Retrieving Comparative Arguments using Deep Pre-trained Language Models and NLU

Notebook for the Touché Lab on Argument Retrieval at CLEF 2020

Viktoriia Chekalina[‡,†] and Alexander Panchenko[‡]

[‡]Skolkovo Institute of Science and Technology, Moscow, Russia
[†]Philips Research Lab RUS, Moscow, Russia

Abstract In this paper, we present our submission to the CLEF-2020 shared task on Comparative Argument Retrieval. We propose several approaches based on state-of-the-art NLP techniques such as Seq2Seq, Transformer, and BERT embedding. In addition to these models, we use features that describe the comparative structures and comparability of text. For the set of given topics, we retrieve the corresponding responses and rank them using these approaches. Presented solutions could help to improve the performance of processing comparative queries in information retrieval and dialogue systems.

1 Introduction

People are faced with a multitude of choice problems on a daily basis. This type of questions can be related to products, e.g., which milk producer to trust, which fruit contains less sugar, or which laptop brand is more reliable. Another popular type of comparison is related to travel destinations, e.g., which cities or national parks to visit. The comparative questions can also involve more complex objects/matters of comparison, e.g., which country is safer to raise children, Germany, or the United States? Finally, the fuzziness of comparative questions can go even further into some philosophical questions with possibly no definitive answer, e.g., which political system is better to maximize the overall average happiness of a population. Therefore, the *comparative information need* is an omnipresent type of information need of users.

While for some categories of products, e.g., mobile phones and digital cameras, tools for side-to-side comparison of features are available, for many domains, e.g., programming languages or databases, this information is not well structured. On the other hand, the Web contains a vast number of opinions and objective arguments that can facilitate the comparative decision-making process. The goal of our work is to develop methods for the retrieval of such textual documents, which are highly relevant for fulfilling the various comparative information needs of the users. Recent research on this topic touched on some aspects of the comparative question answering, e.g., retrieval human-computer interaction interface for comparative queries [17], classification of comparative questions [3] or extraction of objects and aspect from comparative texts [1], inter alia. However, the quality of retrieval of comparative answers was not evaluated to date.

More specifically, this notebook contains a description of our approach used in the submission of the CLEF-2020 shared task on Comparative Argument Retrieval[1] including all details necessary to reproduce our results. The source codes and data used in our submission are also available online.[2] The contribution of our work is three-fold:

1. We are first to use various deep pre-trained language models, such as ULMFiT and Transformer-based, on the task of comparative argument retrieval.
2. We are first to experiment with features based on specialized sequence taggers of comparative structures (detection objects, predicates, and aspects of comparison) that implement a shallow Natural Language Understanding (NLU).
3. We are first to experiment with features based on the density of comparative sentences in a text (based on a pre-trained classifier of comparative sentences [13]).

The remainder of this paper is organized as following: Section 2 introduces the task, then Section 3 presents several variations of the method we proposed. In Section 4, the results of the experiments based on the manual evaluation are discussed. Finally, Section 5 summarizes the main findings and directions for future work.

```
–<topic>
   <number>38</number>
  –<title>
     What are the differences between MySQL and PostgreSQL in performance?
   </title>
  –<description>
     Before starting a new DB-related project, a software developer wants to find some tips about when to use which database. Back in their studies
     some years ago, they had learnt that the choice of a database management system is important when starting a new project. Not having too much
     experience with database-related software development, the user just remembers that historically, MySQL had been a default choice for creating
     and maintaining databases. Even though the performance differences between MySQL and PostgreSQL have been largely eliminated in recent
     versions, there still are differences worth considering like usage of indices, default installation, what types of replication / clustering are available
     and so on. Getting to know these issues will help the developer make a choice for the project.
   </description>
  –<narrative>
     Highly relevant documents discuss differences between MySQL and PostgreSQL focusing on their performance such as query throughput on what
     hardware or time to write large amounts of data, but also development and support effort, etc. Highly relevant documents should provide a
     conclusion on main differences between the two database management systems and about typical scenarios that would favor either option.
     Relevant documents may help to form an opinion on the performance of either database by for instance describing the usage of one of the systems
     in some specific scenario(s). Documents discussing the history of MySQL and PostgreSQL, providing materials to learn SQL syntax, tutorials
     about database usage, etc. that do not elaborate on MySQL or PostgreSQL performance are not relevant.
   </narrative>
 </topic>
```

Figure 1. An example of a topic which specifies a user's comparative information need. Such topics are inputs to our methods. The goal of our methods is to retrieve Web documents which would fulfil information need specified in such topics and helping to make an informed choice.

2 Task: Retrieval of Comparative Arguments on the Web

The track Touché [4] suggests the following goal: given a set of topics, one needs to retrieve and rank documents according to their relevance. The relevant documents are

[1] https://events.webis.de/touche-20/shared-task-2.html
[2] https://github.com/skoltech-nlp/touche

those which are helpful in making the comparative decision, i.e., those which directly compare the target objects facing their pros and cons.

The topic contains a question implying comparison of the two objects, i.e. “What is better, a laptop or a desktop?”, “Which is better, Canon or Nikon?”, “Should I buy or rent?”. An example of the topic is presented on Figure 1. Each topic consists of a title, e.g., a short description similar to those in which a user could enter into an information retrieval engine but also contains two additional fields: description and narrative. These fields specify more closely the context and semantics of the topic, and these are actually used by human annotators to perform judgments of the retrieved documents. In our experiments, we only used the “title” field.

We use topic title as a query in ChatNoir search engine [15][3], which extracts documents from ClueWeb12 corpus.[4] In response to the query ChatNoir returns a set of documents which contains titles, body texts, documents identifiers and search engine’s scores. We try to retrieve 1000 unique documents, but for some queries the system gives less.

The goal of our methods is to find documents that most reliably and completely answer the query question in this set of pre-retrieved candidates. In other words, the document should be relevant to the topic, be trustworthy, and give an entirely and reasonable comparison.

3 Methodology

The main objective of the experiments is to develop a method finding among retrieved documents that meet the comparative criteria most fully and reasonably.

In addition to the search system’s scoring, we employ pre-trained state-of-the-art language models and methods for getting the rate of the document’s comparability.

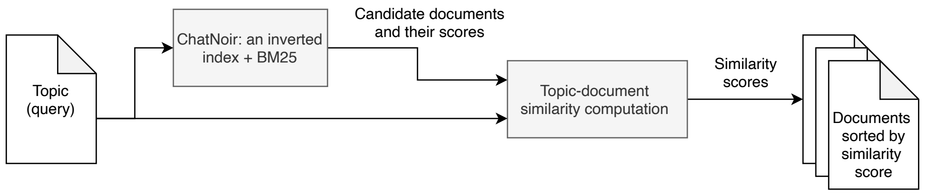
Topic (query)
ChatNoir: an inverted index + BM25
Candidate documents and their scores
Topic-document similarity computation
Similarity scores
Documents sorted by similarity score

Figure 2. Overview of our methods: candidate documents are obtained using a traditional inverted index-based IR system and then re-ranked on the basis of topic-document similarity measures specific to each of the proposed approaches.

This section contains short descriptions of approaches for the computation of the score of one document in the search engine’s response. All of the approaches described

[3] https://www.chatnoir.eu/doc
[4] https://lemurproject.org/clueweb12

below are run on TIRA system [14]. The computation of scores for the entire set of responses by a certain method is schematically shown in Figure 2. The ranking process is completed by sorting documents by these values. Ultimately, each of the presented methods computes a similarity score s_{ij} between a topic t_i and a candidate document d_j from a candidate set:

$$s_{ij} = sim(t_i, d_j). \tag{1}$$

The goal of every presented method is to compute for each topic t_i a vector of scores $\mathbf{s}_i = (s_{i1}, ..., s_{iN_i})$, where N_i is the number of candidate documents for the topic t_i.

Overall, we submitted six various solutions to test topic titles. Since no training data were provided, it is not allowed to evaluate the performance of the suggested strategies during the development. Below, we describe all proposed approaches in detail.

3.1 Baseline based on an inverted index

In our experiments, we utilize ChatNoir system [2] as a candidate documents extractor, which was provided (as a baseline) by organizers. ChatNoir is an Elasticsearch-based[5] engine providing access to nearly 3 billion web pages from ClueWeb and Common Crawl corpora. Query processing shared across several search node allows reaching response time compared to the commercial system. Text relevance estimation is based on custom BM25 scoring function,[6] which ranks the set of texts depends on the query's tokens existing in each response documents.

The defined search system in response to question in the topic title returns documents, its titles and scores. We take scores given by it and create a document ranking based on it, so, similarity score is

$$s_{ij} = cn_{ij}, \tag{2}$$

where cn_{ij} - scores provided by ChatNoir for i-th title to j-th responded document.

It should be noted that the system issue may contain similar documents. We look through the response and remove the document with duplicated titles. We also clean documents' bodies from HTML tags and markups.

3.2 Language model LSTM ULMFiT

The simplest way to estimate the relevance of documents is by mapping the query and response in the same vector space. The relevance is defined as the cosine similarity between retrieved objects.

We assume that the hidden state in the recurrent network implicitly contains information about all processed sequences. Providing topic title and a response document's body to LSTM as an input gives in the hidden state of the last step their compressed representations.

[5] https://www.elastic.co

[6] https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html

The modification of the hidden state at each step depends on the parameters of the model. We employ the weights from pre-trained Universal Language Model Fine-tuning (ULMFiT) [10]. A state-of-the-art language model AWD-LSTM [11] is tuned on Wikitext-103 [12], which collects 28,595 Wikipedia articles and 103 million words. The model class consists of 3 LSTM layers, encoder-decoder, dropouts, and linear layer. We use the class definition from here[7], extract only LSTM layers, and apply them to texts.

We pass the query and the document body through these layers. The input tokens are transferred into vectors using bert-as-service library[8].

Similarity score for this method is computed as following:

$$s_{ij} = cos(h_i, h_j), \tag{3}$$

where h_i is the hidden state of LSTM that was fed with the i-th topic's title and h_j is the hidden state for j-th response's body.

3.3 Attention of a Transformer-based language model

The Transformer is a neural-network encoder-decoder model, being used as an alternative to recurrent neural networks, such as LSTM, e.g., the ULMFiT model described in the previous section. The key innovation of the Transformer is the attention mechanism, which at each step calculates the importance of each word from the input sequence.

Information from pre-trained attention layers can be used to analyze the closeness of the query and the response. A Transformer can deal with a pair of input sequences separated by a special character. The attention layer returns the mutual weights of every word of this pair. Since we are interested in the relation of the topic and retrieved document, the input pair will be composed of them. The entrance is **"[CLS]" + query + "[SEP]" + response document's body + "[SEP]"**, where "[CLS]" and "[SEP]" are special symbols being used when processing the sentences through Transformer.

The appropriate Transformers' head selecting. The attention layer in the standard Transformer provides 12 outputs, named heads. Each of head describes its own, not predefined meaning. For every token in an encoded sequence, one head gives weights for all input tokens. If we encode the input to itself, we get a matrix of adjacent weights for each input word.

Using the obtained matrix we can build a map of attention. On Figure 3 these structures are shown for input *"[CLS]" + Which is better, a laptop or a desktop? + "[SEP]" + Laptop is preferable than desktop because it is more portable. + "[SEP]"*. The bright vertical stripe corresponds to the separation token and should be excluded from consideration. For interconnection estimation of words from different sentences, only the upper left and lower right corners of the map should be taken into account - the "non-diagonal" right upper and lower left parts describe the response of the one sentence in input pair to itself.

[7] https://github.com/fastai/fastai
[8] https://github.com/hanxiao/bert-as-service

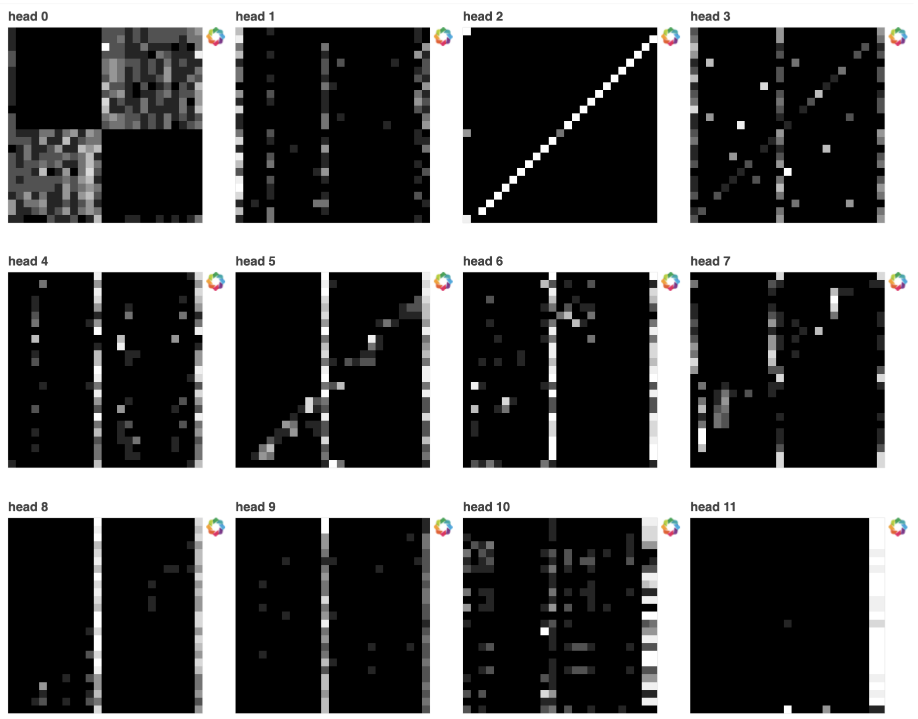

Figure 3. Maps for 12 attention heads for input "[CLS]" + Which is better, a laptop or a desktop? + "[SEP]" + Laptop is preferable than desktop because it is more portable. + "[SEP]".

To use the Transformer efficiently, we need to select those outputs that provide information relevant to response ranking. As it can be observed in Figure 4, the third head determines similar words in sequence's pair, so we take it for scoring.

To select other suitable heads, we design a sandbox experiment. We take a query "Which is better, a laptop or a desktop?" and make a set of 4 documents consisting of 1 sentence. Two of these documents are retrieved from top Google sites to query determined above and are marked as relevant. The other two are taken from "The Hunting of the Snark" by Lewis Carroll, and they are considered as unreasonable. The query and the obtained sentences are in Table 1. The idea of the experiment is to process paired input to the Transformer attention layer and observe at which outputs the value of the sum for the relevant and irrelevant documents differs the most.

We apply the Transformer to query merged with one of four sentences. For each of 12 transformer's heads, we count the sum of weights from the upper left and lower right. The most significant variation appears in 4, 10, 11 heads (Table 2). These heads are taken into consideration when a similarity score creates.

The counting score of similarity using attention layer. The response scoring consists of two steps: first, we concatenate query with enumerated document and process it by

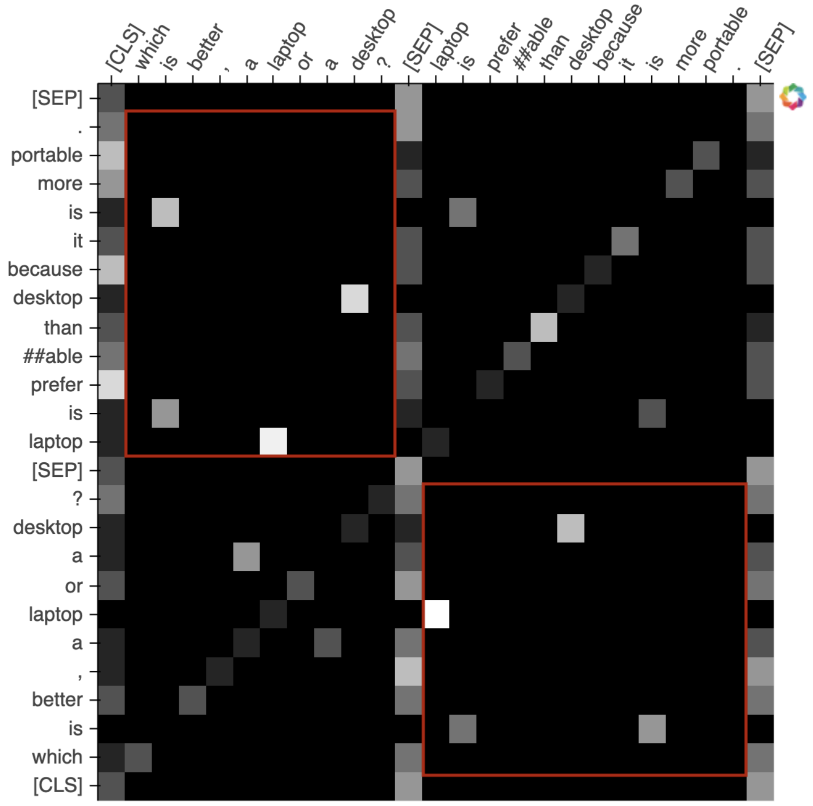

Figure 4. The third attention head for input "[CLS]" + Which is better, a laptop or a desktop? + "[SEP]" + Laptop is preferable than desktop because it is more portable. + "[SEP]". Biggest weights correspond to similar words. Areas highlighted in red describe the response of tokens from one sentence to another and are taken into account in calculating the score, corresponding to two respective sums in Equation 4.

Transformer attention layers; second, we count the sum of the appropriate parts of maps for 3, 4, 10, 11 heads. Thereby, a similarity score for i-th topic title and j-th retrieved document is calculated as

$$s_{ij} = \sum_{h=3,4,10,11} \left[\sum_{l=1}^{Q-1} \sum_{m=Q+1}^{Q+R-1} w_{lm} + \sum_{l=Q+1}^{Q+R-1} \sum_{m=1}^{Q-1} w_{lm} \right], \tag{4}$$

where Q - the length of query, which is i-th topic title, R - the length of j-th document body. W_{lm} is an attention weight for l-th to m-th token in input, ranges of indices l and m describes the proper part of the attention map. The accounted attention heads are enumerated by h. The idea of this approach for the third layer is illustrated in Figure 4. Namely, the zones highlighted in red color zones correspond to similarity of words from query to these from a candidate documents. The other parts represent self-similarity of query and document and thus have somewhat trivial sparsity pattern.

Query	**Which is better, a laptop or a desktop?**
Document 1	Laptop is preferable than desktop because it is more portable.
Document 2	If you need portability, the laptop is the best option than desktop.
Document 3	The crew was complete: it included boots, a maker of bonnets and hoods.
Document 4	Just the pace for snark, the Bellman cried.

Table 1. Possible user request and similar in meaning and distant responses for sandbox experiment, where we selecting appropriate transformer's heads.

Input	**h0**	**h1**	**h2**	**h3**	**h4**	**h5**	**h6**	**h7**	**h8**	**h9**	**h10**	**h11**
Query + Document1	1.62	3.70	0.001	5.54	**7.79**	2.04	4.58	1.38	3.46	6.01	**6.05**	**4.38**
Query + Document2	1.70	4.27	0.001	5.08	**7.53**	1.93	4.69	1.25	3.62	6.37	**6.81**	**5.55**
Query + Document3	1.80	4.20	0.001	4.87	**4.97**	1.72	3.34	1.35	3.09	4.97	**3.73**	**1.11**
Query + Document4	1.67	3.49	0.001	2.83	**4.33**	1.90	3.30	1.40	3.42	4.47	**3.92**	**3.01**

Table 2. Sums of the attention heads' outputs for relevant and unrelated responses. In every cell, there is a value counted over upper left and lower right corners on the attention map. Bold columns have a high variation on close and random answers, which means that they are sensitive to the proximity.

3.4 Bidirectional encoder representations from Transformer (BERT)

The architecture of Bidirectional Encoder Representations from Transformer (BERT) [9] is based on Transformer and is fine-tuned on specific masked language tasks. The result is a bidirectional language model that can give distributed representations of words taking into account contextual information.

We employ bert-as-service library to provide word embedding from BERT. This library uses pre-trained weights from a large uncased model with 340M parameters and encodes every word in query and document title in ChatNoir's response. Vectors corresponding to query and title are averages between all word embedding in defined sequences. The similarity score between query and title is described as

$$s_{ij} = cos(e_{query}, e_{title}), \tag{5}$$

where e_{query} - average between embedded query's tokens, e_{title} - average between embedded tokens in responded document title.

3.5 Comparative feature extraction

The scores by the approaches described above estimate the relevance of the topic and response as the closeness of the texts; in other words, show how possible and appropriate the given response is. The closeness is calculated in the context of well-known models (BERT, ULMFiT), trained on a huge amount of texts of a natural language. Such a method allows us to select documents that are similar in meaning but do not evaluate the quality of the comparison explicitly.

In order to evaluate the document as an argumentation, we use the combination of one of the some recently used methods and the approach giving information about the document's argumentativeness. The resultant similarity score is a multiplication of the score provided by the chosen method and additive term r. This term is counted for one document and represents a composition of features relied on the density of comparative sentences and features derived from the number of comparative parameters existed in the text. Initially, r is equal to 1.

The comparative degree of the document depends on the number of existent comparative sentences. To detect comparative sentences, we use the method described in [13]. It encodes the sentence by the InferSent embedder [8], then applies gradient boosted decision trees (XGBoost) [6] classifier to the resulting features. The XGBoost model is pre-trained on multi-domain Comparative Sentences Corpus 2019, formed by the 7,199 sentences. It determines the probability that considered sentences are being regular or comparative. If the comparative probability is greater than 0.2, the counter of comparative sentences is incremented.

After using the classifier of comparatives, r increases by the number of revealed sentences n:

$$r = r + n, \tag{6}$$

To precise if the document collates exactly to what the user wants to, we formalize the comparative parameters. We determine two comparison objects, predicates (comparison conditions, for example, "cheaper"), and comparison features — aspects ("for children", "for deep learning"). Tagging these parameters in a given sentence leads to the sequence labeling problem. State-of-the-art solutions provide low performance on comparative cases, and we created and trained our own sequence-labeling module to achieve acceptable quality.

Our model consists of a single layer LSTM with 200 hidden units from [7]. To the input of the recurrent network, we enter the BERT embedding of words. We train BERT and LSTM parts of the model together with a learning rate 0.00001 and 0.01, respectively. As a target, we use a custom dataset structurally similar to that by Arora [1][9] composed of 3,967 labeled comparative sentences from the different domains.

To count comparative parameters part of the additive term, first, we process the query by sequence labeling model described above. The model extracts objects, aspects, and predicates and formalize the user's answer. Then we combine the document's title and body, and by a simple search in text, try to detect extracted parameters in it.

The additional term changes according to the following law:

[9] https://github.com/uhh-lt/comparely

$$r = \begin{cases} r * 1.2 & \text{if we find one object} \\ r * 1.5 & \text{if we find two objects} \end{cases} \tag{7}$$

Appearance in the text one of the predicates or aspects in case of objects' existence additionally adds 1 to r:

$$r = r + l, \tag{8}$$

where l - number of predicates or aspects founded in the document.

3.6 Combination of Baseline, number of comparative sentences and comparative structure extraction

For every document in the response, we count additive term and multiply it with the engine's score. The resulting similarity score is

$$s_{ij} = cn_{ij} * r_{ij}, \tag{9}$$

where cn_{ij} is a score issued by ChatNoir system, r_{ij} - additive term for i-th title and j-th document, calculated as described above. For making answer, we rank documents by the resulting values.

3.7 Combination of ULMFiT, number of comparative sentences and comparative structure extraction

In this method, we do the same as in the previous section, with the only difference that scores counted by method from the section 3.2 are used as the basic value. We also compute an additive term r_{ij} for i-th title and j-th document and resulting scoring is the following:

$$s_{ij} = ulm_{ij} * r_{ij}, \tag{10}$$

where ulm_{ij} is a score by an approach based on ULMFiT.

4 Results

4.1 Evaluation

Each retrieved document from each of the seven approaches tested by our team was manually evaluated on the scale 0-1-2, where 0 means no relevance, "1" means the document contains relevant information, e.g., characteristics of one of the object, and "2" means very relevant i.e., the document directly compares the objects mentioned in the topic in the required context.

In addition to assessing the relevance, for every response, we estimate pieces of evidence provided in the document by support retrieval model [5]. Based on these judgments, the official NDCG score was computed for each submission. The results are discussed in the following section. The correspondence of the names of the methods described above and experiment run tags are in the first and the second columns of the Table 3.

	Method	Running tag	NDCG@5
§ 3.1	Baseline based on an inverted index	MyBaselineFilterResponse	**0.564**
§ 3.6	Combination of Baseline and comparative features	Baseline_CAM_OBJ	0.553
§ 3.7	Combination of ULMFiT and comparative features	ULMFIT_LSTM_CAM_OBJ	0.464
§ 3.4	Bidirectional Encoder Representations from Transformer (BERT)	myBertSimilarity	0.404
§ 3.3	Attention of a Transformer-based language model	MethodAttentionFilterResponse	0.223
§ 3.2	Language model LSTM ULMFiT	ULMFIT_LSTM	0.200

Table 3. Results of ranking quality's measure for described methods.

4.2 Discussion

The top-5 discounted cumulative gain(DCG@5) scores for the proposed approaches are in the Table 3.

The Table 3 shows that approaches using only pre-trained language models give the smallest scores. It can be explained by the fact that the information stored in the SOTA linguistic model is sufficient to estimate the appropriateness of the text but not enough to assess how complete, persuasively, and supportive the document is. As in many other tasks, the Attention-based model has a better performance than ULMFit - 0.223 against 0.200. This is due to the fact that the attention mechanism allows us to consider the meaningful context that is located at a distance from the current word, which makes the model more expressiveness. BERT-based model is a bidirectional expansion of the attention layer. Therefore, its application increases the performance to 0.405.

Overall, a combination of the approaches with comparative information shows better performance than the same method without comparative terms. Thus, consideration of comparative structures and sense improves the results for ULMFit from 0.200 to 0.464.

The best quality is provided by the baseline model being cleaned from document duplicates. Its scoring function is based on the BM25 ranking formula but uses a more efficient way of calculating term frequencies [16]. It provides the ability to consider information from all parts of the document - title and body, which gives superiority over methods that process only the title or only the document's body. It should be noticed that the baseline gives NDCG@5 0.565, baseline with CAM, and object extraction - 0.554. One reason for decreasing quality when complementary information is added is choosing the weight with which we consider the CAM information and number of comparative structures.

The main take-aways are as follows. First, the methods for re-ranking of the candidate documents which do not rely on the original baseline score, but instead completely replace it with similarity scores based on the language models do not yield superior results to the baseline; therefore, the original scores shall be used. Among all such, completely baseline-free methods BERT-based similarity yielded the best results. Second, a combination of the custom features based on the density of comparative structures in text combined with the baseline yield better results. Since no training data was provided in this version of the shared task, it was not obvious to test various combinations of the tested features, but given such supervised training data, a promising way to further improve the results is to combine various signals using a supervised machine learning model.

5 Conclusion

In this paper, we present our solution to the Argument retrieval shared task. Our main innovations are (i) the use of large pre-trained language models, (ii) the use of features based on natural language understanding of comparative sentences, and (iii) the use of features based on the density of comparative sentences. It should be noted that modern linguistic models meet the response relevance quite well, but to assess the comparability and argumentation of the answer, we need to add external features.

Overall, according to the experimental results, the baseline information retrieval model proved to be a hard baseline. In fact, among all 11 evaluated runs in the shared task only one outperformed the baseline by more than 0.5% which is a substantial difference.[10] The results suggest that considering the score taking into account claim support and evidence existing, models based on SOTA language models do not work as well as the models which combine comparative structure and comparative sentiment in sentences. We conclude that in future work, more combinations of methods based on a combination of baseline IR models with comparative features shall be investigated.

References

1. J. Arora, S. Agrawal, P. Goyal, and S. Pathak. Extracting entities of interest from comparative product reviews. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1975–1978, New York, NY, USA, 2017. ACM.
2. J. Bevendorff, B. Stein, M. Hagen, and M. Potthast. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In L. Azzopardi, A. Hanbury, G. Pasi, and B. Piwowarski, editors, *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, Mar. 2018. Springer.
3. A. Bondarenko, P. Braslavski, M. Völske, R. Aly, M. Fröbe, A. Panchenko, C. Biemann, B. Stein, and M. Hagen. Comparative web search questions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 52–60, 2020.
4. A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen. Overview of Touché 2020: Argument Retrieval. In *Working Notes Papers of the CLEF 2020 Evaluation Labs*, Sept. 2020.
5. L. Braunstain, O. Kurland, D. Carmel, I. Szpektor, and A. Shtok. Supporting human answers for advice-seeking questions in cqa sites. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe,

[10] https://events.webis.de/touche-20/shared-task-2.html#results

F. Silvestri, G. M. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information Retrieval*, pages 129–141, Cham, 2016. Springer International Publishing.

6. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. J. Smola, C. Aggarwal, D. Shen, and R. Rastogi, editors, *KDD*, pages 785–794. ACM, 2016.
7. A. Chernodub, O. Oliynyk, P. Heidenreich, A. Bondarenko, M. Hagen, C. Biemann, and A. Panchenko. TARGER: Neural argument mining at your fingertips. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 195–200, Florence, Italy, July 2019. Association for Computational Linguistics.
8. A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.
9. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
10. J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.
11. S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. In *ICLR (Poster)*. OpenReview.net, 2018.
12. S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *ICLR (Poster)*. OpenReview.net, 2017.
13. A. Panchenko, A. Bondarenko, M. Franzek, M. Hagen, and C. Biemann. Categorizing comparative sentences. In *Proceedings of the 6th Workshop on Argument Mining*, pages 136–145, Florence, Italy, Aug. 2019. Association for Computational Linguistics.
14. M. Potthast, T. Gollub, M. Wiegmann, and B. Stein. TIRA Integrated Research Architecture. In N. Ferro and C. Peters, editors, *Information Retrieval Evaluation in a Changing World*, The Information Retrieval Series. Springer, Sept. 2019.
15. M. Potthast, M. Hagen, B. Stein, J. Graßegger, M. Michel, M. Tippmann, and C. Welsch. Chatnoir: a search engine for the clueweb09 corpus. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1004–1004, 2012.
16. S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. pages 42–49, 01 2004.
17. M. Schildwächter, A. Bondarenko, J. Zenker, M. Hagen, C. Biemann, and A. Panchenko. Answering comparative questions: Better than ten-blue-links? In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 361–365, 2019.