Slice-Aware Neural Ranking

Gustavo Penha
TU Delft
g.penha-1@tudelft.nl

Claudia Hauff
TU Delft
c.hauff@tudelft.nl

Abstract

Understanding when and why neural ranking models fail for an IR task via error analysis is an important part of the research cycle. Here we focus on the challenges of (i) identifying categories of *difficult* instances (a pair of question and response candidates) for which a neural ranker is ineffective and (ii) improving neural ranking for such instances. To address both challenges we resort to *slice-based learning* (Chen et al., 2019) for which the goal is to improve effectiveness of neural models for slices (subsets) of data. We address challenge (i) by proposing different *slicing functions* (SFs) that select slices of the dataset—based on prior work we heuristically capture different failures of neural rankers. Then, for challenge (ii) we adapt a neural ranking model to learn slice-aware representations, i.e. the adapted model learns to represent the question and responses differently based on the model's prediction of which slices they belong to. Our experimental results[1] across three different ranking tasks and four corpora show that slice-based learning improves the effectiveness by an average of 2% over a neural ranker that is not slice-aware.

1 Introduction

Retrieving text for a given information need is a fundamental task in Information Retrieval (IR). For a long time neural networks failed to convincingly outperform traditional term matching approaches with pseudo-relevance feedback, e.g. RM3 (Abdul-Jaleel et al., 2004), for text retrieval tasks including the classic adhoc retrieval task (Yang et al., 2019a). However, with recent breakthroughs in natural language processing (NLP), neural approaches—prominently BERT (Devlin et al., 2019)—are achieving state-of-the-art effectiveness across a range of text retrieval tasks (Yang et al., 2019b; Nogueira and Cho, 2019).

SF_0
```
def sf_long_question(x, t=5):
    return len(x.question.split(" ")) > t
```

SF_1
```
def sf_BERT_difficulty(x, t=0.1):
    p_rel = np.mean([BERT.pred(x.question, res) \
                     for res in x.rel_resp])
    p_not_rel = np.mean([BERT.pred(x.question, res)\
                     for res in x.not_rel_resp])
    return (p_rel - p_not_rel) < t
```

Figure 1: Examples of slicing functions (SFs) to capture subsets of difficult tuples of question and response list. The SFs also have access to relevance labels for the training set, as they are not required at test time by the slice-aware neural ranker. SF_0 uses the question length as a proxy for question complexity, and SF_1 calculates how distinguishable relevant and non-relevant responses are based on BERT predictions.

Understanding when and why retrieval models fail is an important part of the research cycle. Even tough we have clues about the failures of neural rankers—obtained for instance by the study of question performance prediction (He and Ounis, 2006), diagnostic datasets (Câmara and Hauff, 2020) and error analysis (Wu et al., 2019)—automatically identifying difficult instances (tuples of question and response list) and improving the effectiveness of models for such difficult instances are still open challenges. We consider here difficult instances to be question and responses for which a given neural ranker retrieval effectiveness is below the average. A recent approach, referred to as *slice-based learning* (Chen et al., 2019), has been proposed to identify and improve the effectiveness of subsets of data (so-called *slices*), as opposed to focusing on all data equally. The core idea is that a slice-aware neural model will represent instances differently depending on the slices of data they come from. Slice-based learning has been applied to computer vision and NLP tasks, with overall effectiveness improvements up to 3.5% (Chen et al., 2019) over a model that is not slice-aware.

In this paper we focus on the challenges of (i)

[1]The source code and data are available at https://github.com/Guzpenha/slice_based_learning.

Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI), pages 1–6

ISBN 978-1-952148-67-5

detecting difficult instances for neural rankers and (ii) improving the retrieval effectiveness for such instances. We address the challenges by (i) creating slicing functions (SFs), i.e., functions that define whether an instance belongs to a slice which heuristically capture different errors of rankers (cf. Figure 2 for examples of SFs); and (ii) employing a slice-aware neural ranker, i.e., a neural ranker that learns to represent each instance differently based on its prediction of which slice the input belongs to (cf. Figure 2 for a diagram of the slice-aware neural ranker). Our main research questions are the following two. **RQ1:** To what extent can slice-based learning improve neural ranking models? **RQ2**: What are the underlying reasons for the effectiveness of slice-based learning?

Our experimental results on three different conversational tasks show that slice-based learning is beneficial to IR, showing positive evidence for RQ1. The gains are observed for both overall effectiveness and the effectiveness for slices of the data. Concerning RQ2, we evaluate to which extent the effectiveness gains observed for the slice-aware model come from the effect of ensemble learning (Dietterich et al., 2002), a direction not explored empirically by previous work (Chen et al., 2019). We find that, when using *random* SFs we can also significantly improve upon a non slice-aware neural ranker. We note though that not all improvements of slice-based learning can be attributed to the effect of ensemble learning, and carefully implementing SFs is indeed advantageous.

2 Slice-based Learning

Slice-based learning (Chen et al., 2019) is an approach based on the engineering of SFs that capture slices of data. The SFs all follow the same format: they receive the instance as input (in our case a question and a list of candidate responses) and return a boolean variable indicating whether the instance belongs to the slice. Based on the SFs a neural model is adapted to improve the effectiveness of such slices of data, for example, by having a different set of weights for each slice. Training a different model for each slice, and combining their predictions is inefficient: training and maintaining a different neural ranking model for each slicing function amounts to a large number of parameters and an increased prediction time. As an efficient solution, Chen et al. (2019) proposed Slice-Residual-Attention Modules (SRAMs), which is a slice-aware approach for neural models that shares parameters in a similar manner to multi-task learning (Caruana, 1997).

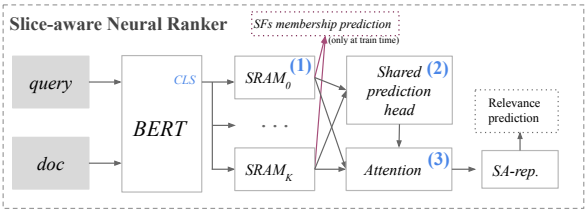

Figure 2: Overview of the slice-aware neural ranker. For each SF we define we have a SRAM module to learn slice-expert representations, that are then combined with an attention mechanism into a slice-aware representation.

3 Slice-based learning for IR

We first introduce the SFs we defined to heuristically capture subsets of data containing different categories of errors, for which the effectiveness is lower than average, based on intuitions drawn from prior work (RQ1). We then introduce the random SFs we deploy to study the effect of ensemble learning in slice-based learning (RQ2). Finally we describe the slice-aware neural ranker.

3.1 Slicing Functions

We divide our SFs into two categories: those based only on the question text (question based) and those that uses both the question and the list of candidate responses (question-responses based). The relevance labels for the training instances are also inputs to the SFs, which are not required at inference time as the slice-aware neural ranker learns to predict slice-membership.

3.1.1 Question-based SFs

Question Length (QL): the number of question terms is higher than the threshold T_{QL}. QL was shown to correlate negatively with the effectiveness of retrieval methods in adhoc retrieval (Bendersky and Croft, 2009). Long questions (questions with high QL) provide a way of expressing complex information needs as opposed to short questions (Phan et al., 2007). Context Length (CL) [2]: the number of turns in the dialogue context is higher than the threshold T_{CL}. CL was shown to correlate negatively with model's effectiveness for the conversation response ranking task when using different neural rankers (Tao et al., 2019).

[2]This SF is only suited for QA tasks with multiple turns.

Question Category (QC): question is about a certain semantic category, e.g. $QC = travel$ selects questions about travel. Knowing which topic a question belongs to can lead to retrieval effectiveness improvements, for instance by using federated search (Shokouhi and Si, 2011), intent-aware ranking (Glater et al., 2017) or multi-task learning (Liu et al., 2015). Instances from different categories could display different effectiveness values, e.g. questions about *physics* could be a potential difficult category. Question type (5W1H): a categorization into types of question (who, what, where, when, why, how), e.g. $5W1H = what$ selects *what* questions. 5W1H has been used to inform dialogue management modules (Han et al., 2013). The type of question can yield different models' effectiveness (Kim et al., 2019).

3.1.2 Question-Responses based SFs

Question Response Term Match (QDTM): The number of words that appear in both the question and a relevant response is smaller than the threshold T_{QDTM}. The difference in vocabulary, i.e. lexical gap, between queries and documents has shown to be a problem in IR (Lee et al., 2008) and has to lead to remedies such as query expansion (Voorhees, 1994) and the use of neural ranking models for semantic matching (Guo et al., 2019). Responses Lexical Similarity (DLS): average TF-IDF similarity between the top-k most similar responses in the candidate list to the relevant response is higher than the threshold T_{DLS}. The amount of internal coherence, i.e. similarity between responses, has been used to predict query difficulty (He et al., 2008). The SFs can be easily extended for multiple relevant responses, e.g. by using the average or considering one representative relevant response.

3.1.3 Random SFs

The random SF randomly samples $X\%$ of the training data, where X is a hyperparameter.

3.2 Slice-Aware Neural Ranker

Figure 2 displays a diagram of the slice-aware neural ranker. Based on a backbone (BERT) that learns a representation of the question and response concatenation, the slice-aware neural ranker learns to (1) predict how much each instance belongs to each of the k slices or not (supervision is based on the boolean output of the k SFs)[3]; has k slice expert representations with its own set of weights trained using a shared prediction head (2) which predicts relevance for the question and response combination using only instances of the slice k; and (3) combines all representations from the SRAMs using attention into a single slice-aware representation that is used to make the final relevance prediction. The SFs are only used during training and thus are not needed at inference time. This is an adaptation of SRAMs (Chen et al., 2019), and the backbone could be replaced by any other neural ranker.

4 Experimental Setup

We employ four datasets and three retrieval tasks: `MSDialog` (Qu et al., 2018) and `MANtIS` (Penha et al., 2019) for conversation response ranking, `Quora` (Iyer et al., 2017) for similar question retrieval and `ANTIQUE` (Hashemi et al., 2019) for non-factoid question answering. We use the official train, validation and test sets provided by the datasets' creators. As a strong neural ranking baseline model we fine-tune BERT [4] for sentence classification, using the CLS token to predict whether the concatenation of a question and response is relevant or not, following recent research in IR (Nogueira and Cho, 2019; Yang et al., 2019b). Using 512 input tokens (larger inputs are truncated) and a batch size of 8 we train each model for 5 epochs.

When employing SRAMs (Chen et al., 2019) with a BERT backbone for neural ranking using both the question-based and question-responses based SFs we refer to the model as BERT-SA. When using random SFs we refer to the model as BERT-SA-R. For the SFs that have a threshold value (e.g., QL), we choose thresholds that select less than 50% of the data to avoid selecting the majority of the training instances in each slice. For SFs that include a categorical value, e.g., question category (QC) *physics*, we add one slice per category in the dataset. For the random SFs we create 10 different slices[5] for which 50% of randomly chosen instances from the training data belong to[6]. We train each model 5 times with different random seeds and report the test set effectiveness using Mean Average Precision (MAP). ΔMAP indicates the difference between BERT-SA(`-R`) and BERT

[3] The model has an extra SF that all instances belong to, so every instance will always belong to at least to this slice.

[4] *bert-base-uncased* with default hyperparameters (Wolf et al., 2019).

[5] Initial experiments varying the number of SFs showed a validation plateau around 10.

[6] Initial experiments varying revealed that only small percentages, less than 20%, degraded the effectiveness.

Table 1: Average of 5 runs for slice-based learning. Superscript † denote statistically significant improvements over the baseline (BERT) where no slice-based learning is applied at 95% confidence interval using Student's t-tests. Bold indicates the highest MAP for each dataset.

		Dev	Test		
		MAP (std)	MAP (std)	slice Δ MAP	
Dataset	Model			Avg.	Max.
ANTIQUE	BERT	0.853 (.026)	0.850 (.015)	-	-
	BERT-SA-R	0.874 (.025)†	0.877 (.005)†	0.028	0.063
	BERT-SA	**0.878 (.024)**†	**0.883 (.005)**†	0.035	0.112
MANtIS_50	BERT	0.655 (.006)	0.684 (.006)	-	-
	BERT-SA-R	0.671 (.006)†	**0.690 (.014)**†	0.025	0.035
	BERT-SA	**0.702 (.006)**†	0.689 (.022)†	0.025	0.034
MSDialog	BERT	0.754 (.010)	0.830 (.002)	-	-
	BERT-SA-R	**0.815 (.009)**†	**0.840 (.011)**†	0.028	0.084
	BERT-SA	0.810 (.009)†	0.818 (.010)	-0.004	0.067
Quora	BERT	0.799 (.037)	0.819 (.008)	-	-
	BERT-SA-R	0.819 (.035)†	0.837 (.004)	0.011	0.038
	BERT-SA	**0.834 (.034)**†	**0.840 (.007)**†	0.019	0.065

for the slices defined by the SFs.

5 Results

Let us first consider RQ1. We observe in Table 1 that with the exception of MSDialog, BERT-SA significantly improves over the baseline (BERT) for both the overall (column MAP) and per slice performance (column slice ΔMAP). **This demonstrates that slice-based learning is useful for neural ranking, with gains up to 3.8% overall and up to 13% per slice in terms of MAP.**

To better understand which features of a slice correlate the most with the observed gains from BERT-SA, we study how three properties of the slices correlate with the slice ΔMAP (i.e., the improvement over BERT): we consider (1) the size of the slice, (2) the classification accuracy of the slice-aware model to predict slice membership, and, (3) the BERT model effectiveness for each slice. The only property that has a statistically significant Pearson correlation (0.504 average for the different datasets) with MAP gains is the BERT baseline performance , suggesting that focusing on failures of neural ranking models (slices for which BERT has low effectiveness) when implementing SFs is effective.

To provide insights into the underlying reasons of the effectiveness of slice-based learning (RQ2), we replace the SFs that capture error categories with random SFs, i.e. BERT-SA-R. We find that this model also has a significantly better effectiveness than the BERT baseline, with the exception of Quora. **This indicates that part of the gains provided by slice-based learning could be attributed to the effect of ensemble learning**, since each slice-aware representation is trained on random parts of the data and are then combined[7]. We note however that the slice gains of BERT-SA are higher than BERT-SA-R for ANTIQUE and Quora with statistical significance. This indicates that not all improvements of slice-based learning can be attributed to the effect of ensemble learning and carefully implementing SFs is advantageous.

6 Conclusion

In this paper we demonstrated that a slice-aware neural ranker is an effective approach to IR, increasing the effectiveness of rankers by margins up to 3.8% overall and up to 13% per slice in terms of MAP. As future work we plan to study slice-aware neural rankers that do listwise optimization—such a ranker could learn better representations particularly for SFs that uses several responses as input.

[7] Another potential reason for the success of slice-based learning could be the capacity obtained by the additional number of weights compared to the baseline (e.g. from 110M to 116M for MANtIS).

Acknowledgements

This research has been supported by NWO projects SearchX (639.022.722) and NWO Aspasia (015.013.027).

References

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. *Computer Science Department Faculty Publication Series*, page 189.

Michael Bendersky and W Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Workshop on Web Search Click Data*, pages 8–14.

Arthur Câmara and Claudia Hauff. 2020. Diagnosing bert with retrieval heuristics. In *European Conference on Information Retrieval*, pages 605–618. Springer.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. 2019. Slice-based learning: A programming model for residual learning in critical data slices. In *NeurIPS*, pages 9392–9402.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, pages 4171–4186.

Thomas G Dietterich et al. 2002. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125.

Rafael Glater, Rodrygo LT Santos, and Nivio Ziviani. 2017. Intent-aware semantic query annotation. In *SIGIR*, pages 485–494.

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. *arXiv preprint arXiv:1903.06902*.

Sangdo Han, Kyusong Lee, Donghyeon Lee, and Gary Geunbae Lee. 2013. Counseling dialog system with 5w1h extraction. In *SIGDIAL*, pages 349–353.

Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W Bruce Croft. 2019. Antique: A non-factoid question answering benchmark. *arXiv preprint arXiv:1905.08957*.

Ben He and Iadh Ounis. 2006. Query performance prediction. *Information Systems*, 31(7):585–594.

Jiyin He, Martha Larson, and Maarten De Rijke. 2008. Using coherence-based measures to predict query difficulty. In *ECIR*, pages 689–694. Springer.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. *data. quora. com.*

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different nlp tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*.

Jung-Tae Lee, Sang-Bum Kim, Young-In Song, and Hae-Chang Rim. 2008. Bridging lexical gaps between queries and questions on large online q&a collections with compact translation models. In *EMNLP*, pages 410–418.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Gustavo Penha, Alexandru Balan, and Claudia Hauff. 2019. Introducing MANtIS: a novel Multi-Domain Information Seeking Dialogues Dataset. *arXiv preprint arXiv:1912.04639*.

Nina Phan, Peter Bailey, and Ross Wilkinson. 2007. Understanding the relationship of information need specificity to search query length. In *SIGIR*, pages 709–710.

Chen Qu, Liu Yang, W Bruce Croft, Johanne R Trippas, Yongfeng Zhang, and Minghui Qiu. 2018. Analyzing and characterizing user intent in information-seeking conversations. In *SIGIR*, pages 989–992.

Milad Shokouhi and Luo Si. 2011. Federated search. *Foundations and Trends® in Information Retrieval*, 5(1):1–102.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. Multi-representation fusion network for multi-turn response selection in retrieval-based chatbots. In *WSDM*, pages 267–275.

Ellen M Voorhees. 1994. Query expansion using lexical-semantic relations. In *SIGIR*, pages 61–69.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *ACL*, pages 747–763.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019a. Critically Examining theNeural Hype: Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *SIGIR*, pages 1129–1132, New York, NY, USA.

Wei Yang, Haotian Zhang, and Jimmy Lin. 2019b. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*.