On the Study of Transformers for Query Suggestion

AGNÈS MUSTAR, SYLVAIN LAMPRIER, and BENJAMIN PIWOWARSKI,
Sorbonne Université, CNRS

When conducting a search task, users may find it difficult to articulate their need, even more so when the task is complex. To help them complete their search, search engine usually provide query suggestions. A good query suggestion system requires to model user behavior during the search session. In this article, we study multiple Transformer architectures applied to the query suggestion task and compare them with recurrent neural network (RNN)-based models. We experiment Transformer models with different tokenizers, with different Encoders (large pretrained models or fully trained ones), and with two kinds of architectures (flat or hierarchic). We study the performance and the behaviors of these various models, and observe that Transformer-based models outperform RNN-based ones. We show that while the hierarchical architectures exhibit very good performances for query suggestion, the flat models are more suitable for complex and long search tasks. Finally, we investigate the flat models behavior and demonstrate that they indeed learn to recover the hierarchy of a search session.

CCS Concepts: • **Information systems** → **Query representation**; **Query suggestion**; *Query intent*; *Query log analysis;*

Additional Key Words and Phrases: User modeling, query prediction, query suggestion, transformers, bert, bart, hierachical model

ACM Reference format:
Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. 2021. On the Study of Transformers for Query Suggestion. *ACM Trans. Inf. Syst.* 40, 1, Article 18 (October 2021), 27 pages.
https://doi.org/10.1145/3470562

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 [49]. Another use of query suggestions is to help the search engines by providing ways to diversify the presented information [59].

This work was supported by the Agence National de la Recherche (ANR), through project CoST, code ANR-18-CE23-0016.
Author's address: A. Mustar, S. Lamprier and B. Piwowarski, Sorbonne Université, LIP6, 4 place Jussieu, F-75005 Paris, France; emails: agnes.mustar@lip6.fr, sylvain.lamprier@lip6.fr, benjamin.piwowarski@lip6.fr.

1046-8188/2021/10-ART18 $15.00
https://doi.org/10.1145/3470562

There are two ways to approach the task of query suggestions. Either in a direct way, seeking directly to improve the user experience. This involves searching for the most suitable queries so that the user accesses the most relevant information as quickly as possible [6]. Such an approach requires a mean to assess what constitutes a relevant suggestion, or data on whether or not suggestions are relevant. The second approach consists in modeling the average user [1, 2, 9, 20, 60, 65]. The goal is to predict the next query based on the current search session—where the data are nowadays abundant. The hypothesis is that suggesting such queries from sessions usually helps users in their search. In the absence of a public dataset allowing to train and evaluate models on the first type of approach, this latter type of approach is usually pursued. This is the scope of this article.

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 [29, 30]: 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 [20].

More recently, **recurrent neural network** (**RNN**)-based methods have been proposed to exploit longer dependencies between queries [1, 2, 20, 60, 65]. RNNs do so by keeping track of the user in a representation/vector space which depends on all the previous actions performed by the user. 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.

Beyond query suggestion, working with representation-based models such as neural networks are particularly interesting, since the learned representations can be useful for models exploiting user sessions, such as in interactive IR models. If a model is able to correctly generate a query, then it means that it has captured (at least partially) the user intent. Developing neural models able to predict with high accuracy the next queries of a user are thus important for building interactive and discussion-based retrieval systems.

Among all the models exploited in NLP and IR, most [40, 52, 56, 61, 63, 66] have benefited from the recently proposed Transformers architecture [63]. Transformer networks, such as **Bidirectional Encoder Representations from Transformers** (BertBert [21], 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. Recently, Garg et al. [23] presented a Hierarchical Transformer for Query Suggestion, with a two-level encoder. Their model outperforms the hierarchical recurrent based models [20, 60], and shows that recurrence is not essential for the query suggestion task. In opposition to this type of hierarchical transformers, we refer in this article to the classical transformer networks as flat transformers.

However, the authors of [23] do not provide a full analysis of whether the hierarchical architecture is important, especially for complex user sessions that are particularly interesting in the context of interactive IR. In this work, we contribute to the study of transformers for the query suggestion task—and more generally, for models able to analyze user sessions:

- We reproduce RNN-based models experiments [20, 60] and extend them by segmenting queries using subword units, which allow transformers to avoid the problem of out-of-vocabulary tokens.

- We also reproduce the Hierarchical Transformer architecture [23], with word and sub-word units, and compare it with flat Transformers.
- We compare the flat Transformers with two pre-trained transformers: Bert, **Bidirectional and Auto-Regressive Transformer (Bart),** and T5, finetuned for our task. We also integrate these pre-trained models to the Hierarchical Transformer.

The analysis is structured into three research questions that we detail below. First, we are interested in the performance of transformers from a global point of view.

Q1. How well the various presented transformers generate queries suggestions compared to the usual baselines?

When a user performs a complex search, it is more difficult to capture the intent of the user. However, such sessions are of particular interest for nowadays IR research, and in particular for interactive IR. We thus pay a particular attention to the robustness of the different models on sessions corresponding to so-called "complex" search tasks. This raises the question of whether all transformers have the same ability to handle long, complex or noisy sessions, or whether, on the contrary, the results are impacted differently depending on the pre-training or the architecture of the transformer.

Q2. Which model is the most robust?
(a) to complex sessions
(b) to noisy sessions
(c) to long sessions

Following the analyzes conducted to answer **Q2.**, we conclude that flatten pre-trained transformers are more resilient to noise, length, and complexity of sessions. We are investigating why these models are more robust, which leads us to our final research question:

Q3. How does the flat transformer generate queries?
(a) On which context's queries does it focus its attention?
(b) On which context's tokens does it focus its attention?
(c) How does it choose the next token to generate?

The analyzes and answers to these questions aim at better understanding the behavior of various Transformer architectures for user modeling.

2 RELATED WORK

A large number of works have focused on the task of query suggestion [50], and related tasks such as query auto-completion [46], based on search logs to extract query co-occurrences [29, 30]. 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 [55]. These works rely on several methods, such as using term co-occurrence [29], using users click information [45], using word-level representation [8], capturing higher order collocation in query-document sub-graphs [7], clustering queries from logs [55], or defining hierarchies of related search tasks and sub-tasks [26, 44]. Some methods finally prevent query sparsity via reformulations using NLP techniques [50]. For instance, Jain et al. [31] propose an end-to-end system to generate synthetic suggestions, based on query-level operations and information collected from available text resources. Broccolo et al. [9] propose to alleviate the sparsity issue by creating a knowledge base from query logs. The database is filled with train log queries to make synthetic documents. The idea, is then to define a function which measures the similarity of a virtual document with this base and a new session. Each token of the session is taken into account independently to calculate this similarity, which allows unseen queries to be treated. The title from the closest virtual documents are the suggestions.

However, such log-based methods suffer from data sparsity and are not effective for rare or unseen queries [60]. In addition, these approaches are usually context-agnostic, focusing on matching candidates with a single query. 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 [12], which alleviates the problem by relating the user sessions to paths in a concept tree, but 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 [38, 58]. Levine et al. [38] 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 [27]. Due to their ability for processing sequences of variable size, 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 [20, 33, 60]. HRED [60] 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 [20] relies on attention mechanism giving a different importance to words and queries in the computed 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 them using a fixed-size vocabulary.

Other RNN-based approaches have also been recently proposed, such as [65], which leverages user clicks and document representations to specify the user intent [1, 2], or [33] which integrates click-through data into homomorphic term embeddings to capture semantic reformulations. Some works have explored the use of long-term search history of users [14], using a RNN-based hierarchical architecture, to score query suggestions. 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 [63], was successfully applied to a large set of NLP applications, such as Constituency Parsing and Automatic Translation [63], Semantic Role Labeling [61], Machine Reading Comprehension [40], and Abstractive Text Summarization [56].

The Transformer architecture has also been used several times in the field of Information Retrieval. Nogueira et al. [48] and Han et al. [25] applied transformers to infer the queries relevant to a document. Nogueira et al. [48] used the pre-trained transformer BERT, and showed that expending the document with the predicted query improves the ad hoc retrieval results, while Han et al. [25] presented a more complex seq2seq architecture: the encoder included a Graph Convolutional Network and an RNN; and the decoder is a transformer. Several works focused on transformers applied to conversational search [3, 19, 68], in particular Yu et al. [68] used a pre-trained model for conversational query rewriting, and showed that even with very limited training data it could achieve very good performances. Finally, transformers have been used for ad hoc retrieval [17, 43, 52, 66], the latest works showing that the transformer-based architectures are outperforming state-of-the-art adhoc models. Dai et al. [17] analyzed the attention weights of BERT to explain its performance in retrieval, but restricted their study to some selected examples.

For the query suggestion task, Garg et al. [23] presented a Hierarchical Transformer that outperforms RNN-based model, and thus showed that recurrence was not crucial for this task. Their model is composed of two encoders, namely a token-level and a query-level one. The first one gives a contextualized representation of each token that depends on the other tokens of the query, while the second one outputs a contextualized representation of each query depending on the other queries of the session. Our work extends this article by providing a thorough analysis of the behavior of (hierarchical) transformer models, as well as experimenting with various pre-trained transformer models.

3 TRANSFORMERS FOR QUERIES SUGGESTION

In this section, we first present the transformer network architecture before describing how we use it for query suggestion.

3.1 The Transformer Architecture

The transformer architecture was introduced in [63]. 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 a sequence x composed of n vectors $x_1, \dots, x_n$ into a sequence $y_1, \dots, y_n$ of the same length, through an attention over a context sequence c composed of n vectors $c_1, \dots, c_n$. 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 [63], but here we focus on the attention mechanism since it is important for our analysis (Section 5).

Attention heads and transformations. At each layer of the encoder or the decoder, the transformation function T is based on the output of a series of H attention-based functions A_h (called *heads*). For each head A_h, the attention mechanism relies on:

- keys $k_h(c_j) \in \mathbb{R}^{d_k}$ computed for each element of the context c_j
- values $v_h(c_j) \in \mathbb{R}^{d_k}$ computed for each c_j
- queries $q_h(x_i) \in \mathbb{R}^{d_k}$ computed for each input $x_i \in \mathbb{R}^d$, with $d = H \times d_k$.

Each input is decomposed in H parts of the same dimension d_k, i.e., $x_i = (x_{1i} \oplus \cdots \oplus x_{Hi})$ where $\oplus$ is a vector concatenation operation. Each x_{hi} is modified by a linear combination of the values $v_h(c_j)$ based on weights derived from the match between the query $q_h(x_i)$ with the different keys $k_h(c_j)$. More formally, we define a head A_h as:

$$A_{hi}(x, c) = x_{hi} + \sum_{j=1}^{m} \underbrace{\alpha_{hij} v_h(c_j)}_{\beta_{hij}(c_j)} \text{ with } \alpha_{hij} \propto \exp\left(\frac{1}{\sqrt{d_k}} q_h(x_i) \cdot k_h(c_j)\right), \tag{1}$$

where we can see that the attention mechanism only modifies the input if both the attention α_{hij} and the value $v_h(c_j)$ are not null. Each key, query, and value function is unique to a given layer and head, but is the same for each input vector. The output of the layer is given by $T(x, c) = (T_1(x, c), \dots, T_n(x, c))$ with

$$y_i = T_i(x, c) = f\left(A_{1i}(x, c) \oplus \cdots \oplus A_{Hi}(x, c)\right),$$

where f is a normalization followed optionally by a feed-forward layer.

The full transformation performed at layer l for a part $\bullet$ of the model is denoted as $T_l^\bullet$ in the following. The parameters of the corresponding heads (queries, keys, and values) are specific to each $T_l^\bullet$, where $\bullet$ is either the encoder self-attention $\mathrm{e} \rightarrow \mathrm{e}$, the decoder self-attention $\mathrm{d} \rightarrow \mathrm{d}$, or the decoder to encoder attention $\mathrm{e} \rightarrow \mathrm{d}$ (see below).

Encoding. When encoding, i.e., processing the input sequence $s^{(0)}$ of token embeddings $s_1^{(0)}, \dots, s_n^{(0)}$, each layer transforms a sequence $s^{(l-1)}$ into $s^{(l)}$ using the transformation $T_l^{\mathrm{e}\rightarrow\mathrm{e}}(s^{(l-1)}, s^{(l-1)})$ based on the heads $A_{hi}^{\mathrm{e}\rightarrow\mathrm{e}}$ ($e \rightarrow e$ for "attention from the **e**ncoder on the **e**ncoder"). Since the context is simply the input here, this is called a *self*-attention mechanism—i.e., each input item, representation is transformed by looking at the whole input sequence. This is repeated L_e times until obtaining the final representation of the encoded sequence $s^{(L_e)}$ which has the same length as the original input, but where each representation is *contextualized* depending on the other tokens of the input.

Decoding. The generating process (called decoding) is based on the same principle—with a small twist since we take into account not only the already generated sequence, but also the input. To compute the probability of generating a new token w given the sequence $w_0, w_1, \dots, w_{n'}$, whose embeddings are $t_0^{(0)}, \dots, t_{n'}^{(0)}$, the decoder uses two attentions: one self-attention $A^{d\rightarrow d}$ (decoder to decoder attention) followed by an attention on the encoded sequence $A^{d\rightarrow e}$ (decoder to encoder attention). The representation at layer l is based on the representation at layer $l-1$ and on the final encoded sequence:

$$d^{(l)} = T_l^{\mathrm{d}\rightarrow\mathrm{e}} \left[T_l^{\mathrm{d}\rightarrow\mathrm{d}} \left(t^{(l-1)}, t^{(l-1)} \right), s^{(L_e)} \right].$$

The process is repeated L_d times, giving rise to the representations $t_1^{(L_d)}, \dots, t_{n'}^{(L_d)}$. The distribution over the next token w (whose embedding is t) is then given by a parametric function applied to the representation of the last previously generated output $t_{n'}$ (which is why there is a token w_0 corresponding to "[START]"—in order to compute the first generated token):

$$p(w|w_1, \dots, w_{n'}) = g(t; t_{n'}^{(L_d)}). \tag{2}$$

3.2 Pre-Trained Transformers

Transformers models have a large number of parameters which make them costly to train. In addition to that, the attention mechanism is computationally expensive, particularly for long sequence: it has a complexity of $O(n^2)$ with respect with the sequence length [64]. Thus they are complex to train. Fortunately, multiple pre-trained models trained on large datasets have been released recently [21, 39, 53, 67]. We compare the results of transformers trained from scratch, to three pre-trained models that we finetune, namely BERT [21], BART [39], and T5 [54].

Bert. The BERT [21] has been trained on a large dataset, the BooksCorpus [70] 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 is made of an encoder and a decoder. It is trained on the same data than BERT, but on multiple tasks: token masking, token 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.

T5. T5 [54] is also a transformer with an encoder and a decoder as described in [14] with minor architecture modifications in the attention. T5 is trained simultaneously on multiple tasks, that's why the author called it a "unified" framework. The task is specified by adding the task name as a prefix in the original input. The network is the same for all inputs, while usually the mutli-task learning model have a specific network for each task [41].

Note that many pre-trained transformers have been released in recent years (BERT [21], BART [39], GPT-2 [53], T5 [54], XML [16], RoBERTa [42], and the famous GPT-3 [10]—whose parameters have not been made public), so it is necessary to choose those we want to experiment with. We choose: (1) BERT because it is the most used transformer; (2) BART because it has an encoder-decoder architecture with very good performance in generation, and especially in summarization, and finally; and (3) T5 because it is one of the last transformers that have been published.

3.3 Using Transformer Networks for Query Suggestion

3.3.1 Problem Setting. Let us consider a session $S = (Q_1, \ldots, Q_{|S|})$ as a sequence of $|S|$ queries, where every $Q_i = (w_{i,1}, \ldots, w_{i,|Q_i|})$ 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:

$$\mathcal{L}(S;\theta) = \sum_{(S,\check{Q})} \log p_\theta(\check{Q}|S) = \sum_{(S,\check{Q})} \sum_{t=1}^{|\check{Q}|} \log p_\theta(w_t|Q_1, \ldots, Q_{|S|}), \tag{3}$$

where $(w_1, \ldots, w_{|\check{Q}|})$ 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. The model is illustrated by Figure 1.

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 [63].

We obtain a contextualized representation for each token of the session with the Encoder E:

$$\mathrm{E}(S) = (h_0, \ldots, h_n), \tag{4}$$

where n is the number of tokens in the whole session: $n = \sum_i |Q_i|$.

We train models with various encoders E described in the next Sections (from 3.3.2 to 3.3.5). The decoding part is the same for all, as described in Section 3.1.

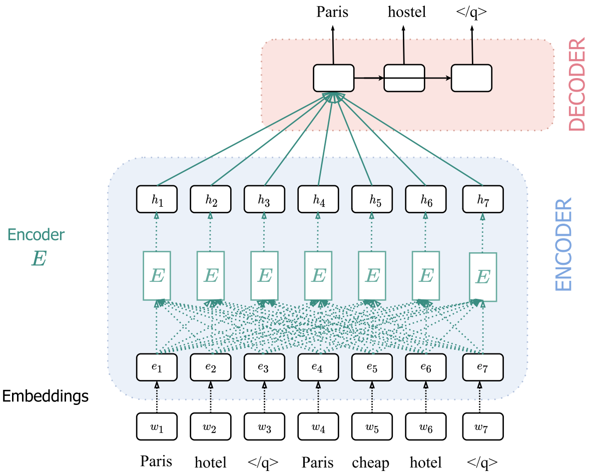

Paris
hostel
</q>
DECODER
ENCODER
Encoder
E
h_1 h_2 h_3 h_4 h_5 h_6 h_7
E E E E E E E
e_1 e_2 e_3 e_4 e_5 e_6 e_7
Embeddings
w_1 w_2 w_3 w_4 w_5 w_6 w_7
Paris
hotel
</q>
Paris
cheap
hotel
</q>

Fig. 1. Flat Transformer for Query Suggestion.

3.3.2 Fully Trained Transformer (TS). The encoder and a decoder and the decoder are fully trained, they have the architecture described in Section 3.1, with $L_d = 6$ layers, with $H = 12$ heads each and a dropout $p = 0.1$. On the top of the decoder, we use a feedforward network with a hidden size of 2,048. For the input tokens, we use the same embeddings for the encoder and the decoder to reduce the number of parameters and to regularize the network [63].

3.3.3 BERT. We use the pre-trained model BERT [21], and extract each hidden layer of the model. We sum the last layer, with the average and the max of these layers.[1] For each token of the input, we have a contextualized embedding of size 768 given by BERT. For the decoding part, we use the same transformer decoder and feedforward network as the ones described in 3.3.2. At the beginning of the training the encoder is frozen and the decoder is trained. We then use a "gradual unfreezing" of the encoder layers as recommended by [28]: when the loss stabilizes, we unfreeze the last frozen layer of the encoder, until all the layers are fine-tuned.

3.3.4 BART. The architecture is complete for text generation, it has an encoder and a decoder. We also use gradual unfreezing to fine-tune the model, but starting from the last layer of the pre-trained decoder. We compare the results of the complete BART model fine-tuned for our task, with the ones of the BART Encoder followed by a fully trained Transformer Decoder.

3.3.5 T5. T5 is a transformer with a pre-trained encoder and a pre-trained decoder. As we did for BART, we compare two versions of the model: the encoder-only version *Enc_T*5, with a fine-tuned encoder and a fully trained decoder, and a version for which we fine-tuned the entire T5 model. We use the training protocol described for BART and BERT.

3.4 Hierarchical Transformer for Query Suggestion

We now describe the hierarchical transformer proposed by [23] (an illustration is given in Figure 2). It is composed of two levels of encoding: a token-level E_T and a query-level one E_Q, each following the same contextualization process as a standard encoder in a transformer model.

First, the token-level Encoder E_T produces a contextualized representation $\mathrm{E}_T(Q_i) = (\tilde{w}_{i,1}, \ldots, \tilde{w}_{i,K})$ of each token of a given query Q_i. Since queries might have a different length, padding is

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

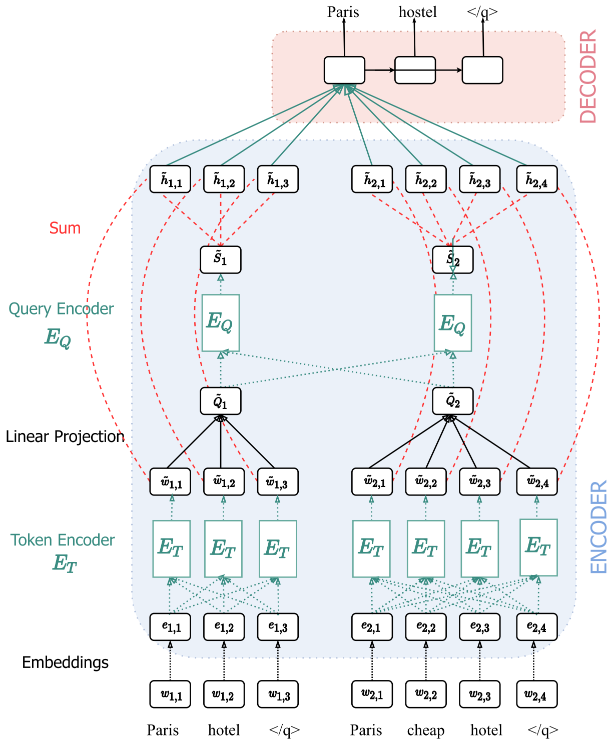


Fig. 2. Hierarchical transformer for query suggestion.

used (e.g., a special [BLANK] token) so that each query is of length K. This representation is then summarized into a query representation $\tilde{Q}_i$ using a linear transformation:

$$\tilde{Q}_i = \mathrm{E}_T(Q_i)\mathrm{W}_P. \tag{5}$$

The transformation matrix is $\mathrm{W}_P \in \mathbb{R}^{Kd \times d}$ where d is the output dimension of each token of the encoder. In our experiments we use $K = 12$, which is enough to cater for most of the queries—the remaining tokens are truncated.

The session-level encoder takes these vectors $\tilde{Q}_i$ as input to transform them into final query representations $\tilde{S} = (\tilde{S}_1, \ldots, \tilde{S}_{|S|})$ that embed context from neighbor queries, using positional encoding.

$$\tilde{S} = \mathrm{E}_Q\left(\tilde{Q}_1, \ldots, \tilde{Q}_{|S|}\right), \tag{6}$$

where $|S|$ is the number of queries in the session.

We then obtain the final representation of a query token by summing its query-wise representation $\tilde{w}_{i,j}$ with the contextualized representation of its corresponding query $\tilde{S}_i$:

$$\tilde{h}_{i,j} = \tilde{w}_{i,j} + \tilde{S}_i. \tag{7}$$

Finally, the decoding part is exactly the same as for other transformer models (Section 3.1).

4 EXPERIMENTS

In this section, we report experimental results comparing the various flat and hierarchical transformer-based models, as well as other neural network baselines.

We first describe the datasets, the compared models and the metrics (Sections 4.1–4.3), before presenting our main results in Section 4.4. In Section 4.5, we pursue our analysis by studying how the models perform when exposed to noise, by altering the sessions (filtering or concatenating). In both cases, we show that hierarchy does not help as much as a good pre-training. Finally, in Section 4.6, we present some queries generated by a selection of models.

4.1 DataSets.

Some datasets allow a fine evaluation of query suggestions, they consist of queries grouped by user sessions and associated with relevant documents. These datasets are: the TREC Session dataset [13] which contains the names of the tasks and relevant documents associated with the user sessions, the conversational dataset SCSdata [62] segmented by task and containing the documents read by the user, and the Webis-SMC-12 dataset [24] which is a subset of AOL for which the sessions have been manually split and annotated into missions. However, these three datasets contain few sessions, respectively, 1,300, 1,000, and 2,200 sessions, which is insufficient to train the models we want to compare. To the best of our knowledge, there is no dataset of sufficient size better suited to the task of suggesting queries than the two query logs datasets: the real dataset AOL web search log and the artificial dataset, MS MARCO Conversational Search [47]. In both cases, the queries are processed by removing all non-alphanumeric characters and lowercasing following [60].

MS MARCO is an artificial dataset, built from real queries. The authors filtered these queries: they removed navigation, bot, junk, and adult sessions and merged users queries with a nearest neighbor search based on their embeddings to create artificial sessions. The MS MARCO dataset is provided in two parts. We use 80% of the first part as the training set, the remaining 20% as the validation set, and the second part of the dataset as the test set. Each set contains, respectively 540, 267, 135, 066, and 75, 193 sessions.

The AOL dataset consists of 16 million real search log entries from the AOL Web Search Engine for 657,426 users. Following [60], we delimit sessions with a 30-minutes timeout for both datasets. 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 [60]. 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. As the real-word AOL dataset is not filtered, it contains typos and noisy sessions. It is made of 860, 155 unique words, whereas the artificial dataset MS MARCO has 28, 968. When building a vocabulary same manner as in [60] (i.e., using the most frequent 90 k words of the training set), 8.9% of the words from the dataset are not in the vocabulary while all MS MARCO words are included in the selected vocabulary.

4.2 Compared Models.

In our experiments we compare a co-occurence based approach, two RNN-based approaches and fully trained and fine-tuned transformer models. The co-occurence based approach is the Inverted Index [9], RNN models are HRED [60], and ACG [20], which we described in Section 2. The fully trained transformer, hereafter referred as TS, is composed of an encoder and a decoder presented in Section 3. The hierarchical transformer H_TS with the two-level encoder is described in Section 3.4. The pre-trained models that we finetune are Bert [21], Bart [39], and T5 [54].

The two RNN-based models and the fully trained transformers TS and H_TS use a fixed vocabulary composed of words, but Bert, Bart, and T5 employ subword tokenizers (WPT) that segment the text into n-grams of varying lengths [57]. For instance, the query “Robert Mitchum” is segmented as `robert [UNK]` with a Word Tokenizer while the WPT returns `robert mitch ##um`. Hence, there is no OOV problem (handled with special OOV token) with the WPT and the vocabulary size is kept below a pre-defined threshold (31 K tokens for Bert, 32 K for T5, and 50 K for BART), which in turns speeds up learning. To analyze the importance of the tokenizer, we consider variants of HRED, ACG, TS, and H_TS based on the Bert tokenizer in our experiments, named HRED-WP, ACG-WP, TS-WP, and H_TS_WP.

To leverage pre-trained models, which is especially important since the number of parameters in transformer models is high, we use the parameters of Bert [21], Bart [39], and T5 [54] to initialize the parameters of our models. More precisely, for the flat architecture (TS), the encoder parameters are either initialized to those of the Bert model, the Bart or t5 encoder. The models are named, respectively, BERT, Enc_BART, and Enc_T5. Since Bart and T5 are not only an encoder as Bert, we also consider a version with both encoder and decoder parameters initialized with pre-trained Bart and T5 parameters, that we refer, respectively, to BART and T5.

For the hierarchical architecture (H_TS), the Query Encoder E_T parameters can also be initialized with those from the Bert, Bart, and T5 encoders, the rest of the architecture remaining trained from scratch. We refer to such models as H_BERT, H_BART, and H_T5.

For all models involving pre-trained transformers, the training procedure is the same: we use the "gradual unfreezing" method, as recommended by [28] and described in Section 3.3.3.

Models optimization is performed on the training sets of sessions with the ADAM optimizer [35]. All hyper-parameters are tuned via grid-search on a validation dataset.

4.3 Metrics

As many other tasks in IR, evaluating the quality of the models is problematic since they can generate many queries in response to a session—and there is no principled way to evaluate their quality. In the following, we describe the metrics that were reported in previous works to compare models, and which try to capture the quality of the system responses.

Perplexity. All compared models generate probability distributions over the sequences. This enables to check how surprised the model is by the target query. However, perplexities of some pairs of methods cannot be compared because the vocabulary size is different (90 K tokens for models without WPT, 31 K tokens with WPT, 50 K for BART's tokenizer, and 32 K for T5). Moreover, former versions of HRED, ACG, TS, and H_TS can generate OOV words, which strongly biases the results. Perplexity is not reported for these last methods.

Query suggestion metrics. As a metric to evaluate generated queries compared to the target ones, we first use the classical metric BLEU [51], 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, $\text{Sim}_{extrema}$ [22], 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 [60]). 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. For each generation metric, we consider the maximum performance of the top-10 queries generated by the models. More precisely, for each model, we first generate (through a beam search

Table 1. Results on the MS MARCO (a) and the AOL DataSet (b)

(a) MS MARCO dataset

	II	ACG	ACG_WP	HRED	HRED_WP	TS	TS_WP	H_TS	H_TS_WP	BERT	H_BERT	Enc_BART	BART	H_BART	Enc_T5	T5	H_T5
EM	0.173	0.044	0.041	0.139	0.129	0.174	0.197	0.164	0.170	**0.223**	0.182	0.184	**0.226**	0.183	0.175	0.203	0.121
BLEU 1	0.584	0.435	0.416	0.572	0.555	0.579	0.596	0.574	0.589	**0.617**	0.597	0.591	**0.618**	0.592	0.598	0.576	0.565
BLEU 2	0.369	0.200	0.182	0.341	0.320	0.372	0.377	0.363	0.371	0.402	0.378	0.385	**0.419**	0.383	0.379	0.375	0.335
BLEU 3	0.218	0.092	0.087	0.193	0.176	0.223	0.248	0.218	0.224	**0.274**	0.234	0.238	**0.275**	0.236	0.230	0.238	0.174
BLEU 4	0.202	0.073	0.068	0.175	0.161	0.213	0.239	0.201	0.206	**0.268**	0.217	0.222	**0.266**	0.221	0.212	0.231	0.149
$sim_{extrema}$	0.835	0.798	0.780	0.828	0.817	0.833	0.840	0.834	0.837	**0.846**	0.839	0.837	**0.848**	0.839	0.837	0.837	0.830
$sim_{pairwise}$	0.677	0.579	0.543	0.635	0.616	0.671	0.682	0.665	0.670	**0.697**	0.677	0.672	**0.697**	0.678	0.675	0.659	0.661
New Words	0.950	0.138	0.354	0.594	0.604	0.886	0.880	0.902	0.899	0.870	0.902	0.902	0.858	0.911	0.879	0.910	0.895
Repetion Rank	8.618	8.767	9.429	8.974	9.141	6.926	6.689	7.055	7.022	6.424	6.755	6.985	5.586	7.098	7.116	6.913	7.318

(b) AOL dataset

	II	ACG	ACG_WP	HRED	HRED_WP	TS	TS_WP	H_TS	H_TS_WP	BERT	H_BERT	Enc_BART	BART	H_BART	Enc_T5	T5	H_T5
EM	0.018	0.017	0.010	0.029	0.036	0.037	0.048	0.046	0.081	0.061	0.085	0.055	**0.119**	0.087	0.052	0.082	0.053
BLEU 1	0.438	0.417	0.388	0.409	0.422	0.439	0.454	0.447	0.493	0.460	0.495	0.455	**0.552**	0.494	0.452	0.519	0.435
BLEU 2	0.148	0.128	0.098	0.122	0.135	0.162	0.178	0.178	0.238	0.194	0.241	0.186	**0.316**	0.240	0.183	0.275	0.166
BLEU 3	0.067	0.037	0.026	0.052	0.059	0.071	0.089	0.102	0.146	0.110	0.150	0.104	**0.231**	0.144	0.098	0.192	0.090
BLEU 4	0.033	0.006	0.004	0.018	0.023	0.027	0.040	0.055	0.086	0.063	0.093	0.058	**0.174**	0.084	0.051	0.148	0.043
$sim_{extrema}$	0.751	0.668	0.687	0.710	0.713	0.729	0.723	0.742	0.762	0.741	0.763	0.739	**0.792**	0.762	0.731	0.776	0.723
$sim_{pairwise}$	0.484	0.408	0.390	0.404	0.415	0.447	0.457	0.462	0.501	0.466	0.504	0.459	**0.558**	0.499	0.454	0.537	0.435
New Words	0.996	0.119	0.588	0.679	0.740	0.916	0.941	0.849	0.881	0.927	0.880	0.919	0.682	0.934	0.902	0.593	0.940
Repetion Rank	9.711	7.138	9.128	7.841	7.157	8.683	8.300	6.830	4.970	6.668	4.203	6.132	2.204	3.665	6.203	1.468	6.324

We Report Different Metrics, along with Two Quality Indicators Best Results for a Metric Are Reported with a Bold Font. Bold values indicates significant gains ($p < 0.05$) compared to T5.

with $K = 20$) 10 queries to suggest to the user given the context.[2] 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., [37]) and corresponds to a fair evaluation of models that try to find a good balance between quality and diversity.

4.4 Results

In this section we aim at answering our first question: **Q1. How well the various presented transformers generate queries suggestions compared to the usual baselines?**

Tables 1 (generation scores), and 2 (perplexity) report results obtained by all the models. We also added two further indicators. First, the ratio of new words (New Words), calculated by counting the number of unique words that appear in the suggested query but were not in the past queries of the session, divided by the count of unique words in this query. Second, the rank of the prediction in the beam search (Repetition Rank) if the predicted query appears in the context (or 10 if it does not).

We first note the difference between the two datasets. As expected, being synthetic, MS Marco is a much easier dataset—more restricted vocabulary and more regular sessions, as acknowledged by the fact that all the metrics are higher for MS Marco.

From a high level point of view, we see that transformers are better performing than the baseline II and that the RNN-based models, HRED, and ACG. Among transformers, more complex and pre-trained models perform better, with the flat architecture with a pre-trained encoder and decoder Bart performing the best. Contrarily to [23], we do not observe a real difference between hierarchical and non hierarchical transformer architectures: The main factor of variation is on what task and dataset the model was pre-trained.

We note that models have different tendencies to copy one of the queries in the session. This is a standard behavior: 3% of queries for MS MACRO and 6% for AOL are among the previous queries of the session. So it is not surprising that more powerful models learn to copy—transformer models

[2] As we want to encourage the models trained with a word tokenizer to generate tokens present in the vocabulary, we follow [34] and apply a penalty on the "OOV" token in the beam search. To compute the metrics, we ignored the OOV token that can be generated by HRED or ACG—queries composed only of OOV words are skipped.

Table 2. Perplexities for Word-Piece Tokenizer-Based Models

	AOL	MS MARCO
ACG WP	1 175	242
HRED WP	1 101	111
TS WP	721	56
H_TS WP	486	56
BERT	492	47
H_BERT	473	64
Enc_BART	557	52
H_BART	209	40
BART	173	39
Enc_T5	92	22
H_T5	215	58
T5	37	21

have a tendency to repeat a seen query compared to ACG or HRED (lower Repetition Rank). We explain this tendency by their ability to retrieve information at arbitrary positions in the input.

Perplexity. We only compare perplexity for models based on the same tokenizer, since otherwise the problem of evaluating prediction with OOV tokens, or of vocabulary with different sizes makes comparisons impossible. We observe that the transformers obtain a much better perplexity than ACG and HRED with WPT. The likelihood of target queries with these last two methods are both about half the one of the transformer model TS_WP. This shows that transformers better explain users' behavior in search sessions. Among transformers, we observe that while the hierarchy is beneficial on the AOL dataset, it is not the case on the MS MARCO dataset. We will discuss this behavior in more details later.

Word Piece Tokenizer. Among RNNs, using WPT is sometimes beneficial for HRED but not for ACG. We explain this because the copy mechanism already allows ACG to produce rare tokens. This ability appears lowered when using word pieces, as assembling unknown words from smaller tokens is much more difficult than copying a whole word for such architectures. For HRED, the Word Piece Tokenizer improves the scores on the AOL dataset, while it degrades them on the MS MARCO one. This is explained by the fact that for the MS MARCO dataset there is no OOV and hence using a WPT is not useful anymore.

For Transformers trained from scratch (TS, TS_WP, H_TS, and H_TS_WP), the Word Piece tokenizer is always beneficial. It could be due to the use of positional embeddings, that makes the copy of consecutive tokens easier. Moreover, the use of this tokenizer reduces the vocabulary size.

The pre-trained models. First, BART (flat transformers with a pre-trained encoder and decoder) outperforms all the models on all metrics. This shows the value of pre-trained models on large dataset and on generative tasks (summarization). When observing the flat pre-trained models scores, we note that they outperform the fully trained version: BERT, Enc_BART, BART, Enc_T5, and T5 are better than TS_WP on the AOL dataset. For the MSMARCO dataset, while BERT and BART have better scores than TS_WP, Enc_BART, and Enc_T5 are similar to TS_WP. We think that because the vocabulary used in the MSMARCO dataset is more restricted, and the dataset more regular, the use of large pre-trained models is less beneficial. While T5 largely outperforms BERT on the AOL dataset, BERT is much better than T5 on the MSMARCO dataset. The unified framework–consisting of training simultaneously the model for various tasks—used to pretrain T5 is useful on a complex dataset, as it probably allows the model to acquire more language knowledge, but it is less efficient on simpler data. Finally, for both datasets, BART performs the best for all metrics. On the AOL dataset, BART improvement is particularly important on BLEU 3 and BLEU 4—which are calculated by considering 3-gram and 4-gram sequences. It indicates that when comparing longer word sequences between target and predictions, BART is the best model, thus it is better at generating longer queries, i.e., longer queries. We think this is because BART has been trained on a summarization task, and is therefore better than the other models at generating comprehensive sequences.

Its scores are also significantly better on the similarity scores $sim_{extrema}$ and $sim_{pairwise}$ on the AOL dataset, which means that this is the best model to capture the word semantic.

The Hierarchy. On the AOL dataset, the hierarchical models perform better than their flat version: TS vs H_TS, TS_WP vs H_TS_WP, BERT vs H_BERT, and Enc_BART vs H_BART except for T5 for which Enc_T5 outperforms H_T5. This could be due to the fact that T5 uses relative positional embeddings, while other models use absolute positional embeddings. H_T5 would have more difficulties to find the exact position of words within queries. Note that for fair comparison H_BART and H_T5 are compared to Enc_BART and Enc_T5 rather than BART and T5 because BART and T5 decoders are pre-trained while H_BART and H_T5 decoders are trained from scratch. This shows that with a suitable encoder the hierarchy is beneficial for the query suggestion task, the two-levels encoder allowing to have a more complex representation of the session.

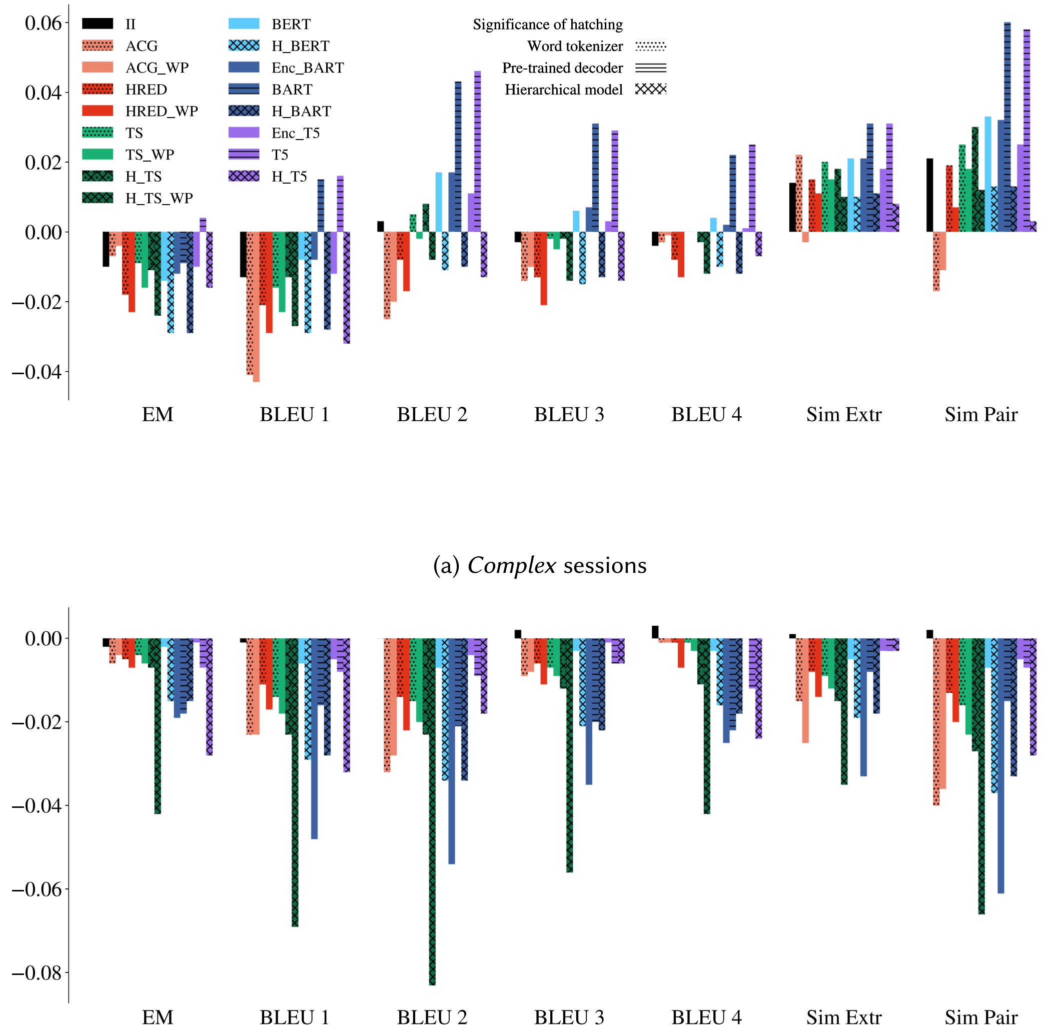
The conclusions are different for the MSMARCO dataset. For the fully trained model TS and TS_WP, and for BART, the hierarchy does not help significantly, while with BERT and T5, the hierarchy decreases the results. We explain this because the queries and the sessions of the MSMARCO dataset are longer, and the model has difficulty to focus its attention on the important queries. We discuss the behavior of the hierarchical models on longer and more complex sessions more in detail below.

4.5 Robustness of (Transformer) Models

We now look more in details in how the models behave regarding different types of sessions to answer the second question **Q2: Which model is the most robust to complex sessions (a), to noisy sessions (b) and to long sessions (c)?** For each type of session, a section is dedicated to the answer.

(a) Transformers results on complex sessions. Focusing on the real-word dataset AOL, which contains many very short and simple search sessions typical of web search, we were interested in how transformer models could handle complex sessions. To identify those, we used a simple heuristic: a complex session (1) consists of at least three queries; (2) contains queries with more than one word; and (3) should not contain spelling corrections. For (3), we used the following heuristic: each of its queries must be sufficiently different from the previous one, i.e., its editing distance (in characters) should be greater than 3.

Figure 3(a) reports the relative results obtained on this subset of 193, 336 complex sessions. In particular, we want to compare the results of the flat and of the hierarchical models. We note the good behavior of pre-trained flat transformers for query suggestion for the complex search task, while it emphasizes the weakness of the pre-trained hierarchical models on these sessions. The flat models improve the results on these sessions over the corresponding hierarchical model on all metrics: BERT is less deteriorated than H_BERT, and likewise BART and Enc_BART are less impacted than H_BART by the complexity of the sessions, and the same is true for T5 models. For



(a) *Complex* sessions

(b) *Concatenated* sessions

Fig. 3. Difference between the performance on all the AOL sessions and on the noisy version (filtered/ concatenated). Negative values indicate a degradation.

the fully trained models, TS_WP is also less impacted that H_TS_WP on this subset of sessions on all metrics. This shows again the robustness of flat models.

(b) Results on noisy 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 (in average) is not the same as the intent driving the users' behavior when formulating test queries, models must have learned to identify thematic breaks, and to ignore this noisy information.

Figure 3(b) shows percentages of performance loss for every metric. We can see that for all models, the flat architectures are much less impacted than their corresponding hierarchical counterpart. This is an important result, since the test sessions were arbitrarily split according to a 30-minute timeout, which might not correspond to users' intent changes. It shows that with the hierarchy, the transformers lose their ability to focus on relevant part, and so to adapt themselves to longer sessions.

(c) Sessions Lengths. We study the impact of the sessions lengths on the two pre-trained models BERT and BART (flat and hierarchical versions) on the AOL dataset. Results are reported in Figure 4. Whatever the metric, the hierarchical models (in green) perform better than the flat ones (in red) for short sessions. However, for longer sessions (above seven queries), it is the other way around. The flat models scores remain stable while the scores of the hierarchical models decrease. The hierarchical architecture of [23] is adapted to short and more simple sessions search, but for longer and complex tasks the flat transformers are more suitable. We believe that this is due to the fact that hierarchical transformers cannot focus reliably on the relevant parts of the session.

4.6 Generated Queries

Finally, in Table 3, we give examples of query suggestions for three sessions, and multiple models: HRED_WP (which is the best among the RNN baselines), the fully trained transformers TS_WP and H_TS_WP, and the pre-trained ones Enc_BART, H_BART, and BART.

First, we note that the RNN-based model HRED_WP generates the same word several times in a row. This behavior is very common for HRED_WP. For the session presented in the first column, it suggests "divorce groups groups", for the second "maryland hotel hotel ocean", and for the third "disney resorts resorts". Note that this is something the transformer models never do. Moreover, HRED_WP does not introduce new words, it reformulates the queries of the context by mixing words order. On the contrary, the transformer models proposed more diverse suggestions.

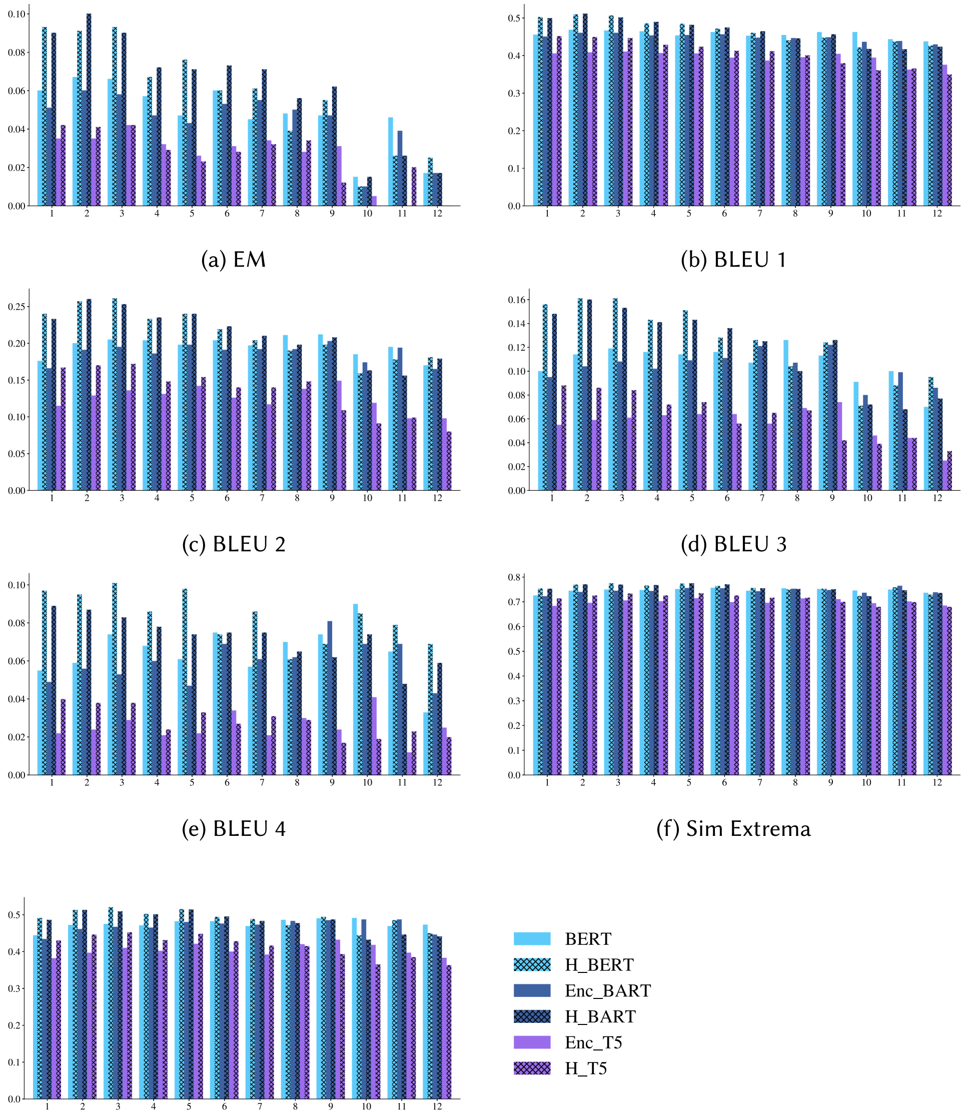
We note that the hierarchical models have a greater tendency to copy words from the context compared to their flat version (we study this behavior in the next section). H_TS introduces only one new word ("free") in the suggestions of the first session, while TS_WP proposes several new themes ("listings", "ebay", and "aol"). The second presented session contains a typo: "marylandocean" instead of "maryland ocean" with a blank space. The hierarchical H_BART did not succeed to correct this typo, it proposes "marylando ocean city" because it is more willing to copy words from the context, and thus a part of this typo, while the flat transformer models did not.

The pre-trained models BART and H_BART propose more diverse suggestions compared to the fully trained models. In the session of the third column, the user performs queries on several topics of the same subject. While the various models succeed to integrate the diverse themes in the suggestions, the pre-trained models introduced more new topics : "texas", "hotels", and "ebay". Finally, we notice that the suggestions of BART tend to be longer than the ones of the other models, confirming the experimental results shown earlier.

5 TRANSFORMER FOR QUERIES SUGGESTION ANALYSIS

We now investigate the behavior of this latter model BART and design experiments to answer the last question **Q3: How does the flat transformer generate queries?**

Several papers propose to analyze transformers to check which information is learned or used [11, 15, 32] through either probing different parts of the layer, or by looking at the attention toward the input [15]. In this section, we follow this latter line of work, focusing on specific properties of transformers for query generation.



(a) EM

(b) BLEU 1

(c) BLEU 2

(d) BLEU 3

(e) BLEU 4

(f) Sim Extrema

(g) Sim Pairwise

(h) Legend

Fig. 4. Models scores depending on the length of the sessions.

To do so, we focus on the attention of the decoder toward the encoder output (see Section 3.1), i.e., the attention weights computed for $A^{\mathrm{d}\rightarrow\mathrm{e}}$. When generating the $(t+1)^{\mathrm{th}}$ token, we denote $\alpha_{lhij}^{(t)}$ the attention from the i^{th} decoder token into the j^{th} encoded token for each layer l and attention head h. To summarize this information, we: (1) average the attention over the different heads – following [15]; and (2) only look at the attention of the j^{th} output token when generating the $j+1^{\mathrm{th}}$ output token. The rationale for the latter is that the generated token at step $j+1$ mostly depends on the final representation $t_j^{(L_d)}$ of the decoder token j, as shown in Equation (2). Moreover,

Table 3. Generated Queries for Three Sessions

	Q1. divorce chat rooms Q2. divorce support groups	Q1. maryland ocean city Q2. marylandocean vity hotel	Q1. carobean cruises Q2. spa resorts Q3. disney world
HRED_WP	- divorce support groups	- maryland hotel hotel	- disney world resorts
	- divorce chat groups	- maryland hotel ocean	- disney resorts resorts
	- divorce divorce groups	- maryland hotel hotel ocean	- disney world
	- divorce groups groups	- maryland hotel	- disney vacation resorts
	- divorce support	- maryland hotel ocean ocean	- disney resorts
TS_WP	- chat room listings	- ocean city maryland	- disney world
	- ebay	- ocean city md	- travelocity
	- aol chat	- mapquest	- disney world hotels
	- chat rooms	- ocean county maryland	- disney world cruise
	- divorce chat room	- expedia	- disney cruise
H_TS_WP	- divorce support groups	- maryland ocean city	- disney world
	- free divorce support groups	- ocean city maryland	- sea world
	- divorce	- hotels in maryland	- disneyworld
	- divorce chat rooms	- hotel ocean city	- carnival cruise
	- divorce support	- mapquest	- spa resorts
Enc_BART	- divorce chat rooms	- maryland hotel	- disney world
	- divorce chat room	- maryland hotels	- disney world cruises
	- divorce support group	- mapquest	- disney world texas
	- divorce support	- maryland	- disney world hotels
	- divorce chat	- maryland beach hotel	- disney world resort
H_BART	- divorce support groups	- maryland ocean city	- disney world
	- divorce	- marriott hotels	- spa resorts
	- free divorce chat rooms	- marylando ocean city	- disney world cruise
	- divorce help	- marriott	- disney world resorts
	- free divorce help	- mapquest	- ebay
BART	- divorce chat rooms	- maryland ocean city hotel	- disney world
	- divorce support groups	- maryland ocean city	- spa resorts
	- free divorce support groups	- maryland ocean city hotels	- disneyworld
	- divorce chat room	- maryland ocean town hotel	- disney world cruise
	- free divorce chat rooms	- maryland ocean city resort	- disney world hotels

The two first queries of the sessions are given in the top of the Table (Q1 and Q2), and the first five suggestions of each model reported below.

we observed that the attention did not vary much during the generation process, and hence those values are close to their average. We denote those averaged and picked attentions of token i on token j at the layer l as $\tilde{\alpha}_{lij}$.

Finally, as shown in [11], the attention weight might not be a reliable indicator in all cases, since the actual modification of the representation depends on the value $v_h(s_i^{(L)})$ as shown in Equation (1). To cater for this problem, we define the *importance* (of an attention) $\beta_{lhij}^{(t)}$ as $\alpha_{lhij}^{(t)}\|v_{lh}(s_i^{(L)})\|$. As for the attention, we summarize those values as $\tilde{\beta}_{lij}$. Unless specified, we focus on results for BART —but most of the behavior is shared by the different versions of the transformers we analyzed.

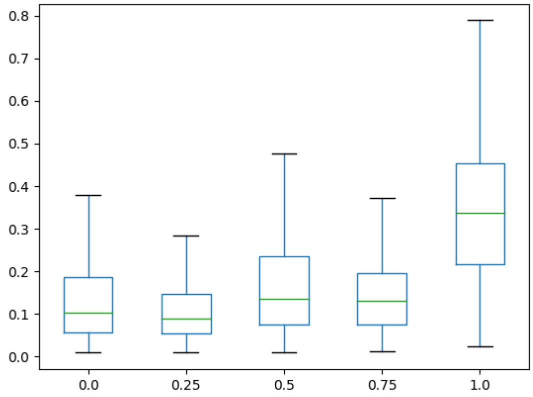


Fig. 5. Importance of the queries depending on their (normalized and using quantiles) positions in a session (average over layers).

5.1 The Growing Importance of Queries

In this section, we will answer the first sub-question **Q3. (a) On which context's queries does the flat transformer focus its attention?**

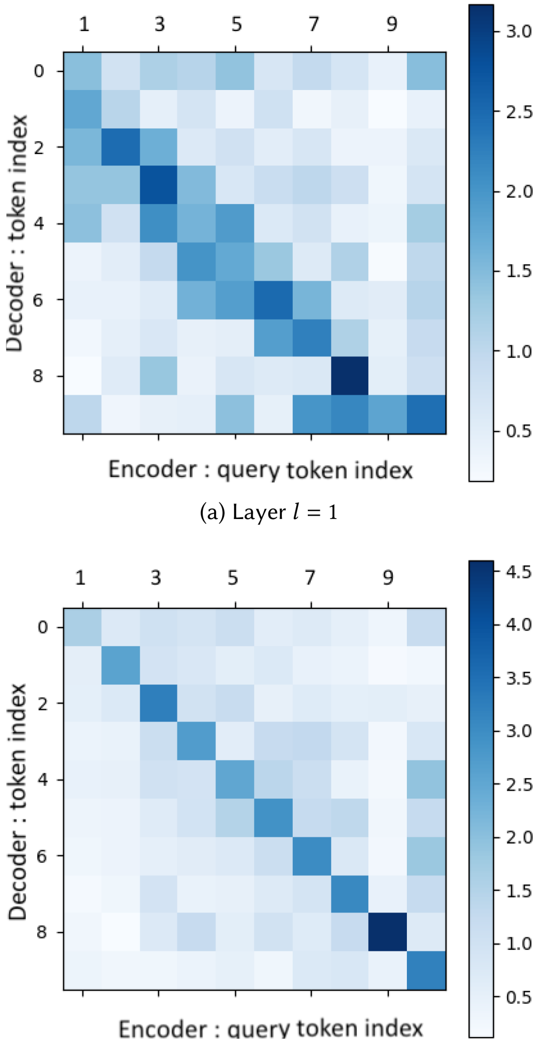
Sordoni et al. [60] claim that the last query—which they called the anchor query—plays a crucial role in queries suggestions. We verify this claim by assessing whether more attention was paid to the last queries in a session or not. For long enough sessions (≥ 5 queries), and for each query, we first sum the importance $\tilde{\beta}_{lij}$ over its tokens, and normalize the value by dividing it by its maximum value, so that we can average sessions of varying length. For the same reason, we normalize the index of each query by the length of the session, i.e., $i/|S|$. In Figure 5, we plot the boxplot of the importances given the normalized index of the query in the session. The x-axis corresponds to the position of the query in the session (from left to right: from the beginning to the end of the session), and the y-axis to the importance of the query. We see that there is a trend showing that last queries are more important for the prediction of the transformers since they have more impact on the vector used for predicting the output. It also explains the robustness of BART on concatenated sessions 3(b).

5.2 The Importance of the Context's Tokens

We now answer the second sub question of **Q3. (b) On which context's tokens does BART focuses its attention?**

For each decoded token (including the special token START numbered 0), we first look at the importance assigned to encoded tokens. In Figure 6, each cell (i, j) in the grid gives the importance of the jth token (of each query in the session, e.g., the second token of each query in the session is numbered "2") when decoding the ith token of the target query.

We only plot the importance for two representative layers (1 and 12), as we can distinguish two layers groups that behave similarly (not shown here: 1–4 and 8–12). We can observe that at layer 1–4, the importance focuses on tokens that match the same position (e.g., the first tokens of each query and the first decoded token). For the decoder token START (numbered 0), the importance is more broadly distributed—which is sensible since nothing has been generated so far. On layers 8–12, the importance focuses on tokens that match the *next* token position (e.g., the first tokens of each input query for START, the second tokens of each input query for t_1, and so on). This shows



(a) Layer $l = 1$

(b) Layer $l = 12$

Fig. 6. Importance of the tokens depending on their position in the queries (attention of the decoder on the encoder), for layer 1 (a) and layer 12 (b) of the encoder. The X-axis corresponds to the context— i.e., the encoder tokens (averaged over all queries), while the Y-axis corresponds to the decoder—i.e., the decoder tokens. For the decoder, 0 corresponds to the START token. For instance, from (a) we see that when generating the 3rd token (row of index 2), the attention is focused mostly on the second token, and also (but less) on the first and third ones. This is different for the same token at layer 12 (b), where most of the attention is focused on the third token of every past query. Results are averaged over 20,000 sessions.

that transformers first focus on the matching encoded token before selecting the next token to generate.

The figure also underlines that BART, even without explicit hierarchy architecture, is able to capture the basic structure of sessions, the attention being in average more focused around the matching tokens (i.e., same position) of the queries present in the context session (as shown by the diagonal in both graphs).

5.3 Generating a New Token

Finally we answer the last sub question **Q3. (c) How does the model choose the next token to generate?**

This brings interesting questions in terms of the generative process of the transformer-based architectures. For the START decoder token, the only explanation is that they first focus on the "[SEP]" encoded tokens, and then shift their attention to the next ones—relying on the position embedding that is added to the encoded token representations. For the next tokens to be generated, this is less obvious since the model could simply focus on a matching token (e.g., the decoder token "cat" matches the encoded tokens "cat"). As queries are often repeated within a session with small variations, the tokens might be in the same positions (in average) in the session queries and in the generated query. Consequently, to generate the next token, there are two possibilities: either the transformer shifts the attention towards a token to the right (position-based decision), or, the (query) language model of the decoder proposes a direction in the token space, which is then matched if an encoded token lies in this direction in the representation space.

To look into this, we used sub-sessions of the form

$$\ldots\ldots \mid \ldots \text{A B C} \ldots \mid \ldots \text{A B}$$

for which the next query to be predicted (in red) contains a bi-gram of tokens (A,B) that exists in the past queries, followed by a different token C. For example, the target query contains "black/A cat/B" and the session contains a query with tokens "black/A cat/B sold/C". We calculate the probability of generating after "black/A" in the target:

- the target token ("cat/B") with a probability $P(B|S, A)$
- the token following the bi-gram in the context ("sold/C") with a probability $P(C|S, A)$

We do this for two settings: (1) using the original context session as S and (2) using a modified context session S for which we swapped tokens B and C in the context (i.e., substituting "black/A sold/C cat/B" –"black/A cat/B sold/C"). The goal is to assess whether the model favors a **language model** (**LM**) that captured that B usually follows A, or rather a copy mechanism that mainly considers **positions from the context session** (**POS**). Following this process, the average probabilities are computed over a set of 20,000 sessions and are reported in Table 4 for the different transformers.

First, when position (in the context session) and language model agree (first and second columns), the probabilities are high for the real target and low otherwise. Among the different models, we note that the best performing models (Section 4.4) have a very high probability of generating the token B (between 0.7 and 0.8).

When position (in the context session) and language model disagree (4th and 5th column), the behavior of the architectures is quite different. Apart from the TS_WP (and to a lesser extent its hierarchical version) which mostly follows the language model (0.03 vs 0.19) and ignores the context session, we see that all the other models assign balanced probabilities to position and language in these swapped sessions.

Sufficiently powerful flat models such as BART appear sufficient to capture the query organization of sessions, while keeping enough flexibility to adapt to perturbations. We indeed observe that BART has both high probabilities of either following the language model or the position-based prediction (total probability of 0.63), which is nearly as high as when the context session and language model match (0.70). This difference with the other models might explain why BART is performing so well: it leverages both the copying mechanism and its powerful language model.

Table 4. Probabilities on Mixed and Unmixed Sessions

Session *S*	Original			B/C swapped		
	... A B C ...			... A C B ...		
probability	$p(B \mid S, A)$	$p(C \mid S, A)$	total	$p(B \mid S, A)$	$p(C \mid S, A)$	total
favors	LM/POS			LM	POS	LM/POS
Transformer WP	0.19	0.03	0.22	0.19	0.03	0.22
H Transformer WP	0.67	0.01	0.68	0.37	0.22	0.59
BERT	0.46	0.01	0.47	0.17	0.23	0.40
Enc_BART	0.51	0.00	0.51	0.21	0.20	0.41
Enc_T5	0.57	0.02	0.59	0.21	0.26	0.47
BART	0.70	0.03	0.73	0.35	0.28	0.63
T5	0.80	0.02	0.82	0.36	0.36	0.72
H BERT	0.63	0.01	0.64	0.20	0.34	0.54
H BART	0.72	0.01	0.73	0.29	0.28	0.57
H T5	0.68	0.01	0.69	0.31	0.27	0.58

For each original and swapped sessions, the preference of the model is highlighted in red (for differences above 0.01).

5.4 Human Evaluation

To further investigate the ability of the flat models, we conducted a human evaluation by comparing 100 queries predicted for AOL and MS Marco by all the models. The judges were presented complete sessions and corresponding suggestions predicted by each model. They had no knowledge of the ground truth or the user's goal. In our user modeling framework, we seek to evaluate whether suggestions make sense to annotators based on the user's session, not only whether they are syntactically correct. That's why judges were asked to evaluate the suggestions that were most likely to meet the user's need in the session by answering the question "is this query likely to follow in the session?". They were asked to rank the predictions from most to least suitable. Annotators are supposed to be able to infer the user's purpose from the session. Indeed, no more can be expected from an optimal policy that only has the user session at its disposal, and this is what we are trying to assess. Giving the user's purpose to the annotators could have biased the evaluation by leading the annotators to evaluate too negatively many suggestions, even though they corresponded to average user behavior. We further asked the annotator to rank exact repetitions and generic queries (e.g., "google") as bad predictions. We report in Table 5 the % of times a model is judged better than another one.

The evaluation confirms the results obtained with the other metrics. The models ACG, HRED and the different transformers are increasingly better (e.g., on AOL, 27% of predicted queries are better for Bart than for HRED, and 17% for the other way around). Among transformers, pre-trained models perform better (5%–10% gap), with Bart doing slightly better than Bert. Regarding WordPiece tokenization, they do perform better except for Transformer on AOL, and for ACG.

6 CONCLUSION

In this article, inspired by the success of transformer-based models [63] in various NLP and IR tasks, we looked at the various architectures that could be applied to query generation. We compared tokenizers, architectures, and different pre-training methods. We show that while hierarchical models permit to obtain better performance than corresponding flat architectures, they are not adapted for long and complex sessions. We conducted a deeper analysis on the flat models to understand why they are better at handling these sessions. We analyzed their generation process, and found

Table 5. Human Evaluation on 100 Queries for MS Marco and AOL

	HRED	HRED WPT	ACG	ACG WPT	TS	TS-WPT	BERT
				MS MACRO			
HRED WPT	19% vs 18%						
ACG	26% vs 29%	22% vs 22%					
ACG WPT	20% vs 22%	17% vs 21%	22% vs 24%				
TS	32% vs 11%	33% vs 13%	38% vs 16%	36% vs 13%			
TS WPT	37% vs 10%	35% vs 10%	42% vs 15%	39% vs 11%	15% vs 10%		
BERT	41% vs 10%	38% vs 8%	42% vs 15%	43% vs 11%	25% vs 15%	22% vs 18%	
BART	43% vs 9%	42% vs 11%	45% vs 11%	44% vs 9%	27% vs 15%	21% vs 16%	19% vs 16%
				AOL			
HRED WPT	23% vs 16%						
ACG	13% vs 24%	10% vs 29%					
ACG WPT	4% vs 24%	6% vs 32%	7% vs 15%				
TS	34% vs 17%	31% vs 20%	35% vs 3%	43% vs 5%			
TS WPT	28% vs 15%	24% vs 18%	32% vs 5%	38% vs 5%	13% vs 18%		
BERT	34% vs 13%	31% vs 18%	41% vs 9%	44% vs 6%	28% vs 24%	28% vs 19%	
BART	38% vs 17%	35% vs 20%	41% vs 11%	45% vs 8%	30% vs 28%	31% vs 24%	26% vs 24%

Each cell is the % of times model in row is better than model in column vs the reverse (and the remaining % is equality).

that the flat transformer is, on one hand, a position model that is able to recover the structure of a web search session (input queries are concatenated), and on the other hand, a good (query) language model. Future work will focus on improving the hierarchical architecture, so the model can handle more complex search tasks, and incorporating signals of various natures (longer history, clicked documents) into transformer-based architectures. Our study is limited to query-based search sessions, but the hierarchical structure of data is also present in conversational searches [4, 69]. However, while in our case the user is modeled according to their own past actions only, the setting of conversational search requires to consider external data such as available documents in the collection, or the IR system's answers, to drive the user toward their target documents. Our study could be extended in future work to the conversational search setting by integrating actions from the search agent in the model.

It will also focus on working on architectures able to cope with long sessions, potentially all the user history, using other recently introduced transformers [5, 18, 36] that overcome the limit of the maximum context length.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *Proceedings of the International Conference on Learning Representations*.

[2] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2019. Context attentive document ranking and query suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 385–394.

[3] Mohammad Aliannejadi, Manajit Chakraborty, Esteban Andrés Ríssola, and Fabio Crestani. 2020. Harnessing evolution of multi-turn conversations for effective answer retrieval. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*. ACM, New York, NY, 33–42.

[4] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, New York, NY, 475–484.

[5] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. arXiv:2004.05150. Retrieved from https://arxiv.org/abs/2004.05150.

[6] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, New York, NY, 795–804.

[7] 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, New York, NY, 56–63.

[8] 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 & Development in Information Retrieval*. ACM, New York, NY, 345–354.

[9] Daniele Broccolo, Lorenzo Marcon, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. 2012. Generating Suggestions for Queries in the Long Tail with an Inverted Index. *Information Processing and Management* 48, 2 (March 2012), 326–339.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems*. 1877–1901.

[11] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers. In *Proceedings of the International Conference on Learning Representations*.

[12] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aqware 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, New York, NY, 875–883.

[13] Ben Carterette, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson. 2016. Evaluating Retrieval over Sessions: The TREC Session Track 2011-2014. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 685–688.

[14] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2018. Attention-based hierarchical neural query suggestion. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 1093–1096.

[15] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bERT look at? an analysis of bERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. ACL, Florence, Italy, 276–286.

[16] Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Proceedings of the Advances in Neural Information Processing Systems*.

[17] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 985–988.

[18] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context". In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ACL, 2978–2988.

[19] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2020. TREC CAsT 2019: The conversational assistance track overview. (2020). arXiv:2003.13624. Retrieved from https://arxiv.org/abs/2003.13624.

[20] 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 26th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, 1747–1756.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North american Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, 4171–4186.

[22] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Proceedings of the Nips, Modern Machine Learning and Natural Language Processing Workshop*. Curran Associates Inc., Red Hook, NY.

[23] Vikas K. Garg, Inderjit S. Dhillon, and Hsiang-Fu Yu. 2019. Multiresolution transformer networks: Recurrence is not essential for modeling hierarchical structure. (2019). arXiv:1908.10408. Retrieved from https://arxiv.org/abs/1908.10408.

[24] Matthias Hagen, Jakob Gomoll, Anna Beyer, and Benno Stein. 2013. From search session detection to search mission detection. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*. Centre de hautes études internationales d'informatique documentaire, Paris, FRANCE, 85–92.

[25] 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 *Proceedings of the World Wide Web Conference*. ACM, New York, NY, 2792–2798.

[26] 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, New York, NY, 829–838.

[27] 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.* IEEE Computer Society, Washington, DC, 1443–1454.

[28] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.* ACL, 328–339.

[29] 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 (May 2003), 638–649.

[30] Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. 2011. Synthesizing High Utility Suggestions for Rare Web Search Queries. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, NY, 805–814.

[31] Alpa Jain, Umut Ozertem, and Emre Velipasaoglu. 2011. Synthesizing high utility suggestions for rare web search queries. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, NY, 805–814.

[32] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bERT learn about the structure of language?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* ACL, 3651–3657.

[33] 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, New York, NY, 197–206.

[34] 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 *Proceedings of the 5th International Conference on Spoken Language Processing.* ISCA.

[35] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations.*

[36] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations.*

[37] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. 2020. VideoFlow: A conditional flow-based model for stochastic video generation. In *Proceedings of the International Conference on Learning Representations.*

[38] 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 & Development in Information Retrieval.* ACM, New York, NY, 865–868.

[39] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* ACL, 7871–7880.

[40] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.* ACL, 1694–1704.

[41] Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. 2020. The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.* ACL, 118–126.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: a robustly optimized BERT pretraining approach. (2019). arXiv:1907.11692. Retrieved from https://arxiv.org/abs/1907.11692.

[43] 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.* ACM, New York, NY, 1101–1104.

[44] 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 & Development in Information Retrieval.* ACM, New York, NY, 285–294.

[45] 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, New York, NY, 469–478.

[46] 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.* ACM, New York, NY, 1755–1758.

[47] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-located With the 30th Annual Conference on Neural Information Processing Systems (CEUR Workshop Proceedings).*

[48] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. (2019). arXiv:1904.08375. Retrieved from https://arxiv.org/abs/1904.08375.

[49] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: A machine learning framework for ranking query suggestions. In *Proceedings of the 35st International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, NY, 25–34.

[50] Umut Ozertem, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu. 2012. Learning to suggest: A machine learning framework for ranking query suggestions. In *Proceedings of the 35th International ACM SIGIR Conference on Research & Development in Information Retrieval.* ACM, New York, NY, 25–34.

[51] 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 of the Association for Computational Linguistics.* ACL, Philadelphia, Pennsylvania, 311–318.

[52] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. (2019). arXiv:1904.07531. Retrieved from https://arxiv.org/abs/1904.07531.

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

[54] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[55] 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, New York, NY, 841–850.

[56] Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers Unite! Unsupervised Metrics for Reinforced Summarization Models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing.* ACM, Hong Kong, China, 3246–3256.

[57] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* ACM, Berlin, Germany, 1715–1725.

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

[59] Yang Song, Dengyong Zhou, and Li-wei He. 2011. Post-Ranking Query Suggestion by Diversifying Search Results. In *The 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, NY, 815–824.

[60] 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, New York, NY, 553–562.

[61] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

[62] Johanne R. Trippas, Damiano Spina, Paul Thomas, Mark Sanderson, Hideo Joho, and Lawrence Cavedon. 2020. Towards a model for spoken conversational search. *Information Processing and Management* 57, 2 (2020), 102–162.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Curran Associates Inc., Red Hook, NY, 6000–6010.

[64] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. (2020). arXiv:2006.04768. Retrieved from https://arxiv.org/abs/2006.04768.

[65] 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, Republic and Canton of Geneva, CHE, 1563–1571.

[66] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple applications of BERT for ad hoc document retrieval. (2019). arXiv:1903.10972. Retrieved from https://arxiv.org/abs/1903.10972.

[67] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[68] Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul N. Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. Few-Shot Generative Conversational Query Rewriting. In *The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 1933–1936.

[69] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating Clarifying Questions for Information Retrieval. In *Proceedings of the Web Conference 2020.* ACM, New York, NY, 418–428.

[70] 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 *Proceedings of the IEEE International Conference on Computer Vision.* 19–27.

Received May 2020; revised June 2021; accepted June 2021