Using BERT and BART for Query Suggestion

Agnès Mustar
Sorbonne Université, CNRS, LIP6, F-75005
Paris, France
agnes.mustar@lip6.fr

Sylvain Lamprier
Sorbonne Université, CNRS, LIP6, F-75005
Paris, France
sylvain.lamprier@lip6.fr

Benjamin Piwowarski
Sorbonne Université, CNRS, LIP6, F-75005
Paris, France
benjamin.piwowarski@lip6.fr

ABSTRACT

Transformer networks have recently been successfully applied on a very large range of NLP tasks. Surprisingly, they have never been employed for query suggestion, although their sequence-to-sequence architecture makes them particularly appealing for this task. Query suggestion requires to model behaviors during complex search sessions to output useful next queries to help users to complete their intent. We show that pre-trained transformer networks exhibit a very good performance for query suggestion on a large corpus of search logs, that they are more robust to noise, and have a better understanding of complex queries.

CCS CONCEPTS

• **Information systems** → **Query suggestion**; **Query reformulation**; *Query representation*; *Language models*; • **Computing methodologies** → Learning latent representations.

KEYWORDS

Queries suggestion, Transformers, User modeling

1 INTRODUCTION

To explore the space of potentially relevant documents, users interact with search engines through queries. This process can be improved, since when looking for information, users may have difficulties to express their needs at first sight, and hence may have to reformulate the queries multiple times to find the documents that satisfy their needs. This process is particularly exacerbated when the user is accomplishing a complex search task.

Among the different ways to help users in exploring the information space, modern search engines provide a list of query suggestions, which help users by either following their current search direction – e.g. by refining the current query – or by switching to a different aspect of a search task [28] if the proposed queries match a different aspect of the user information need. Another use of query suggestions is to help the search engines by providing ways to diversify the presented information [36].

To suggest useful queries, most models build upon web search logs, where the actions of a user (queries, clicks, and timestamps) are recorded. User sessions are then extracted by segmenting the web search log. The first query suggestion models exploited the query co-occurrence graph extracted from user sessions [14, 15]: if a query is often followed by another one, then the latter is a good potential reformulation. However, co-occurrence based models suffer from data sparsity, for instance when named entities are mentioned, and lack of coverage for rare or unseen queries. Moreover, these models are difficult to adapt when using a wider context than the last submitted query [7].

More recently, recurrent neural network-based (RNNs) methods have been proposed to exploit longer dependencies between queries [1, 2, 7, 37, 40]. RNNs do so by keeping track of the user in a representation/vector space which depends on all the actions performed by the user so far. Such models have improved the quality of suggestions by capturing a broader context, but are limited by the relatively short span of interaction that RNNs are able to capture.

Common NLP tasks [22, 31, 34, 38, 39, 41] have benefited from the recently proposed Transformers architecture [39]. Transformer networks, such as Bert [8], capture long-range dependencies between terms by refining each token representation based on its context before handling the task at hand. They are thus a particularly interesting architecture for query suggestion since query terms are often repeated throughout a session, and their interaction needs to be captured, to build a faithful representation of the current user state.

In this work, we compare and analyse the results of pre-trained transformers for query suggestion to the ones from RNN-based models.

2 RELATED WORK

A large number of works have focused on the task of query suggestion [29], and related tasks such as query auto-completion [26], based on search logs to extract query co-occurrences [14, 15]. From a given single query formulated by a user, the goal is to identify related queries from logs, and to suggest reformulations based on what follows in the retrieved sessions, assuming subsequent queries as refinements of former ones [33]. These works rely on several methods, such as using term co-occurrence [14], using users click information [25], using word-level representation [4], capturing higher order collocation in query-document sub-graphs [3], clustering queries from logs [33], or defining hierarchies of related search tasks and sub-tasks [11, 24]. Some methods finally prevent query sparsity via reformulations using NLP techniques [29]. [15] proposes an end-to-end system to generate synthetic suggestions, based on query-level operations and information collected from available text resources.

However, such log-based methods suffer from data sparsity and are not effective for rare or unseen queries [37]. In addition, these approaches are usually context-agnostic, focusing on matching candidates with a single query. But, when the query comes in a session with some previous attempts for finding relevant information, it is crucial to leverage such context for capturing the user intent and understanding its reformulation behavior. Note the approach in [5],

which alleviates the problem by relating the user sessions to paths in a concept tree also suffers from data sparsity issues.

Instead of trying to predict directly a query, it is possible to learn how to transform it. Most approaches operate at a high level, with term retention, addition and removal as the possible reformulation actions [20, 35]. [20] consider these actions as feedback from the user – e.g. a term that is retained during the whole session should be considered as central for the user intent. Depending on the previous sequence of users' actions, these methods seek to predict the next action. These methods are interesting because they model the user behavior in a session. However, they fail at capturing the semantic of words, which is essential.

To cope with limitations of log-based and action-based methods, some works propose to define probabilistic models for next query prediction [12]. Due to their ability for processing sequences of variable size, Recurrent Neural Networks (RNNs) have been widely used for text modeling and generation tasks, with an encoder that processes an input sequence by updating a representation in $\mathbb{R}^n$, and a decoder that generates the target sequence from the last computed representation. Some works have adapted these ideas to a sequence of queries [7, 16, 37]. HRED [37] proposes to use two encoders: a query-level encoder, which encodes each query of the user session independently, and a session-level encoder, which deals with the sequence of query representations. Instead of using a hierarchical representation, ACG [7] relies on attention mechanism that is used to give a different importance to words and queries in the representation. Another improvement of ACG is to deal with Out-Of-Vocabulary (OOV) words through the use of a copy mechanism, which allows the model to pick tokens from the past user queries rather than generating from using the standard RNN decoding.

Other RNN based approaches have also been recently proposed, such as [40], which leverages user clicks and document representations to specify the user intent [1, 2], or [16] which integrates click-through data into homomorphic term embeddings to capture semantic reformulations. In this work, as a starting point, we restrict to queries in sessions as input data, but other sources of information can be added to such models.

In parallel, the Transformers architecture, a recent and effective alternative to RNNs models introduced in [39], was successfully applied to a large broad of NLP applications, such as Constituency Parsing and Automatic Translation [39], Semantic Role Labeling [38], Machine Reading Comprehension [22], and Abstractive Text Summarization [34].

The Transformer has also has been used several times in the field of Information Retrieval. [27] and [10] applied transformers to infer query from a document. [27] used the pretrained transformer BERT, and showed that expanding the document with the predicted query improve the ad hoc retrieval results, while [10] presented a more complex seq2seq architecture: the encoder included a Graph Convolutional Network and a RNN; and the decoder is a transformer. Transformers have also been used for ad hoc retrieval [6, 23, 31, 41]. [23] used BERT features in existing ranking neural models, and outperforms state-of-the-art ad hoc ranking scores.

3 TRANSFORMERS FOR QUERIES SUGGESTION

In this section, we first present the transformer network architecture and pre-trained transformers before describing how we use them for query suggestion.

3.1 The Transformer architecture

The transformer architecture was introduced by [39]. It is composed of parametric functions that successively refine the representation of sequences, both for the encoder and the decoder. In our case, the encoder is used to represent the session, and the decoder to generate the next query.

Each layer of the encoder or the decoder transforms the sequence x composed of n vectors $x_1, \ldots, x_n$ into a sequence $y_1, \ldots, y_n$ of the same length, through an attention over a context sequence c composed of n vectors $c_1, \ldots, c_n$. Each time, the central mechanism is to use an attention mechanism – other operations are conducted to ensure a stable and efficient learning process, and are detailed in [39].Special tokens are used to separate texts (`[SEP]`) or perform classification (`[CLS]`).

3.2 Pre Trained Transformers

Transformer models have a lot of parameters, and can be long to train. Recently, multiple pre-trained models trained on large datasets have been released [8, 21, 32, 42]. We compare the results of the fully trained transformer, to two pre-trained models that we finetune: BERT [8] and BART [21].

Bert. The Bidirectional Encoder Representations from Transformers [8] have been trained on a large dataset, the BooksCorpus [43] on two tasks, namely predicting some masked tokens of the input, and on predicting whether one sentence follows another. It is a state-of-the-art model, which is used for different tasks. BERT corresponds to the encoder part only – we have to train a decoder for our specific task.

Bart. Bidirectional and Auto-Regressive Transformer [21] is made of an encoder and a decoder. It is trained on the same data than BERT, but for multiple tasks: token masking, tokens detection, text infilling, sentence permutation, and document rotation. Because it has a decoder and it is trained on these tasks, the authors claim that BART is better than BERT for text generation. They also released fine-tuned versions of BART for other tasks. We use the weights of the model fine-tuned on CNN/DM, a news summarization dataset, because as a text generation task it was the closest task to the query suggestion task.

3.3 Using Transformer networks for Query Suggestion

3.3.1 Problem Setting. Let us consider a session $S = (Q_1, ..., Q_{|S|})$ as a sequence of $|S|$ queries, where every $Q_i = (w_{i,1}, ..., w_{i,|Q_i|})$ is a sequence of $|Q_i|$ words. The goal of query suggestion is to suggest the most relevant query for the user intent represented by the session. However, no perfect ground truth can be easily established for such problems: defining the perfect query for a given specific under defined need, given a sequence of past queries, is

an intractable problem, which requires to consider very diverse (in nature and complexity) search tasks, depends on the user state, the IR system and the available information in the targeted collection. Following other works on model-based query suggestion, we thus focus on predicting the next question within an observed session.

We suppose that our dataset is composed of pairs $(S, \check{Q})$ where $\check{Q}$ is the query following a sequence of queries S. Our aim is thus to find the parameters θ that maximize the log probability of observing the dataset:

$$\mathcal{L}(S;\theta) = \sum_{(S,\check{Q})} \log p_\theta(\check{Q}|S) = \sum_{(S,\check{Q})} \sum_{t=1}^{|\check{Q}|} \log p_\theta(w_t|Q_1,\ldots,Q_{|S|}) \tag{1}$$

where $(w_1, ..., w_{|\check{Q}|})$ are the token of the query $\check{Q}$. We describe below how we use the transformer – we tried to build different architectures based on the transformer, but the simplest one worked the best throughout all our pilot experiments.

Input. For a session, the input of the transformer is simply the concatenation of all the words of all the queries separated by a token [SEP], i.e. the [SEP] is used to mark the beginning of a new query in the session:

$$S = [[SEP] \underbrace{w_{1,1} \ldots w_{1,|Q_1|}}_{Q_1} [SEP] \ldots [SEP] \underbrace{w_{|S|,1} \ldots w_{|S|,|Q_{|S|}|}}_{Q_{|S|}} [SEP]]$$

This sequence is then transformed by using the token embeddings added to positional embeddings (one per distinct position) – this is how Transformers recover the sequence order [39].

3.3.2 BERT. We use the pre-trained model Bert [8], and extract each layer of decoder. We sum the last layer, with the average and the max of these layers . For each token of the input, we have a contextualized embedding of size 768 given by Bert. For the decoding part, we use a transformer decoder and feedforward network. At the beginning of the training the encoder is frozen and the decoder is trained. Then we use the gradual unfreezing method, as recommended by [13]: when the loss stabilizes, we unfreeze the last frozen layer of the encoder, until all the layers are finetuned.

3.3.3 BART. The architecture is complete for text generation, it has an encoder and a decoder. We also use gradual unfreezing to finetune the model, but starting from the last layer of the pre-trained decoder.

4 EXPERIMENTS

Datasets. We conduct our experiments of the AOL dataset. It consists of 16 million real search log entries from the AOL Web Search Engine for 657,426 users. Following [37], we delimit sessions with a 30-minutes timeout. The queries submitted before May 1, 2006, are used as the training set, the remaining four weeks are split into validation and test sets, as in [37]. The queries are processed by removing all non-alphanumeric characters and lowercasing following [37]. After filtering, there are 1,708,224 sessions in the training set, 416,450 in the test set and 416,450 in the validation set.

Based on https://github.com/hanxiao/bert-as-service, and our own preliminary experiments

Compared Models. In our experiments we compare RNN-based approaches against fine-tuned transformer models. The RNN models are HRED [37] and ACG [7] described in section 2. The pre-trained models that we finetune are BERT [8] and BART [21].

In order to isolate possible causes of performance variations, models optimization is performed on the training sets of sessions with the ADAM optimizer [18]. All hyper-parameters are tuned via grid-search on the validation dataset.

Query suggestion metrics. As a metric to evaluate generated queries compared to the target ones, we first use the classical metric BLEU [30], which corresponds to the rate of generated n-grams that are present in the target query. We refer to BLEU-1, BLEU-2, BLEU-3 and BLEU-4 for 1-gram, 2-grams, 3-grams and 4-grams respectively. We also calculate the exact match EM (equals to 1 if the predicted query is exactly the observed one, 0 otherwise).

As EM can be too harsh, we also use a metric, $\text{Sim}_{extrema}$ [9], which computes the cosine similarity between the representation of the candidate query with the target one. The representation of a query q (either target or generated) is a component-wise maximum of the representations of the words making up the query (we use the GoogleNews embeddings, following [37]). The extrema vector method has the advantage of taking into account words carrying information, instead of other common words of the queries

However, this component-wise maximum method might excessively degrade the representation of a query. As an alternative, we propose to compute $\text{Sim}_{pairwise}$ as the mean value of the maximum cosine similarity between each term of the target query and all the terms of the generated one.

Finally, as discussed in section 3.3, there is no ground truth on what the best queries to suggest are. Instead, for each generation metric, we consider a standard max-pooling from the top-10 queries generated by the models. More precisely, for each model, we first generate (through a beam search with $K = 20$) 10 queries to suggest to the user given the context. The reported value for each metric (BLEU, EM, $\text{Sim}_{extrema}$ and $\text{Sim}_{pairwise}$) is the maximum score over the 10 different generated queries. This is usually employed for assessing the performance of a probabilistic model w.r.t. a single target (see e.g., [19]) and corresponds to a fair evaluation of models that try to find a good balance between quality and diversity.

4.1 Results

Tables 1 report results obtained by all models on generated queries. We added two further indicators, the ratio of new words, and the rank of the prediction in the beam search if the predicted query appears in the context (or 10 if it doesn't, so values can be averaged).

From a high level point of view, we see that pre-trained transformers are better performing than the RNN-based models, HRED and ACG, on all metrics and that Bart performs better than Bert.

We also note that models have different tendencies to copy one of the queries in the session. Note that this is a standard behavior, 6% of the AOL queries to predict are among the previous queries of the session. So it is not surprising that more powerful models

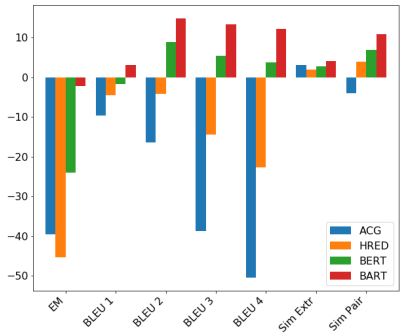
As we want to encourage the models trained with a word tokenizer to generate tokens present in the vocabulary, we follow [17] and apply a penalty on the "OOV" token. To compute the metrics, we ignored the OOV token that can be generated by HRED or ACG – queries with only OOV words are skipped.

Table 1: Results on the AOL dataset. ⋆ indicates significant gains ($p < 0.05$) compared to BERT. † indicates significant gains ($p < 0.05$) compared to HRED.

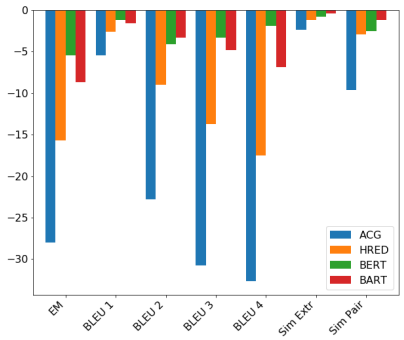
	ACG	HRED	BERT	BART
EM	0.018	0.030	0.060†	**0.121**†*
BLEU 1	0.418	0.408	0.459†	**0.551**†*
BLEU 2	0.126	0.122	0.193†	**0.313**†*
BLEU 3	0.038	0.052	0.109†	**0.231**†*
BLEU 4	0.005	0.017	0.060†	**0.172**†*
$sim_{extrema}$	0.665	0.710	0.741†	**0.789**†*
$sim_{pairwise}$	0.405	0.404	0.467†	**0.553**†*
New Words	0.118	0.681	0.927	0.684
Repetition Rank	7.109	7.789	6.713	2.196

Figure 1: Difference (in %) between the performance on all the AOL sessions and one of its subsets (potentially modified). Negative values indicate a degradation.

(a) *Complex* sessions



(b) *Concatenated* sessions



learn to copy – transformer models have thus a tendency to repeat a seen query compared to ACG or HRED (lower Repetition Rank). This tendency is explained by their ability to retrieve information at arbitrary positions in the input. While BERT is generating much new words while keeping a repetition rank low, BART has a higher repetition rank and is adding less new words.

Results on complex sessions. We were also interested in how the different models could handle complex sessions. To identify those, we used a simple heuristic, which was empirically validated on a sample of sessions: A complex session (1) consists of at least three queries; (2) contains queries with more than one word; and (3) to discard sessions that only contain spelling corrections, each of its queries must be sufficiently different from the previous one – we use a simple editing distance in characters with 3 as threshold.

Figure 2a reports the relative results obtained on this subset of 193,336 complex sessions (compared to corresponding results from all sessions of the AOL dataset). It emphasizes the good behavior of transformers for query suggestion (and especially pre-trained ones), since on this subset of sessions, they improve over the other models on every metric. Moreover, the transformers have a better performance on complex search sessions than when considering all sessions (except for EM), which means that this type of model is particularly well suited to support users having a complex information need, and also shows that HRED and ACG are more suited for simple reformulations (spelling, etc.).

Results on concatenated sessions. To assess the robustness of the approaches, we add one random session at the start of each session of the test set. Since the intent of these added sessions is (in average) not the same as the intent driving the user's behavior when formulating test queries, models must have learned to identify thematic breaks, and to ignore this noisy information. Figure 2b shows percentages of performance loss for every metric. We can see that, while RNN-based models have a lesser performance, pre-trained transformers are greatly less impacted than others. This is an important result, since test sessions were arbitrarily split according to a 30-minute timeout, which might not correspond to users' intent changes. Pre-trained transformers can adapt themselves to longer history, by efficiently focusing on the relevant part. We believe that this is due to the fact that those models have learned to detect topic changes on much more data. The same observations were made on additional experiments (not shown here), where some context queries were replaced at random. Again, pre-trained transformers showed a lower performance decrease, showing that they were better to ignore noisy context queries.

5 CONCLUSION

In this paper, inspired by the success of transformer-based models [39] in various NLP and IR tasks, we looked at its application to query generation and found that in this domain again, transformers could better handle this task than RNN-based models – even when trying to incorporate some of the elements at the origin of its success. The transformers have proven to to be more resilient to noise, to be able to detect thematic boundaries in multi-task sessions, and generate more diverse results than the previously proposed models. Future work will focus on integrating various sources of information beside queries, and to develop architectures able to cope with long sessions (potentially all the user history).

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context Attentive Document Ranking and Query Suggestion. *CoRR* abs/1906.02329 (2019). arXiv:1906.02329 http://arxiv.org/abs/1906.02329

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. (2018).

[3] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 56–63.

[4] Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. 2012. Efficient query recommendations in the long tail via center-piece subgraphs. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 345–354.

[5] 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*. ACM, 875–883.

[6] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. *arXiv preprint arXiv:1905.09217* (2019).

[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*. ACM.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, Vol. 2.

[10] Fred X Han, Di Niu, Kunfeng Lai, Weidong Guo, Yancheng He, and Yu Xu. 2019. Inferring Search Queries from Web Documents via a Graph-Augmented Sequence to Attention Network. In *The World Wide Web Conference*. ACM, 2792–2798.

[11] Ahmed Hassan Awadallah, Ryen W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM.

[12] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE '09)*. IEEE Computer Society, Washington, DC, USA, 12. https://doi.org/10.1109/ICDE.2009.71

[13] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[14] 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).

[15] Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. 2011. Synthesizing High Utility Suggestions for Rare Web Search Queries. In *ACM SIGIR* (Beijing, China) *(SIGIR '11)*. ACM, 10.

[16] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation Inference Network for Context-Aware Query Suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM.

[17] Atsuhiko Kai, Yoshifumi Hirose, and Seiichi Nakagawa. 1998. Dealing with out-of-vocabulary words and speech disfluencies in an n-gram based speech understanding system. In *Fifth International Conference on Spoken Language Processing*.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. 2019. VideoFlow: A Flow-Based Generative Model for Video. *CoRR* abs/1903.01434 (2019). arXiv:1903.01434 http://arxiv.org/abs/1903.01434

[20] Nir Levine, Haggai Roitman, and Doron Cohen. 2017. An extended relevance model for session search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 865–868.

[21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[22] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint 1712.03556* (2017).

[23] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1101–1104.

[24] Rishabh Mehrotra and Emine Yilmaz. 2017. Extracting hierarchies of search tasks & subtasks via a bayesian nonparametric approach. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[25] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. 2008. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM.

[26] Bhaskar Mitra and Nick Craswell. 2015. Query Auto-Completion for Rare Prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (Melbourne, Australia) *(CIKM '15)*. ACM, 4.

[27] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019).

[28] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. [n.d.]. Learning to Suggest: A Machine Learning Framework for Ranking Query Suggestions a Machine Learning Framework for Ranking Query Suggestions. In *ACM SIGIR* (2012).

[29] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *ACM SIGIR*. ACM.

[30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[31] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint 1904.07531* (2019).

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019).

[33] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*. ACM, 841–850.

[34] Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers Unite! Unsupervised Metrics for Reinforced Summarization Models. *CoRR* abs/1909.01610 (2019). arXiv:1909.01610 http://arxiv.org/abs/1909.01610

[35] Marc Sloan, Hui Yang, and Jun Wang. 2015. A term-based methodology for query reformulation understanding. *Information Retrieval Journal* 18, 2 (2015), 145–165.

[36] Yang Song, Dengyong Zhou, and Li-wei He. [n.d.]. Post-Ranking Query Suggestion by Diversifying Search Results. In *ACM SIGIR* (New York, NY, USA, 2011) *(SIGIR '11)*. ACM. https://doi.org/10/b6d6s9 ZSCC: 0000049.

[37] 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. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM.

[38] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

[39] 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*.

[40] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query suggestion with feedback memory network. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1563–1571.

[41] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple applications of bert for ad hoc document retrieval. *arXiv preprint 1903.10972* (2019).

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).

[43] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *IEEE international conference on computer vision*. 19–27.