ILPS at TREC 2019 Conversational Assistant Track

Nikos Voskarides
University of Amsterdam
Amsterdam, The Netherlands
n.voskarides@uva.nl

Dan Li
University of Amsterdam
Amsterdam, The Netherlands
d.li@uva.nl

Andreas Panteli
University of Amsterdam
Amsterdam, The Netherlands
andreas.panteli@student.uva.nl

Pengjie Ren
University of Amsterdam
Amsterdam, The Netherlands
p.ren@uva.nl

ABSTRACT

This paper describes the participation of the *UvA.ILPS* group at the TREC CAsT 2019 track. We propose a cascade architecture that consists of (i) a unsupervised initial retrieval step that uses on a query expansion model that extracts words from the previous turns that are relevant to the current turn, and (ii) a supervised neural ranker that is based on BERT. We use transfer learning to pretrain our neural ranker with a single-turn passage ranking dataset (MS MARCO) and a multi-turn passage ranking dataset that we induced from a dataset originally proposed for a different task (QuAC). Official results show that our best run outperforms the median run by 25.6% in terms of NDCG@5 and 26.4% in terms of NDCG@1000.

1 INTRODUCTION

Conversational AI has recently received a lot of attention in both the IR and NLP communities [7]. The TREC Conversational Assistant track is one of the first steps towards building conversational search systems. This year's task is to perform passage ranking over a large collection in a conversational multi-turn setting, where the sequence of queries is predefined.

In our participation we focus on designing a ranking system particularly for this task, and on using transfer learning to transfer knowledge from different datasets to our task. Our ranking system follows a cascade architecture that consists of the initial retrieval step (Section 2.1) and the re-ranking step (Section 2.2). The former step relies on an unsupervised ranker and a query expansion model, while the latter relies on a neural supervised ranker. Since our training dataset is relatively small, we employ transfer learning to pretrain the supervised ranker (Section 2.3).

2 METHODOLOGY

We follow a cascade architecture, which is standard in document and passage retrieval especially for large collections [15]. Given a question at the i-th turn, q_i, together with the previous questions $[q_1, ...q_{i-1}]$, in the initial retrieval step, we produce a ranked list of passages and keep the top-2000. Then, in the reranking step, we rerank the set of passages obtained by the first step and keep the top-1000.

2.1 Unsupervised initial retrieval

The goal of the initial retrieval step in our cascade architecture is to achieve high recall up to a reasonable depth before using a more sophisticated reranking module. This is especially challenging in our setting, where queries are sequential and the "interpretation" of the current turn query depends on queries of the previous turns. This can be due to phenomena such as the usage of pronouns (see turn #2 in Table 1) and the omission of contextual information (see turn #3 in Table 1). Furthermore, low recall in the initial retrieval step is an important bottleneck for applying neural IR models in practice [8]. Therefore, before diving into neural IR models, we focus on building a strong unsupervised model for initial retrieval for this task.

Table 1: Sequence of queries for the topic: "Career choice for Nursing and Physician's Assistant", TREC 2019 CAsT training data.

Turn	Query
1	What is a physician's assistant?
2	What are the educational requirements required to become one?
3	What does it cost?
...	...
11	What is the fastest way to become a NP?
12	How much longer does it take to become a doctor after being an NP?

We use standard query likelihood with dirichlet smoothing and RM3 relevance feedback as the ranking model [1, 17, 18]. In order to build a more self-contained representation of the current turn query, we propose a query expansion model to extract words that capture relevant information from the previous turns and add them to the query of the current turn. Our query expansion model builds on the following two assumptions:

(1) *Word centrality*: words that are central in the queries up the current turn capture the main theme of the conversation.
(2) *Word recency*: words that appear in the most recent turns of the conversation are more relevant to the current turn.

In order to model word centrality (assumption 1), we first construct an undirected weighted word graph, where the nodes are the unique words of the queries up to current turn. We add edges for each pair of words in the graph and weigh them using the cosine similarity of their respective word2vec vectors [12]. We drop edges that have a weight < 0.1. The centrality score $c(w)$ of a word w is the sum of the weights of the edges that connect to that word.

In order to model word recency (assumption 2), we calculate the recency score $r_i(w)$ of a word w at turn i as follows:

$$r_i(w) = \sum_{j \in T(w)} \exp^{-\lambda(i-j)}, \quad (1)$$

where $T(w)$ are the turns that word w appears and λ is a decay factor.

The final score $s(w)$ of a word w at turn i is calculated as a linear combination of the above scores:

$$s(w) = \alpha \cdot c(w) + (1 - \alpha) \cdot r_i(w), \quad (2)$$

where α is a parameter that controls the relative importance of word centrality and recency. We add the top-k scoring words calculated using Equation (2) to the original query q_i.

2.2 Supervised neural reranking

In this step we build a supervised neural ranker to re-rank the set of passages obtained using the previous step.

In contrast to the initial ranker which only uses the original q_i and the expanded words to form the query, our neural ranker uses BERT to encode the all queries up to the current turn $[q_1, ...q_{i-1}, q_i]$ alongside with the passage p. We add a dropout layer and a linear layer l_a on top of the `CLS` node in the last layer of the BERT model to produce a matching score [11].

We combine the BERT matching score and the score obtained by the initial retrieval step (for which we perform min-max normalization per query [17]) to produce the final matching score using an output linear layer l_b with a tanh activation function.

We train the neural ranker using pairwise ranking loss [16]. During training we sample as many negatives as positives per query.

2.3 Transfer learning

The biggest challenge in using the neural ranker described above is that the training dataset provided is very small. The TREC 2019 CAsT training dataset consists of only 30 topics (269 queries in total), and not all topics have relevant passages for all turns. To address this challenge, we use *transfer learning* [10, 14]. We pretrain our models in the following ways:

2.3.1 Language model pretraining. We initialize the BERT parameters using a pretrained model that is trained on a large open domain corpus with a language modeling objective [3].

2.3.2 Single-turn passage ranking pretraining. We use a subset of the MS MARCO passage ranking dataset [13] to pretrain the BERT parameters and the l_a layer on single-turn questions.

2.3.3 Multi-turn passage ranking pretraining. Since there is no available large-scale dataset for multi-turn passage ranking, we adjust an existing dataset for this task. We use the QuAC dataset that was originally proposed for interactive question answering, where the relevant answers exist in a single Wikipedia section. We detail how we adjust this dataset for multi-turn passage ranking in Section 3.1.3. Using this dataset, we pretrain the BERT parameters, and the l_a and l_b layers.

Finally, we fine-tune the whole model on a subset of the training topics of the TREC CAsT training dataset.

Table 2: Excerpt from an example dialog taken from the QuAC training data. The paragraph originates from the section "History" of the Wikipedia article on "Saosin". We denote the answer spans of each query turn in the paragraph with a superscript.

Turn	Query
1	Who formed Saosin?
2	When was the band founded?
3	What was their first album?
...	...

Wikipedia paragraph: The [original lineup for Saosin, consisting of Burchell, Shekoski, Kennedy and Green]$^{\text{turn1}}$, was [formed in the summer of 2003]$^{\text{turn2}}$. On June 17, the band released their [first commercial production, the EP Translating the Name]$^{\text{turn3}}$.

3 EXPERIMENTAL SETUP

3.1 Datasets

3.1.1 TREC CAsT. The TREC 2019 CAsT dataset consists of 30 training and 50 evaluation topics. Each topic consists of a sequence of queries. Out of 30 training topics, only 13 had at least one query with a relevant document. Out of the 50 evaluation topics, only 20 were assessed by expert annotations to an average depth of eight turns. The passage collection includes three passage corpora: MS MARCO [13] (Bing), TREC CAR [4] (Wikipedia passages) and Washington Post (WP) [9] (passages from news articles). Note that in order to fine-tune our models we kept 5 of the training topics aside for development and used the remaining 8 for training.

3.1.2 MS MARCO. For single-turn passage ranking (see Section 2.3.2), we used the MS MARCO passage ranking dataset, because it uses the MS MARCO passage collection, which is a subset of the passage collection of TREC CAsT. We sampled 100K triplets from the training set and 200 queries from the development set due to time restrictions.

3.1.3 QuAC. The QuAC dataset [2] was originally constructed by asking two crowd workers (a student and a teacher) to perform an interactive dialog about a specific topic (e.g. history of the Saosin music band). The student asks questions about the topic, whereas the teacher answers by providing spans from a Wikipedia text about that topic. A single QuAC conversation, contains up to 12 queries about a single Wikipedia paragraph with answer spans from the same paragraph associated with each query. Table 2 shows an example dialog from the QuAC dataset.

We induce an artificial multi-turn passage ranking dataset from QuAC as follows. Recall that TREC CAR is one of the passage collections in TREC CAsT. Since QuAC and TREC CAR both use Wikipedia as the underlying collection, for each query, we automatically map the passage that contains the answer span to a passage in TREC CAR and thereby generate a positive query-passage pair for each query. We obtain the mapping from QuAC passages to TREC CAR passages by first creating a query that consists of the answer span and 65 characters left and right of the span to avoid mismatching. Second, we query the TREC CAR collection, and if

Table 3: Experimental results on the official evaluation topics.

Run	NDCG@5	NDCG@1000	MAP@5	MAP@1000	Recall@1000	MRR
Median	0.2960	0.3840	0.0420	0.1740	-	-
`ilps-lm-rm3-dt`	0.2671	0.4253	0.0406	0.2307	0.6503	0.5309
`ilps-bert-feat1`	**0.3719**	**0.4857**	0.0532	0.2614	**0.6808**	0.6176
`ilps-bert-feat2`	0.3534	0.4739	0.0537	0.2576	0.6741	0.6060
`ilps-bert-featq`	0.3685	0.4801	**0.0587**	**0.2636**	0.6735	**0.6569**

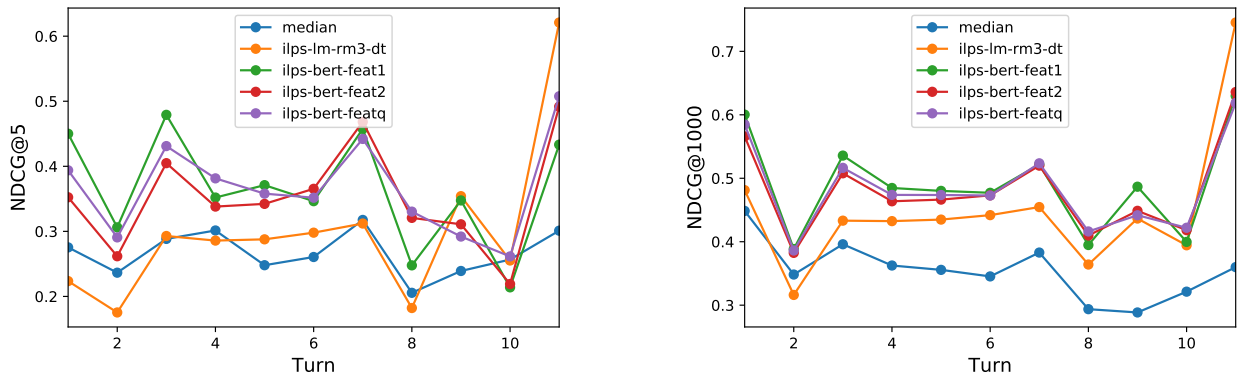

Figure 1: Ranking performance in terms of NDCG@5 (left) and NDCG@1000 (right) per turn averaged over topics for our runs and the median.

the top-ranked passage contains the constructed query, we keep that passage as the positive passage for the query.

Note that, in a QuAC dialogue, the first turn query is often not self-contained because it may depend on the Wikipedia article's title and section title. To address this, we substitute pronouns (he, she, her, him, it, they, them) and determiners (his, hers, its, theirs) in the first turn query with the Wikipedia page title. Also, we exclude dialog queries with a *CANNOTANSWER* label. In sum, we created additional 30,510 queries out of a total of 83,568 from the QuAC dataset; 21,168 queries were used for training and 9,342 queries for validation.

It is important to note that pretraining with this dataset can be limiting since its relevant passages originate from the TREC CAR collection only. Another limitation that might cause discrepancy is that a question in the QuAC dialogue may depend on not only the previous questions as in CAsT but also on the previous answers in the dialog, which is not the case for TREC CAsT. we plan to address these limitations in future work.

3.2 Runs

We submitted the following automatic runs:

* `ilps-lm-rm3-dt`: Uses the initial retrieval model only (Section 2.1).
* `ilps-bert-feat1`: Uses the full cascade architecture (Sections 2.1 and 2.2). We apply language modeling pretraining (Section 2.3.1), followed by and single-turn passage ranking pretraining (Section 2.3.2). Fine-tuning is done using the TREC CAsT dataset.
* `ilps-bert-feat2`: Same as `ilps-bert-feat1` with different hyperparameters.
* `ilps-bert-featq`: Same as `ilps-bert-feat1` with the difference that multi-turn passage ranking pretraining (Section 2.3.3) is applied after single-turn passage ranking pretraining and before fine-tuning.

For pretraining on both single-turn and multi-turn passage ranking, and for fine-tuning on CAsT, we use early stopping on the corresponding validation set based on the MRR score.

3.3 Implementation

We index the collections and perform initial retrieval using the Python Anserini implementation [5]. We build on the implementation of BERT for ranking (VanillaBERT) in PyTorch provided by MacAvaney et al. [11]. For text tokenization we use the BERT tokenizer from the python library *pytorch-pretrained-bert 0.6.2* [6].

3.4 Parameter configuration

Initial ranker. For QL with Dirichlet smoothing we set $\mu = 2500$. For RM3, we use 10 feedback documents and 10 terms, and set the original query weight to 0.8. For the query expansion model, we set $k = 10$, $\alpha = 0.2$ and $\lambda = 0.1$ based on preliminary experiments.

Supervised neural ranker. The BERT weights were initialized by importing the weights from the *bert-base-uncased* model released by Hugging Face [6]. We use a learning rate of 3e−6 and dropout

probability of 0.2 on the l_a and l_b layers. We keep the BERT weights fixed for the first 3 epochs and use a batch size of 2.

4 RESULTS

Overall performance. Table 3 lists the results of our four runs on the official evaluation set. First, we observe that the three runs that use the full cascade architecture (`ilps-bert-feat*`) outperform the median run by a large margin on all metrics. `ilps-bert-feat1` performs best in terms of NDCG and Recall while `ilps-bert-featq` performs best in terms of MAP and MRR. `ilps-bert-featq`'s performance indicates that pretraining using an artificially induced multi-turn passage retrieval dataset is beneficial. Its MRR score (0.6569) indicates that, on average, it ranks a relevant passage at the first or the second position of the ranking. Furthermore, we observe that `ilps-lm-rm3-dt`, which only uses our recall-oriented initial retrieval step, outperforms the median run at lower cutoffs. Also, all of the four runs achieve relatively high recall@1000 (about two thirds of the relevant passages are retrieved).

Performance per turn. Figure 1 shows the performance of our runs and the median per turn averaged over topics in terms of NDCG@5 and NDCG@1000. For both metrics, We observe that the performance of our `ilps-bert-feat*` runs are relatively robust across different turns. As expected, we observe a gradual decrease in performance towards later turns (except in turns 2 and 7, for which further investigation is needed). For NDCG@5 (left), we observe that the three `ilps-bert-feat*` runs outperform `ilps-lm-rm3-dt` and the median run up to turn 8. Note that for turns after turn 8 we only average over a small subset of the topics, hence the average might be less representative. We do not observe large differences in performance among the `ilps-bert-feat*` runs. For NDCG@1000 (right), we observe similar patterns with NDCG@5. Also, the recall-oriented `ilps-lm-rm3-dt` run outperforms the median run by a large margin in all turns except the second.

5 CONCLUSION

We presented our participation in the TREC 2019 CAsT track. Official results show that our best runs achieve competitive performance and outperform the median run by a large margin.

REFERENCES

[1] Nasreen Abdul-jaleel, James Allan, W Bruce Croft, O Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *In Proceedings of TREC-13*. Citeseer.

[2] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC : Question Answering in Context. *CoRR* abs/1808.07036 (2018). arXiv:1808.07036 http://arxiv.org/abs/1808.07036

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC Complex Answer Retrieval Overview. In *TREC*.

[5] Emily Wang (emmileaf). 2019. *Pyserini: Anserini Integration with Python.* https://github.com/castorini/anserini/blob/master/docs/pyserini.md

[6] Hugging Face. 2019. *PyTorch-Transformers.* https://github.com/huggingface/pytorch-pretrained-BERT

[7] Jianfeng Gao, Michel Galley, Lihong Li, et al. 2019. Neural approaches to conversational AI. *Foundations and Trends® in Information Retrieval* 13, 2-3 (2019), 127–298.

[8] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2019. A Deep Look into Neural Ranking Models for Information Retrieval. *Information Processing & Management* (2019), 102067. https://doi.org/10.1016/j.ipm.2019.102067

[9] Jamie Callan Jeff Dalton, Chenyan Xiong. 2019. *The TREC Conversational Assistance Track (CAsT).* http://www.treccast.ai/

[10] Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story Ending Prediction by Transferable BERT. *CoRR* abs/1905.07504 (2019). arXiv:1905.07504 http://arxiv.org/abs/1905.07504

[11] 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 (SIGIR'19)*. ACM, New York, NY, USA, 1101–1104. https://doi.org/10.1145/3331184.3331317

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959

[13] 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. *CoRR* abs/1611.09268 (2016). arXiv:1611.09268 http://arxiv.org/abs/1611.09268

[14] Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks. *CoRR* abs/1811.01088 (2018). arXiv:1811.01088 http://arxiv.org/abs/1811.01088

[15] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, New York, NY, USA, 105–114. https://doi.org/10.1145/2009916.2009934

[16] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. 2008. COFI RANK - Maximum Margin Matrix Factorization for Collaborative Ranking. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 1593–1600. http://papers.nips.cc/paper/3359-cofi-rank-maximum-margin-matrix-factorization-for-collaborative-ranking.pdf

[17] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple Applications of BERT for Ad Hoc Document Retrieval. *CoRR* abs/1903.10972 (2019). arXiv:1903.10972 http://arxiv.org/abs/1903.10972

[18] Chengxiang Zhai and John Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. ACM, New York, NY, USA, 334–342. https://doi.org/10.1145/383952.384019