Transformer Models for Recommending Related Questions in Web Search

Rajarshee Mitra, Manish Gupta, Sandipan Dandapat
Microsoft, Hyderabad, India
{rajarshee.mitra,gmanish,sadandap}@microsoft.com

ABSTRACT

People Also Ask (PAA) is an exciting feature in most of the leading search engines which recommends related questions for a given user query, thereby attempting to reduce the gap between user's information need. This helps users in diving deep into the topic of interest, and reduces task completion time. However, showing unrelated or irrelevant questions is highly detrimental to the user experience. While there has been significant work on query reformulation and related searches, there is hardly any published work on recommending related questions for a query. Question suggestion is challenging because the question needs to be interesting, structurally correct, not be a duplicate of other visible information, and must be reasonably related to the original query. In this paper, we present our system which is based on a Transformer-based neural representation, BERT (Bidirectional Encoder Representations from Transformers), for query, question and corresponding search result snippets. Our best model provides an accuracy of ~81%.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Machine learning**; **Natural language processing**; • **Information systems** → **Content ranking**.

KEYWORDS

People Also Ask, related question, context embedding, deep learning, BERT, transformers, snippets

ACM Reference Format:
Rajarshee Mitra, Manish Gupta, Sandipan Dandapat . 2020. Transformer Models for Recommending Related Questions in Web Search. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412067

1 INTRODUCTION

Beyond ten blue links, the search engine result page (SERP) on popular web search engines like Google and Bing also contains other related information. One such related information module is *People Also Ask* [19] which allows users to discover related questions pertaining to the current query. Fig. 1 shows PAA questions from

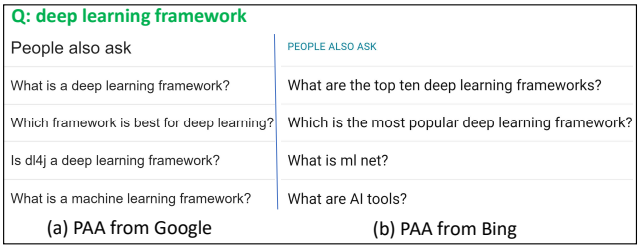

Fig. 1: An Example PAA Result from Two Most Popular Search Engines (retrieved on 18-Mar-2020)

Google and Bing for the query "deep learning framework". Such a module can be useful in (1) reducing gap between user's information need and the query, especially for non-expert users, (2) structured exploration of a research topic of user's interest, and (3) faster intent satisfaction. Building such a module involves interesting problems like: (1) finding relevant questions given a query, (2) linking such questions with Question-Answer (QA) pairs in a QA database, and (3) deciding the position of the PAA module on the SERP depending on the query. In this paper, we focus on the first problem.

We define a question to be relevant to a query based on the following criteria: (a) Question is a paraphrase of the query. Example: Query="*height of Eiffel tower*", Question="*How high is Eiffel tower?*" (b) Question is grounded on the *same entity* as in the query. These entities often include same person, same location, and other named entities. Example: Query = "*does Donald Trump have kids*", Question = "*Does Trump own Trump Towers?*" (c) Question is talking about a *related entity* (different from the query entity) within same domain like similar disease, similar hotel, etc. Example: Query = "*fever symptoms*", Question = "*How to know if you have malaria?*" (d) Question is a good follow-up to the original user query. Example: Query = "*netflix subscription*", Question = "*What are the best movies on Netflix?*" (e) Question is grounded on same domain like same movie, book or discipline (but different entity). Example: Query = "*10.12*", Question = "*What is the latest MacOS?*"

The objective of a good PAA question recommendation system is to block any irrelevant questions given a query. The task is challenging due to the widespread ambiguity in the query intent. E.g., consider the query "*oggy and the cockroaches*". While a question like "*Which channel oggy and the cockroaches airs on?*" is relevant, another question like "*Why do I have cockroaches in my home?*" is irrelevant. It is challenging even for a human to judge relevance in this example unless she knows that the query is primarily about a TV series. Clearly, simple word match based methods would fail for such examples. Although entity detection from queries seems obvious, open entity extraction from queries performs poorly especially for fresh entities.

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

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412067

Our system for recommending questions for PAA given a query depends on three important components: (1) a high-recall candidate generation module, (2) expanding the query (and question) using snippet text from SERP results, and (3) semantic similarity computation between query and question. The candidate generation module consists of a query log based related queries finder, followed by a query to question converter. Next, the proposed system expands both the query as well as the question by combining all snippets from their SERP results. We use high-ranked snippets since they contain relevant information about a query, and play a crucial role in perceived relevance [5, 12]. Finally, to estimate query-question relevance, we compute semantic similarity between query and a candidate question using recurrent or Transformer [27] based models like BERT (Bidirectional Encoder Representations from Transformers) [8].

In this paper, we make the following main contributions. (1) We provide a concrete definition of related questions given a query, for PAA module in web search. (2) We experiment with recurrent as well as Transformer based models for estimating query-question relevance. (3) We assess the importance of using snippets to improve query-question relevance estimation. (4) On a large dataset with 120K query-question pairs, we demonstrate the efficacy of our proposed system.

2 RELATED WORK

Related Search Suggestions: Abundant work has been done on suggesting related queries [1–3, 7, 16, 23, 26], ranking or filtering them [20, 21] as well as diversification of suggestions. Related query suggestions have been useful to show "related search" module on SERP, automated query refinement and query completion. Query suggestions rely on the "wisdom of the crowd", i.e., they are mined from query logs, specifically using the session logs and the query-clickthrough bipartite graph. Work on query suggestion relevance mostly uses two methods: (1) Two queries are related if they co-occur across many sessions [2, 10, 14, 21]. (2) Two queries are related if there is a large intersection between the set of documents clicked for the queries [20]. To create new unseen queries, Bonchi et al. [3] create a term-query graph which can produce related queries even for unseen queries as long as the terms are not unknown. Synthetic query suggestions have also been generated by generalization, narrowing or showing lateral suggestions[13]. PAA distinctively differs from related searches by recommending natural language questions rather than queries. One obvious method to generate PAA questions is to use related search to generate questions. We compare against this baseline in Section 4.

Query-Question Relevance: Query-question relevance estimation is important for community question answering (QA) forums in identifying existing QA pairs which match a new query. Previous work on answering a question given FAQ pages or a database of QA pairs (FAQ-Finder [9], Auto-FAQ [28], [15, 24, 25]) was based on traditional feature engineering for surfacing statistical/semantic similarities between query and questions. Recently, Transformer based models have been shown to be quite successful for the task, especially in chatbot settings where the query is being matched with a question-answer pair rather than just a question [6]. Although our problem setting also involves estimating query-question relevance, unlike QA systems, we do not have any answer in our context. Instead, we have snippets for SERP results as the context. Nevertheless, we compare with a context-less query-question relevance method in Section 4.

3 MODELS FOR RECOMMENDING RELATED QUESTIONS

Given a query q, we use the standard session log co-occurrence based related search method to get a set of related queries Q. Further, we use a simple n-grams model trained over a large question corpus to predict whether the query is a well-formed question or not. If it is not well-formed, we use the Keyword to Question (K2Q) system [29] to generate a natural language question from this query. We get these (query, question) pairs annotation (with 3-way redundancy) to generate a binary labeled query-question dataset.

3.1 SERPSim PAA

For every (query q, related question q') pair in the dataset, the SERPSim PAA module establishes similarity between q and q' based on similarity between SERP for q and q'. Specifically, it computes cosine similarity between titles and snippets of top 10 search results for query q and top 10 search results for question q'. Pairs with similarity greater than a threshold are predicted as relevant. The threshold is tuned on validation data.

3.2 Recurrent PAA

Fig. 2(a) illustrates the architecture of our Recurrent PAA method. Query q and question q' are encoded using GloVe embeddings [22] (we use 300D). Both query and question are processed using a shared encoder which is detailed in Fig. 2(b). We discuss details of the individual layers in the following.

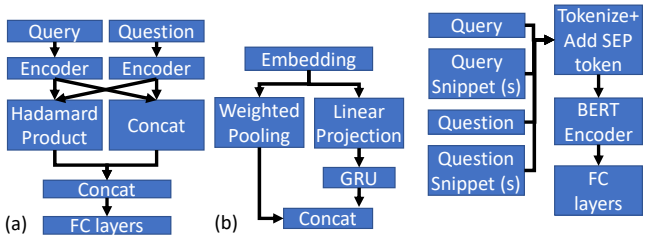

Fig. 2: (a): Recurrent PAA architecture (b) Recurrent PAA encoder

Fig. 3: BERT-based PAA+Snippets architecture

3.2.1 Encoder. The encoder contains two types of encoding – *weighted pooling* and *linear projection followed by Gated Recurrent Units (GRU)*. The final encoded features are concatenation of these two encodings.

Weighted Pooling learns the degree of attention that the encoder must give to individual words in q or q'. The attention weights are then used to compute a weighted sum of the inputs to compute the final representation. Given an input sequence of m word vectors (each with d dimensions) which form rows of matrix $Q \in \mathbb{R}^{m \times d}$,

weighted pooling performs the following operations:

$$w' = tanh(Q.W); \quad a = softmax(w'); \quad u = \sum_{t=1}^{m} a_t Q_t \qquad (1)$$

where Q_t is the input representation for the t^{th} word, $W \in \mathbb{R}^{\text{dx1}}$ are parameters to be learnt, $a \in \mathbb{R}^m$ are the attention weights, and $u \in \mathbb{R}^d$ is the pooled sequence.

Linear Projection followed by GRU: Concurrently, we also project the initial vector representations through a linear transformation and subsequently run a single layer GRU on it. The last state of the GRU is used as the representation.

$$p_t = V.Q_t; \quad s_t = GRU(s_{t-1}, p_t); \quad v = s_{m-1} \qquad (2)$$

where V are parameters to be learnt along with GRU parameters. We finally concatenate u and v from Eqs. 1 and 2 to create a shared fixed-length sentence representation $h = [u; v]$ for a (query, question) pair.

3.2.2 Interaction between Query and Question, and Fully Connected Layers. We compute two kinds of features to model the relationship between query and question: (1) Hadamard product (2) Concatenation of the query and question vectors. We finally concatenate these two vectors to create a single representation that forms the input to a multi-layer fully connected neural network. We use 2 dense layers followed by a final softmax output layer to predict relevance between the query and our candidate question. We use *tanh* nonlinearity.

3.3 BERT-based PAA

BERT is a 12-layer transformer-based [27] language model that has proved to be very beneficial for various downstream Natural Language Processing (NLP) tasks [8, 17, 18]. We used the pre-trained model which has been trained on Books Corpus and Wikipedia using the MLM (masked language model) and the next sentence prediction (NSP) loss functions. The query and question are concatenated into a sequence and are separated with a special "SEP" token. The sequence is prepended with a "CLS" token. The representation C for the "CLS" token from the last encoder layer is fed to a single dense layer followed by tanh activation. Output of this layer is used for classification by connecting it to an output softmax layer. We finetune the entire model using labeled training data for the query-question relevance prediction task.

3.4 BERT-based PAA+Snippets

We observe that rich context could be useful in improved semantic understanding of short queries and questions. From SERP, for each query (and question), we extract top N snippets and use the concatenated text as context. BERT-based PAA+Snippets is similar in architecture to BERT-based PAA except that input sequence consists of query, question, query-snippets and question-snippets, separated by a "SEP", and prepended with a "CLS" token (Fig. 3).

4 EXPERIMENTS

4.1 Dataset

We create our dataset using anonymous search queries from a popular search engine. For each search query, we form an initial set of candidate questions using session logs and K2Q as discussed in Section 3. We randomly sample queries and randomly draw 6 questions for each query to create a dataset containing (query, question) pairs. These pairs are then annotated by human judges, as relevant or not, with 3-way redundancy with an inter-annotator agreement of 0.7. We split the data into disjoint train, dev and test sets containing 110000, 1000 and 9000 pairs, respectively. The dataset is class-balanced. Table 1 shows the percentage of various question types on our dataset. Note that question types are not mutually exclusive, e.g., a follow-up question could still have the same entity.

Table 1: Percentage of various question types (not mutually exclusive) in our dataset.

Question type	Data percent
Paraphrase	30
Question with matching entity	60
Question with related entity	20
Follow-up question	40
Same domain question	30

For every query, we extract top N snippets from the SERP. For the candidate questions also, we query them on the same search engine and extract corresponding top N snippets. We keep the maximum length of each snippet to be 80 words.

4.2 Experimental Settings and Baselines

We maintain a vocabulary size of 54000 words and use the Stanford CoreNLP package[1] for tokenization. We set a hidden dimension of 150 for all the recurrent units. For recurrent PAA, the fully connected linear layers have dimensions set as 150, 100 and 2. We use *tanh* activation function in all the hidden layers. We use a batch size of 50 for Recurrent PAA. For BERT-based PAA, we use a batch size of 30. While training the BERT-based PAA model, we fine-tune the BERT encoder along with our classification task. For all the tasks, we use the cross-entropy error and ADAM optimizer. For our BERT-based PAA+Snippets model, we vary #snippets as 1, 2 and 3.

We compare our proposed models against two baseline methods: Deep Semantic Similarity Model (DSSM) [11] and Universal Sentence Encoder (USE) [4]. For these baselines, we use the pretrained sentence embedding from DSSM and USE models respectively, and then stack a fully connected layer on top of them. We train only last layer weights using our dataset to minimize cross-entropy loss.

4.3 Results

We report overall accuracy, precision and recall of the 'relevant' class for various methods on test set in Table 2. Our proposed model "BERT-based PAA+2 snippets" performs the best. While adding snippets to the BERT-based PAA models helps, adding two snippets

[1] https://stanfordnlp.github.io/CoreNLP/download.html

Table 2: Accuracy comparison across various methods.

Model	Accuracy	Precision	Recall
DSSM + Classification	0.69	0.70	0.69
USE + Classification	0.71	0.70	0.70
SERPSim PAA	0.65	0.76	0.58
Recurrent PAA	0.75	0.75	0.75
BERT-based PAA	0.77	0.78	0.77
BERT-based PAA + 1 snippet	0.79	0.76	0.81
BERT-based PAA + 2 snippets	**0.81**	**0.79**	**0.82**
BERT-based PAA + 3 snippets	0.79	0.75	0.81

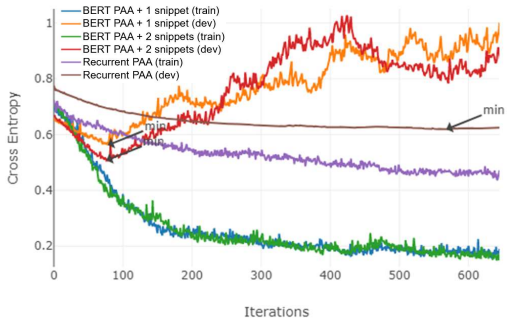

Fig. 4: Cross entropy loss vs #iterations.

provides better results compared to adding three. In terms of accuracy, our best model provides an improvement of ∼16% compared to basic SERPSim PAA model, and ∼10% and ∼12% compared to the USE and DSSM baseline models resp. Fig. 4 shows the training progress of all the models in a cross entropy loss vs #iterations plot. We observe that (1) Among BERT-based PAA models, the one with 2 snippets has much better minima compared to the 1 snippet model, for the dev set. (2) Minima for models with BERT-based models is much lower compared to Recurrent PAA model both for the train and dev set. (3) BERT-based PAA models converge much faster compared to Recurrent PAA model.

4.4 Qualitative Results

Table 3 shows a few predictions from our best BERT-based PAA with snippets model. It is able to correctly predict irrelevant examples by "looking" at words along with the right context. It can disambiguate between 'PC virus' versus 'virus as a pathogen', and between the 'time after time' song and the usual meaning of time. Also, the model discovers obviously related words like wife and children; and machine learning and AI.

Table 3: Sample Prediction Results

Query	Question	Prediction
blue is the warmest colour reviews	Why is sky blue in colour?	Irrelevant
how do I remove virus from PC	What virus causes high fever?	Irrelevant
time after time song	what time is it now?	Irrelevant
donald trump wife	how many children does melania trump have	Relevant
machine learning	what are the branches in AI?	Relevant
photosynthesis	how do plants make their food?	Relevant

We observed that most of the errors made by our best model were because of bad snippet or snippet digressing from the query. E.g., for q='flu', and query snippet='Symptoms. Initially, the flu may seem like a common cold with a runny nose, sneezing and sore throat. But colds usually develop slowly, whereas ...', the question='how to prevent common cold?', and question snippet as 'The common cold, also known simply as a cold, is a viral infectious disease of the upper respiratory tract that primarily affects the nose. The throat, sinuses, and ...', our model predicted relevant while the judges marked irrelevant with the comment that 'question should stick to flu'. The problem is that focus is more on common cold and throat in the snippet, such that the attention on query subsides.

5 CONCLUSION

In this work, we discussed the problem of recommending questions given a query in web search. Specifically, we looked at questions that could be useful for the *People Also Ask* module. We experimented with multiple techniques including the basic SERP similarity based method, recurrent neural network based model and Tranformer based BERT model. We obtained best accuracy with BERT based model especially when we also encode contextual snippets as part of the input. Compared to typical sentence embedding based methods, our system performs much better in terms of precision, recall as well as accuracy.

REFERENCES

[1] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. 2004. Query Recommendation Using Query Logs in Search Engines. In *EDBT*. 588–596.
[2] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. 2008. The query-flow graph: model and applications. In *CIKM*. 609–618.
[3] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. 2012. Efficient query recommendations in the long tail via center-piece subgraphs. In *SIGIR*. 345–354.
[4] D. Cer, Y. Yang, S.-Y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and Ray. Kurzweil. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175 (2018).
[5] A. Chuklin and M. de Rijke. 2015. The Anatomy of Relevance: Topical, Snippet and Perceived Relevance in Search Result Evaluation. *CoRR* abs/1501.06412 (2015).
[6] S. Damani, K. N. Narahari, A. Chatterjee, M. Gupta, and P. Agrawal. 2020. Optimized Transformer Models for FAQ Answering. In *PAKDD*. 235–248.
[7] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury. 2017. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *CIKM*. 1747–1756.
[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018).
[9] K. Hammond, R. Burke, C. Martin, and S. Lytinen. 1995. FAQ finder: a case-based approach to knowledge navigation. In *Conf. on AI for applications*, Vol. 114.
[10] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *JASIST* 54, 7 (2003), 638–649.
[11] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. 2333–2338.
[12] M. A. Islam, R. Srikant, and S. Basu. 2018. Micro-Browsing Models for Search Snippets. *CoRR* abs/1810.08223 (2018).
[13] A. Jain, U. Ozertem, and E. Velipasaoglu. 2011. Synthesizing high utility suggestions for rare web search queries. In *SIGIR*. 805–814.
[14] E. C. Jensen, S. M. Beitzel, A. Chowdhury, and O. Frieder. 2006. Query Phrase Suggestion from Topically Tagged Session Logs. In *FQAs*. 185–196.
[15] V. Jijkoun and M. de Rijke. 2005. Retrieving answers from frequently asked questions pages on the web. In *CIKM*. 76–83.
[16] R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *WWW*. 387–396.
[17] X. Liu, K. Duh, and J. Gao. 2018. Stochastic Answer Networks for Natural Language Inference. *CoRR* abs/1804.07888 (2018).
[18] X. Liu, P. He, W. Chen, and J. Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. *CoRR* abs/1901.11504 (2019).
[19] Y. Matias, D. Keysar, G. Chechik, Z. Bar-Yossef, and T. Shmiel. 2017. Generating related questions for search queries.
[20] Q. Mei, D. Zhou, and K. W. Church. 2008. Query suggestion using hitting time. In *CIKM*. 469–478.
[21] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. 2012. Learning to suggest: a ML framework for ranking query suggestions. In *SIGIR*. 25–34.
[22] J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.
[23] E. Sadikov, J. Madhavan, L. Wang, and A. Y. Halevy. 2010. Clustering query refinements by user intent. In *WWW*. 841–850.
[24] E. Sneiders. 1999. Automated FAQ answering: Continued experience with shallow language understanding. In *Question Answering Systems, AAAI*. 97–107.
[25] W. Song, M. Feng, N. Gu, and L. Wenyin. 2007. Question similarity calculation for FAQ answering. In *Semantics, Knowledge and Grid*. IEEE, 298–301.
[26] H. Vahabi, M. Ackerman, D. Loker, R. A. Baeza-Yates, and A. López-Ortiz. 2013. Orthogonal query recommendation. In *RecSys*. 33–40.
[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is All you Need. In *NIPS*. 6000–6010.
[28] S. D. Whitehead. 1995. Auto-FAQ: An experiment in cyberspace leveraging. *Computer Networks and ISDN Systems* 28, 1-2 (1995), 137–146.
[29] Z. Zheng, X. Si, E. Y. Chang, and X. Zhu. 2011. K2Q: Generating Natural Language Questions from Keywords with User Refinements. In *IJCNLP*. 947–955.