Learning to Rank with BERT for Argument Quality Evaluation

Charles-Olivier Favreau, Amal Zouaq and Sameer Bhatnagar
Polytechnique Montreal
2500 Chem. de Polytechnique, Montreal, Canada

Abstract

The task of argument quality ranking, which identifies the quality of free text arguments, remains, to this day, a challenge. While most state-of-the-art initiatives use point-wise ranking methods and predict an absolute quality score for each argument, we instead focus on learning how to order them by their relative convincingness, experimenting with several learning-to-rank methods for argument quality. We leverage BERT's powerful ability in building a representation of an argument, paired with learning-to-rank approaches (pointwise, pairwise, list-wise) to rank arguments according to their measure of convincingness. We also demonstrate how an ensemble of models trained with different ranking losses often improves the performance at identifying the most convincing arguments of a list. Finally, we compare BERT coupled with learning-to-rank methods to state-of-the-art approaches on all major argument quality datasets available for the ranking task, demonstrating how a learning-to-rank approach generally performs better at outlining the topmost convincing arguments.

Introduction

Establishing a position on a topic is defined by someone's ability to defend a stance. Argumentation is a tool to convince an audience of a stance on a given topic. For example, given the topic "Zoos should be abolished" and the stance "Pro", one could argue that a zoo's whole business model is to take animals from their natural habitats and exploit them for money (Dataset *IBM ArgQ 30k* (Gretz et al. 2020)). This yields the question: what defines a good argument? Or, how can one identify a convincing argument? Natural language processing defines this task as the automatic assessment of argument quality. In fact, (Habernal and Gurevych 2016) proposed 2 tasks in the field of computational argumentation: First, the task of predicting the most convincing argument out of a pair of arguments and second, the task of ranking a list of arguments, according to their convincingness. While the performance of state-of-the-art models on the first task is impressive, the task of ranking arguments in order of convincingness proves to be more challenging, as demonstrated by (Toledo et al. 2019). As one would expect, ranking a list of arguments is more complex than simply choosing the most convincing argument out of a pair of arguments.

In this work, we focus on the second task: ranking a list of arguments for a given topic, in order of convincingness. Most proposed solutions so far approached the task as predicting an absolute quality score for each argument individually, defined as point-wise ranking. While these methods produce the wanted results as the list of the predicted scores of convincingness can be sorted to show the ranking of the arguments, we hypothesize that there are ranking capabilities potentially lost during the learning process by not comparing the arguments together. We propose to define the problem as a true ranking task, where we do not evaluate each argument's individual measure of quality, but instead focus on evaluating its relative convincingness compared to other arguments.

More generally, we try to answer the following research question: *How can learning to rank techniques coupled with pretrained language models contribute to automatic argument quality evaluation?* In this work, we leverage a neural approach to *learning-to-rank*, built on top of BERT (Devlin et al. 2018). This method combines BERT's strong ability to build an argument's representation, and different ranking loss functions (pointwise, pairwise, list-wise). The main contribution of our work is to demonstrate the effectiveness of neural approaches to learning-to-rank on the task of ranking arguments by their measure of convincingness. We demonstrate how our method delivers stronger capability in outlining the *topmost convincing arguments* of a list.

This paper is organized as follows. First, we present previous state-of-the-art approaches in the section Related Work. Second, we describe the datasets used to compare our approach to previous ones. We explain the architecture of our solution, our methodology and the performance metrics used to evaluate our model in the section Learning-to-rank. The section Results compares our approach to state-of-the-art models.

Related Work

As evaluating the quality of an argument or, in other words, its convincingness, remains a challenge in the field, we review the main methods presented in the last 5 years for automatic argument quality evaluation.

First task: Predicting the most convincing argument of a pair

(Habernal and Gurevych 2016) pioneered the task of assessing argument convincingness by defining it as the following problem: choosing the most convincing argument out of a pair of arguments with the same stance on the same topic. Many notable state-of-the-art solutions outperform (Habernal and Gurevych 2016)'s initial model. (Simpson and Gurevych 2018) propose to utilize scalable Bayesian preference learning for argument quality evaluation, producing a classifier which achieves impressive results. (Gleize et al. 2019) present a Siamese BLSTM using word2vec embeddings. Finally, (Toledo et al. 2019) demonstrate how BERT with a binary classification head outperforms both previously cited solutions.

Second task: Ranking list of arguments

Most state-of-the-art methods to solve the task of ranking a list of arguments for a given stance on a topic consist of point-wise ranking. (Simpson and Gurevych 2018), along with their model presented for the first task, also present a model leveraging scalable Bayesian preference learning for the regression task of predicting a quality score for every argument in a list. (Gleize et al. 2019) reuse one leg of their Siamese BLSTM from the first task to predict a quality score for each argument individually. Both (Toledo et al. 2019) and (Gretz et al. 2020), initiatives by IBM research, present BERT with a regression head to solve the task at hand. To predict a quality score, BERT takes 2 sequences as input: the topic and the argument.

(Potash, Ferguson, and Hazen 2019) stand out from other state-of-the-art approaches by proposing a learning-to-rank architecture similar to RankNet (Burges et al. 2005) using a sum of word embeddings for the representation of an argument. The model produces scores independently for each argument, normalizing the scores of argument pairs using the Softmax function. It is trained on preference pairs of arguments and, therefore, can be trained on the pairwise labeled datasets. The trained model is then evaluated on the classification task or the ranking task. (Potash, Ferguson, and Hazen 2019) manage to outperform previous state-of-the-art solutions considering Spearman ranking metric.

Looking at state-of-the-art methods in argument quality evaluation, we can observe the effectiveness of using BERT to build embedding representation of the arguments. In fact, methods using BERT deliver the best performance for predicting the most convincing argument of a pair. However, on the task of ranking a list of arguments, point-wise approaches to learning-to-rank fall short. (Potash, Ferguson, and Hazen 2019) demonstrate the benefits of moving to a pairwise approach. Based on those observations, in this work, we compare a pointwise, pairwise and list-wise approach to learning-to-rank on top of BERT, evaluated on all the major argument quality datasets.

Datasets

In this section, we briefly describe all the major publicly available argument quality datasets released in the last 5 years and used in our experiments. Descriptive statistics are shown in Table 1. These datasets present differences in the way the arguments were gathered and the way they were annotated.

For instance, the UKP datasets (Habernal and Gurevych 2016) were annotated as pairs through crowdsourcing, where each annotator had to choose the most convincing argument out of the pair (aka pair annotations). (Habernal and Gurevych 2016) also rank the arguments in order of convincingness for each topic, by computing a score for each argument from the pair annotations. They build a graph representation where nodes represent arguments and directed edges represent pair annotations. PageRank is applied to rank all the arguments for each topic, resulting in the dataset *UKP-ConvArgRank* which is used for the task of ranking arguments for each topic.

Similarly, *IBM-EviConv* is annotated as a set of evidence pairs extracted from Wikipedia. To extract rankings from those annotations, (Gleize et al. 2019) use one leg from the Siamese BLSTM trained on the pair annotations, to generate a score for each argument.

IBM-ArgQ is built using two different labelling approaches: each individual argument is annotated with an absolute quality score and also, argument pairs are labelled (similar to previous approaches). This resulted in, as per other initiatives, two datasets: one for the task of predicting the most convincing argument of a pair and the other for the task of ranking a list of arguments for a topic. However, in this case, both datasets were built directly from human annotators, avoiding a transformation step such as in (Habernal and Gurevych 2016) and (Gleize et al. 2019).

In the case of *IBM-ArgQ-Rank-30k*, the arguments are annotated directly with an individual score. The arguments are annotated as a binary decision. For each argument, the annotators were asked if they would recommend a friend to use that argument or not. Each argument is annotated by 10 people. A continuous quality score, between 0 and 1, is then derived from these binary annotations. (Gretz et al. 2020) uses two different ways of deriving that score: a weighted-average (WA) and the MACE probability (MACE-P). This process yields a dataset with a continuous quality score for each argument, aiming at the task of ranking the arguments for each topic.

Learning-to-rank

A learning-to-rank approach is defined by the scoring function it uses, and the loss function applied to the produced scores.

BERT Learning-to-rank model

Focusing on the ranking task, the learning objective is, given a Topic T, a stance S and a list of arguments A_1, A_2, ..., A_n, to assign a rank to each argument A_i from the most convincing rank to the least convincing rank. We propose to use TFR-BERT (Han et al. 2020), a learning-to-rank approach paired with BERT. A ranking head is used on top of BERT, allowing to apply a ranking loss function (see the section Ranking Loss Functions) over multiple arguments at once.

Dataset name	Number of arguments	Number of topics	Task	Source	Arg Length			Topic Length			Mean Topic Arg Count
					Mean	Min	Max	Mean	Min	Max	
UKPConvArg1Strict	11650 pairs of arguments	32	PA	Extracted from createdebate.com, convinceme.net	263	37	753	56	26	92	32
UKPConvArg1-Ranking	1052 arguments	32	Ranking								
IBM-ArgQ-Pairs	9100 pairs of arguments	22	PA	Actively collected arguments from crowds	138	36	275	42	31	63	240
IBM-ArgQ-Args	5300 arguments	22	Ranking								
IBM-EviConv	5697 pairs of arguments	69	PA & Ranking	Automatically retrieved Wikipedia sentences	189	60	495	34	20	55	26
IBM-ArgQ-Rank-30k	30000 arguments	71	Ranking	Actively collected arguments from crowds	107	35	251	34	21	52	429

Table 1: Statistics on most common datasets for the argument quality evaluation task. PA stands for Pair Classification.

This neural approach to learning-to-rank is implemented using the TF Ranking library (Pasumarthi et al. 2018).

Input representation The ranking model needs to be able to grasp the quality of an argument with respect to a topic. The BERT module is responsible for building a representation demonstrating the association between the argument and the topic. Each argument's text is concatenated to its respective topic's text, in a typical BERT pair representation: [CLS] *Topic* [SEP] *Argument* [SEP].

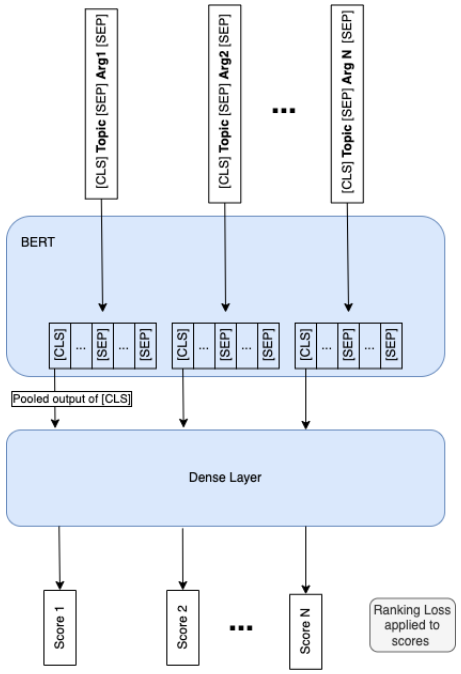

Figure 1: Architecture of the BERT ranking model based on (Han et al. 2020).

Architecture As shown in figure 1, for each argument, the BERT module takes a topic & argument concatenated sequence and outputs the hidden units of the [CLS] token of the last layer. The pooled outputs of each topic & argument sequence, for every argument in the list to rank, are fed into a dense layer which acts as a scoring function. The scoring function learns to associate a score to every argument. A ranking loss function is applied to the scores generated by the scoring function and is used to update the model's weights. The loss function used determines how many arguments are considered at once when calculating the loss for backpropagation over the model's weights.

Ranking Loss Functions

In this work, we compare the performance of 3 types of ranking losses, introducing list-wise ranking loss functions to argument quality evaluation. When training a ranking model, the loss function can either be applied to the arguments individually (pointwise loss), by pairs (pairwise loss) or altogether (listwise loss). For point-wise losses, the arguments are considered independently. Pairwise losses use argument pairs to calculate the loss. List-wise losses consider the order of the whole list of arguments. Table 2 shows the specific loss functions we explore for each type of loss.

Loss Type	Loss Function	Equation
Pointwise	Mean Squared Loss	$\mathcal{L}(\{y\},\{s\}) = \sum_i (y_i - s_i)^2$
Pairwise	Pairwise Hinge Loss	$\mathcal{L}(\{y\},\{s\}) = \sum_i \sum_j I[y_i > y_j] \max(0, 1 - (s_i - s_j))$
	Logistic Loss	$\mathcal{L}(\{y\},\{s\}) = \sum_i \sum_j I[y_i > y_j] \log(1+\exp(-(s_i - s_j)))$
Listwise	List MLE Loss	$\mathcal{L}(\{y\},\{s\}) = -\log(P(\pi_y \mid s))$ where $P(\pi_y \mid s)$ is the Plackett-Luce probability of a permutation π_y conditioned on scores s.
	Softmax Loss	$\mathcal{L}(\{y\},\{s\}) = -\sum_i y_i \cdot \log\left(\frac{exp(s_i)}{\sum_j exp(s_j)}\right)$
	Approx NDCG Loss	$\mathcal{L}(\{y\},\{s\}) = -\frac{1}{DCG(y,y)} \sum_i \frac{2^{y_i} - 1}{\log_2(1 + rank_i)}$ $rank_i = 1 + \sum_{j \neq i} \frac{1}{1 + \exp\left(\frac{-(s_j - s_i)}{temperature}\right)}$

Table 2: Equations of loss functions, where s is the list of scores predicted and y is the list of scores used as ground truth.

Transforming scores into ranks

For all the datasets mentioned in the section Datasets, the quality score is an absolute value and cannot be directly used with a learning to rank model. We must first sort all the arguments for a given topic by quality score, from lowest to highest. From that sorted list of arguments, we attribute a relevancy rank to each of the argument, with the highest rank for the highest score. To transform scores into ranks, we use the function rankdata from the Scipy library [1]. To deal with

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rankdata.html

arguments with tied scores, we choose the strategy 'dense' to transform the scores into ranks in a way that limits the range of the rank values. This implies only a single rank value is assigned to arguments with tied scores, and ensures the ranking model is able to learn to rank two arguments as equal if they have the same quality score.

Training parameters

Maximum sequence length The maximum length of the sequence passed as input to the BERT module is calculated for every dataset. It is fixed to a value that ensures 95% of the arguments aren't truncated (sequence length including the topic & the argument combined as seen in the section Learning-to-rank). For the rest of the arguments, the total sequence is truncated to the fixed maximum length. The reason for this decision is to facilitate the training. Choosing a maximum sequence length that would ensure the remaining 5% of the arguments aren't truncated usually increased drastically the sequence length and caused the model to be heavier to train.

Argument batches During the training phase, memory limits did not allow to fit the whole list of arguments for a topic. Thus, for each topic in our datasets, we divide the list of arguments into smaller lists of 12 arguments as shown in table 3, the same list size used by (Han et al. 2020). At inference time, however, full lists of arguments are fed to the model for predictions, ensuring the model is evaluated on unmodified data (test set).

Tied ranks and scores The methodology used to divide a list of arguments into smaller lists of 12 arguments must take into account the number of arguments with the same score, otherwise it affects the training of the model. In fact, many arguments have the same score, especially the most convincing ones and the least convincing ones, resulting in equal ranks. Feeding the model with batches of arguments with equal ranks would result in poor training. Consequently, we divide the list of arguments in such a way that each batch has arguments of rank values well spread across the rank range. To do so, we divide the list of arguments, sorted by convincingness, into 12 slices. Each batch_i takes argument i of every slice, generating uniform batches of size 12, while ensuring no argument overlap between batches. In other words, every list of arguments fed to the model for training contains strong and poor quality arguments, as well as arguments considered relatively convincing. This allows for effective learning of the ranking function.

Loss Function	Learning rate	EPOCH	optimizer	Train Batch size	Dropout rate	List size
MSE Loss	1e-5	2	adam	6	0.1	12
Hinge Loss	1e-6	3				
Logistic Loss						
List MLE						
Softmax Loss						
Approx NDCG Loss						

Table 3: Training parameters of TFR-BERT for the argument quality ranking task.

Ensemble TFR-BERT Han et al. demonstrate how an ensemble approach to TFR-BERT, combining multiple ranking losses, can improve predictions. We use the same approach for the task of argument quality, combining the predictions of multiple versions of TFR-BERT, each trained using a different ranking loss. For each prediction, we average the list of scores predicted over the different versions of TFR-BERT. This increased the model's performance considerably, as shown in section Results.

Metrics

Most initiatives in the argument quality evaluation field used Pearson & Spearman to evaluate the ranking task. As we focus more on the ranking perspective of the task and less on predicting an absolute score for each argument, we employ the NDCG (Normalized Discounted Cumulative Gain) to evaluate our model's performance, as it is a metric commonly used for learning-to-rank.

We also report our results using Pearson, Spearman and Kendall's Tau to compare our results to other initiatives in the state-of-the-art.

Results

We evaluate the model TFR-BERT, presented in the section Learning-to-rank, on all major argument quality evaluation datasets available. When a division into train, validation and test sets was not provided by the dataset, we divided them as 20% of the topics assigned to the test set, 20% assigned to the validation set and the remaining to the train set. Table 4 shows descriptive statistics on the train, validation and test sets. Test sets never contain a topic already seen by the model during training. For reproducibility purposes, we provide all the datasets[2] as lists of ordered arguments following the methodology described in the section Transforming scores into ranks.

Dataset	Train		Valid		Test	
	Topic	Args	Topic	Args	Topic	Args
UKP Rank	18	602	7	222	7	228
IBM Evi	36	6632	12	2006	21	2756
IBM ArqQ Rank	12	2625	5	1586	5	1087
IBM Arg 30K	49	20974	7	3208	15	6315

Table 4: Division of datasets into train, valid and test sets.

UKP Rank

We first evaluate the performance of TFR-BERT on the *UKP Rank* dataset for the argument quality ranking task, comparing different loss functions. Table 5 shows Pearson, Spearman, Kendall's Tau as well as the NDCG@K metrics for every model. The first 3 metrics give a measure of how well the model ranks all the arguments of the list. The NDCG@K metrics show how the model performs at outlining the top K most convincing arguments. We can see in table 5 that the majority of TFR-BERT variants outperform BERT across many metrics, including models trained with pointwise, pairwise and list-wise losses. The

[2] http://www.labowest.ca/FLAIRS2022/datasets

	Loss	Model	PEARSON	SPEARMAN	TAU	NDCG@5	NDCG@10	NDCG@15
UKP Rank	Pointwise	BERT	0.44	0.56	0.40	0.53	0.62	0.68
		TFR-BERT MSE Loss	0.45	0.68	0.51	0.59	0.67	0.72
	Pairwise	TFR-BERT Hinge Loss	0.44	0.60	0.46	**0.63**	**0.72**	0.75
		TFR-BERT Logistic Loss	0.38	0.59	0.45	0.43	0.57	0.61
		State-of-the-art: Sum-of-Words-Embeddings + FFNN	**0.48**	**0.69**	**0.52**	-	-	-
	List-wise	TFR-BERT Softmax Loss	0.40	0.67	0.51	0.49	0.61	0.66
		TFR-BERT List MLE	0.36	0.61	0.45	0.36	0.54	0.60
		TFR-BERT Approx NDCG Loss	0.47	0.59	0.44	0.54	0.66	0.69
	Mix	**TFR-BERT Ensemble Losses**	**0.48**	0.68	0.51	0.60	**0.72**	**0.77**
IBM Evi	Pointwise	BERT	0.57	0.51	0.37	0.88	**0.90**	0.89
		TFR-BERT MSE	0.56	0.50	0.36	0.90	**0.90**	0.91
	Pairwise	TFR-BERT Hinge Loss	0.53	0.46	0.34	0.88	0.88	0.88
		TFR-BERT Logistic Loss	0.37	0.36	0.26	0.86	0.84	0.86
	List-wise	TFR-BERT Softmax Loss	0.60	0.54	0.39	**0.91**	**0.90**	**0.92**
		TFR-BERT list MLE	0.38	0.29	0.21	0.77	0.79	0.81
		TFR-BERT Approx NDCG Loss	0.55	0.52	0.36	0.90	0.89	0.80
	Mix	**TFR-BERT Ensemble Losses**	**0.61**	**0.56**	**0.41**	**0.91**	0.89	0.89
IBM ArgQ Rank	Pointwise	**State-of-the-art:** BERT	**0.42**	**0.41**	0.22	0.55	0.60	0.63
		TFR-BERT MSE	0.30	0.29	0.20	0.63	0.64	0.66
	Pairwise	TFR-BERT Hinge Loss	0.31	0.31	0.21	0.61	0.63	0.64
		TFR-BERT Logistic Loss	0.33	0.34	**0.24**	0.60	0.63	0.66
	List-wise	TFR-BERT Softmax Loss	0.34	0.33	0.23	0.57	0.61	0.62
		TFR-BERT List MLE	0.32	0.31	0.21	0.58	0.61	0.64
		TFR-BERT Approx NDCG Loss	0.29	0.32	0.22	0.62	0.64	**0.67**
	Mix	**TFR-BERT Ensemble Losses**	0.35	0.34	0.23	**0.64**	**0.67**	0.66
IBM Arg 30K	Pointwise	**State-of-the-art:** BERT	**0.52**	**0.48**	**0.32**	0.85	0.87	0.86
		TFR-BERT MSE	0.50	0.45	0.32	0.87	0.87	0.87
	Pairwise	TFR-BERT Hinge Loss	0.49	0.45	0.31	**0.90**	0.89	0.88
		TFR-BERT Logistic Loss	0.50	0.45	0.31	0.88	0.88	0.88
	List-wise	TFR-BERT Softmax Loss	0.49	0.43	0.30	0.86	0.86	0.86
		TFR-BERT List MLE	0.51	0.45	0.32	0.89	**0.90**	**0.89**
		TFR-BERT Approx NDCG Loss	0.43	0.42	0.30	0.88	0.87	0.87
	Mix	**TFR-BERT Ensemble Losses**	**0.52**	0.47	**0.32**	0.89	0.89	0.88

Table 5: Evaluation of TFR BERT using different ranking losses on all major argument quality datasets.

Ensemble TFR-BERT, which combines models trained with *MSE*, *Hinge*, *Pairwise Logistic* and *Approx NDCG* losses respectively, is the best performing variant of TFR-BERT. Comparing it to the state-of-the-art (Potash, Ferguson, and Hazen 2019)'s Sum-of-Words-Embeddings with Feed Forward Neural Network, we find that Ensemble TFR-BERT performs similarly to their solution for Pearson, Spearman and Kendall's Tau metrics. Ensemble losses and the *Pairwise Hinge* loss are the best performing TFR-BERT variants according to NDCG@K metrics, outperforming BERT by a significant margin.

IBM Evi Dataset

For model evaluation on *IBM Evi*, we used the exact same test set as in the published dataset (Gleize et al. 2019). We then reserved 25% of the training set for the validation set, as shown in table 4. Table 5 shows that 2 variants of TFR-BERT outperform BERT for Pearson, Spearman & Kendall's Tau metrics: TFR-BERT trained with *Softmax* loss function and Ensemble TFR-BERT, which combines models trained with *MSE*, *Softmax* and *Approx NDCG* losses respectively. Those two variants of TFR-BERT also have the edge on BERT for the NDCG@5 metric. TFR-BERT trained with *Softmax* loss function is the best performing model across NDCG@K metrics, while Ensemble TFR-BERT is the best performing model across Pearson, Spearman & Tau metrics.

IBM ArqQ Rank

Table 5 shows that *IBM ArqQ Rank* was the most challenging dataset to rank for TFR-BERT. Variants of TFR-BERT managed to outperform (Toledo et al. 2019)'s state-of-the-art BERT on Kendall's Tau and NDCG@K metrics. Ensemble TFR-BERT, which combines models trained with *MSE*, *Hinge*, *Logistic*, *Softmax* and *list MLE* losses respectively, remains the best performing model across most metrics on *IBM ArqQ Rank*.

IBM Arg 30K

To ensure a proper comparison to the state-of-the-art on dataset *IBM Arg 30K*, we use the exact same division into

train, validation and test sets as described in the published dataset (Gretz et al. 2020), as shown in table 4. Comparing the TFR-BERT architecture to BERT, we can see that Ensemble TFR-BERT, which combines models trained with *MSE*, *Hinge*, *Logistic*, *Softmax* and *list MLE* losses respectively, performs similarly to BERT on Pearson, Spearman & Kendall's Tau but outperforms it on NDCG@K metrics.

Discussion

The results presented demonstrate that TFR-BERT, evaluated on every major argument quality dataset, generally outperforms state-of-the-art solutions on NDCG@K metrics and performs similarly to the state-of-the-art on Pearson, Spearman metrics & Kendall's Tau. The model's performance for the NDCG@K metric shows the model is excellent at returning the top K most *convincing* arguments. Considering applications of argument quality ranking, one could say that returning the top K best arguments of a list has more value than the whole ranked list in itself. This reinforces our call for the usage of the NDCG@K metric for the task of argument quality ranking.

Comparing the different types of ranking losses, we can observe the pairwise and list-wise ranking losses usually performed better for the NDCG@K metrics, and thus at identifying top K most convincing arguments of a list. While one loss function did not stand out as generally the best across datasets, an ensemble model of multiple TFR-BERT trained with different loss functions always yielded better results. On almost every dataset it was evaluated on, ensemble TFR-BERT outperforms every TFR-BERT trained using only one ranking loss function, generally performing more uniformly across all metrics, demonstrating a more robust approach to argument quality ranking.

From the results in table 5, we can observe a noticeable difference in performance from one dataset to another. Each dataset has their own score of quality, each different in the way the score is calculated, either in the way it is transformed from pairwise annotations or the way it is collected. In future work, we plan to conduct a qualitative evaluation of the datasets and also explore the feasibility of using a normalized score for all argument quality datasets, thus unifying them.

Conclusion

In this work, we propose a different view on the task of ranking arguments by quality. Steering away from trying to predict an absolute quality score for each argument, we instead focus on learning how to order them by their relative convincingness. We propose to use an architecture based on learning-to-rank built on top of BERT. Pairing a learning-to-rank approach with BERT's powerful ability in building a representation of an argument yields stronger ranking capabilities. This shows in the results obtained using TFR-BERT, which demonstrate better performance for the NDCG@K metrics, and thus superior capability at outlining top K most convincing arguments. We argue that this might have more significant applications than focusing on ranking all the arguments with equal importance. We also demonstrate how combining multiple ranking loss functions (pointwise, pairwise and list-wise) as an Ensemble model of TFR-BERT shows better performance across many metrics. In future work, we want to explore neural learning-to-rank methods with other pre-trained language models: RoBERTa or Electra, for the argument quality ranking task.

Acknowledgements

This research is supported by an INSIGHT grant from the Social Sciences and Humanities Research Council of Canada (SSHRC).

References

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, 89–96. New York, NY, USA: Association for Computing Machinery.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Gleize, M.; Shnarch, E.; Choshen, L.; Dankin, L.; Moshkowich, G.; Aharonov, R.; and Slonim, N. 2019. Are you convinced? choosing the more convincing evidence with a Siamese network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 967–976. Florence, Italy: Association for Computational Linguistics.

Gretz, S.; Friedman, R.; Cohen-Karlik, E.; Toledo, A.; Lahav, D.; Aharonov, R.; and Slonim, N. 2020. A large-scale dataset for argument quality ranking: Construction and analysis. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(05):7805–7813.

Habernal, I., and Gurevych, I. 2016. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1589–1599. Berlin, Germany: Association for Computational Linguistics.

Han, S.; Wang, X.; Bendersky, M.; and Najork, M. 2020. Learning-to-rank with BERT in tf-ranking. *CoRR* abs/2004.08476.

Pasumarthi, R. K.; Wang, X.; Li, C.; Bruch, S.; Bendersky, M.; Najork, M.; Pfeifer, J.; Golbandi, N.; Anil, R.; and Wolf, S. 2018. Tf-ranking: Scalable tensorflow library for learning-to-rank. *CoRR* abs/1812.00073.

Potash, P.; Ferguson, A.; and Hazen, T. J. 2019. Ranking passages for argument convincingness. In *Proceedings of the 6th Workshop on Argument Mining*, 146–155. Florence, Italy: Association for Computational Linguistics.

Simpson, E., and Gurevych, I. 2018. Finding convincing arguments using scalable Bayesian preference learning. *Transactions of the Association for Computational Linguistics* 6:357–371.

Toledo, A.; Gretz, S.; Cohen-Karlik, E.; Friedman, R.; Venezian, E.; Lahav, D.; Jacovi, M.; Aharonov, R.; and Slonim, N. 2019. Automatic argument quality assessment – new datasets and methods.