we tried 0.5, 0.75, 1.0, and 1.25, and settled on a value of 0.75.

In Tables 5 and 6, we evaluate our best performing model against several alternatives. By using bootstrap methods, we are able to assess confidence intervals for both perplexity and MRR. We note that only the finetune and click-only models have significantly worse perplexities, and that all the rest remain within 3% of each other.

The goal of the baseline model is to minimize perplexity of the reformulation dataset, therefore, we expect it to have the lowest perplexity. Combining the data without duplication did not sacrifice any perplexity. 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 6 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.

Table 7: Offline Perplexities on Reformulation Data and MRR@6 on feedback data, Hyperparameter tuning

λ	$\epsilon = -0.05$	$\epsilon = 0.0$	$\epsilon = 0.05$
0.5	(14.56, 0.5949)	(14.76, 0.5965)	(14.72, 0.5931)
0.75	(14.76, 0.6014)	(14.72, 0.5979)	(14.80, 0.5931)
1.0	(14.95, 0.6011)	(15.05, 0.6008)	(14.71, 0.6002)
1.25	(15.19, 0.5959)	(14.76, 0.6011)	(15.08, 0.5999)

Table 8: Online improvement observed relative to the baseline (control). Bold indicates statistically significant at p<0.05.

Model	Suggestion CTR	Search CTR
Reformulation only (baseline)	-	-
Click-only	**+10.26%**	+0.17%
Fine-tune	**+11.50%**	+0.23%
Mixed reformulation and click	**+2.17%**	+0.20%
User Feedback Seq2Seq $\lambda = 0.75$	**+5.14%**	**+0.48%**

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 sacrifices a few relative percentage points of perplexity to the best models, and achieves MRR in the same cluster as the fine-tuned model.

Experiments with adding negative samples had no significant impact on model perplexity, but did reduce the MRR performance somewhat. We did not necessarily expect any metric benefit; the model should be easily able to optimize its performance with respect to the metrics given without necessarily adding contrived data to the training set but not the test set. We merely did this to override the model any instinct for producing poor quality results that evaded offline metric detection, to ensure a higher quality user experience.

Optimizing for two objectives allows room for qualitative assessment; our model is not the best on either metric, but yields satisfactory performance on both metrics simultaneously. We contrast it with the the click only model which suffers from an order of magnitude worse perplexity, and the mixed data model, which does not yield as significant a boost to MRR.

Alternatively, we can also introduce an ad-hoc metric that combines, by weighted sum, the perplexity and MRR. However, it is difficult to decide on what exact mixture of metric is valuable for our purposes. Instead, our real single metric that we should focus on will be our business metric: search click-through-rate. In the next section, we examine the model and two key variants for their online performance.

6.2 Online Experiments

We deployed five models to a randomly sampled 10% of LinkedIn traffic for two weeks: the baseline model, finetuned model, click-only, mixed reformulation and click and the new seq2seq-click $\lambda = 0.75$ augmented model. In Table 8, 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 our true-north metric. It 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. Increasing suggestion CTR only amounts to producing "click-bait". Nevertheless, it is a useful measurement of the effects of the models. 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 ($p < 0.05$), 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.

The model was deployed in January of 2020 and has been active at 25% of member traffic ever since, where it continues to provide benefit to users.

7 CONCLUSION AND FUTURE WORK

7.1 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, it is natural to continue to train on user generated data, and only use feedback data to guide and refine the model. The pairwise penalty is a natural addition to our loss function to achieve this. 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.

7.2 Future Work

The most obvious future directions would be to add this model to a fuller query suggestion framework, for example, one that incorporates search history[18], member specific features, and transformer networks.

As a future improvement, we can gather more detailed ranking information in the training data. We can rank clicked query suggestions with document clicks higher than clicked suggestions that only led to viewing the search results page. Artificially poor suggestions can be more than simply 'unclicked' suggestions, they can actually be modeled as worse than those. In other words, we can encode a richer relation like $p(\text{clicked suggestion AND clicked document}) > p(\text{clicked suggestion}) > p(\text{not\_clicked}) > p(\text{bad})$. This can be achieved by either (1) encoding each (adjacent) relationship in a pairwise term, or by (2) moving to a list-wise ranking framework.

Finally, we can incorporate adversarial feedback into this process. If we allow for a separate ranker model (e.g. [13]), we may be able to build a more complete picture of user preference. Though GAN models for text are still difficult to use successfully[3], we may be able to relax the requirement that the ranker be "fooled" by the generator. Care would need to be taken to avoid overfitting to the quirks of the ranker.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

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

[3] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language GANs Falling Short. *CoRR* abs/1811.02549 (2018). arXiv:1811.02549 http://arxiv.org/abs/1811.02549

[4] 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 *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 875–883.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[6] 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

[7] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1747–1756.

[8] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 696–703.

[9] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1443–1452.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] 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. *Journal of the American Society for Information Science and Technology* 54, 7 (2003), 638–649.

[12] Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. 2019. Click feedback-aware query recommendation using adversarial examples. In *The World Wide Web Conference*. 2978–2984.

[13] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*. 3155–3165.

[14] Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*. Da Nang, Vietnam.

[15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *The Conference on Empirical Methods in Natural Language Processing*.

[16] Stefan Riezler and Yi Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics* 36, 3 (2010), 569–582.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 86–96. https://doi.org/10.18653/v1/P16-1009

[18] 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. (2015), 553–562.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[20] Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2062–2068.

[21] 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 *Advances in neural information processing systems*. 5998–6008.

[22] Ellen M Voorhees and Donna K Harman. 2000. *The eighth text retrieval conference (TREC-8)*. Technical Report.

[23] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1, 2 (1989), 270–280.