

# Touché Task 2: Comparative Argument Retrieval

## A document-based search engine for answering comparative questions

Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

Daniel Helmrich, Denis Streitmatter, Fionn Fuchs and Maximilian Heykeroth

University Leipzig, Germany

### Abstract

While the retrieval of simple facts works very well with modern search engines, the quality of results for comparative questions is rather mediocre. With the goal of developing a retrieval algorithm for finding documents containing arguments that help to answer those questions, we tried to evaluate many different approaches to both query expansion and document ranking. Our aim was to build a modular and highly configurable system that provides the necessary tools to help with evaluation-driven experimentation. The results show that we found approaches that were able to outperform the baseline. Especially simple approaches like query term counting have proven to be promising. One of the simplest approaches combining query term count and ChatNoir scores improves the NDCG by around 5% w.r.t. the baseline. However, due to the limitations of the provided rankings and the limited time frame further work remains.

### Keywords

information retrieval, comparative argument retrieval, comparative questions

## 1. Introduction

Since the development of the World Wide Web in 1989, it has evolved to the main source of information for a large part of the world's population. Nearly every arbitrary fact can be searched for using search engines like DuckDuckGo, Bing, or Google, be it *"How long is the Nile?"* or *"How much nutmeg is dangerous?"*. The retrieval of such facts works pretty well with today's search engines, but the need for information goes far beyond that. The comparison of the wide range of options for such things as products, brands, or lifestyle choices is just as crucial as the retrieval of simple facts. Of course, there are systems for comparison available, like [diffen.com](http://diffen.com); but while they provide an appropriate comparison for certain topics, their range is quite limited. They are not able to answer questions outside of the few domains they offer (e.g. insurance, food, or technology). For users in need of more detailed questions, the remaining option is to consult regular search engines. On those, however, comparative queries like *"Which is healthier: Green or Black tea?"* or *"Which beverage has more calories per glass: beer or cider?"* do not perform nearly as well, leaving much room for improvement. In an attempt of

---


CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

✉ [dh86hogi@studserv.uni-leipzig.de](mailto:dh86hogi@studserv.uni-leipzig.de) (D. Helmrich); [streitmatter@informatik.uni-leipzig.de](mailto:streitmatter@informatik.uni-leipzig.de) (D. Streitmatter);

[ff87bake@studserv.uni-leipzig.de](mailto:ff87bake@studserv.uni-leipzig.de) (F. Fuchs); [mh40qyqu@studserv.uni-leipzig.de](mailto:mh40qyqu@studserv.uni-leipzig.de) (M. Heykeroth)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

tackling this issue, in the following paper we present a modular evaluation system<sup>1</sup> dedicated to assessing various approaches for comparative argument retrieval.

The basis for retrieving documents is the ChatNoir search engine [1]. A synopsis of other approaches tackling this issue can be found in the overview paper [2] of the task. The source code was also submitted to TIRA [3], an evaluation platform for information retrieval tasks, to make further automatic evaluation possible.

## 2. Related Work

Since the same task was already assigned during CLEF 2020, various groups already tried to tackle the issue. The conclusive paper by Bondarenko et al. [4] gives an overview of the implemented systems and ideas. Five groups participated in the task, implementing 11 approaches. This gives a great starting point for this task since it notes effective and not-so-successful approaches to follow/ignore. The students' ideas vary greatly, from using different query expansion techniques to implementing various (re-)ranking algorithms to push the ChatNoir results with good arguments to the top.

The approach standing out here is the one by Abye et al. [5] yielding the highest NDCG of all the different runs. In their approach, they utilize synonyms and antonyms for the query expansions and extract important comparison features (e.g "better") from the query. To retrieve these synonyms they use the WordNet lexical database and a Word2Vec trained on the English Wikipedia. Four different queries based on these synonyms and antonyms are created for their query expansion approach. After this, they rerank the received documents by using different scores, like the PageRank or the number of comparative sentences in the results.

An existing system to answer comparative questions on the general domain is CAM (short for comparative argument machine) [6]. The interface allows the input of the two targeted objects (e.g python and java) and arbitrarily many further comparison aspects (e.g faster). The results are sentences from the Common Crawl<sup>2</sup> retrieved via Elasticsearch [7] and then reduced to comparative sentences (either by certain keywords or a machine learning approach) and ranked. The system was evaluated by persons using either CAM or a keywords-based search, measuring the speed and confidence of the answer. The results show that the studies' participants are faster and more confident using the CAM system. The difference to our task is here that this approach finds *argument sentences* in documents while our task is finding the best *argumentative documents*.

A tool for detecting arguments in the text is TARGER by Chernodub et al. [8]. It is open-source and publicly available, making argument mining accessible for everyone. It provides an algorithm for tagging arguments in text as well as a retrieval functionality for finding arguments for a given topic. The neural tagger lets the user choose between different models of datasets and word embedding techniques. It returns a JSON file with a list of words each tagged as claim, premise, or not part of an argument. Additionally, the confidence for each rating is returned.

In an attempt to improve the query expansion performance of argument retrieval systems,

---

<sup>1</sup>The source code can be found at <https://git.informatik.uni-leipzig.de/depressed-spiders/comparative-argument-retrieval>

<sup>2</sup><http://commoncrawl.org/>

[9] investigated three approaches based on transformers, including query expansion with transformer-based models [10] like GPT-2, BERT, and Google’s Bert-like universal sentence encoder (USE). The GPT-2 based query expansion is the most successful one, increasing the retrieval performance over the baseline by 6.878%.

By crawling multiple debate portals like Debatewise, Debatepedia, Debate.org, etc. in 2019, [11] created a corpus called “args.me corpus” that consists of ca. 380.000 arguments from all kind of debates. These arguments were extracted using specific heuristics.

### 3. Methodological Approach

#### 3.1. Overview

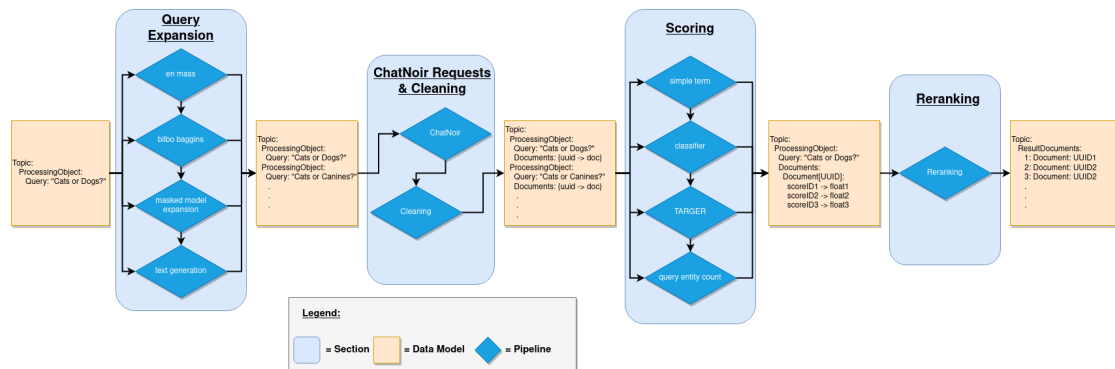


Figure 1: A schematic overview over the process

To test multiple approaches we implemented a modular system, consisting mainly of a data model and a pipeline changing it. Figure 1 shows the (schematic) change of the data model (light orange squares) and the processing order of the pipeline (blue diamond-shapes), structured into different sections (light blue). The data model is initialized with the query and the first section, **Query Expansion**, will extend this by adding further, related queries to find relevant documents. The data model, enriched with multiple queries, is then sent to the ChatNoir [1] endpoint for retrieving identifiers of relevant documents, as well as their corresponding full HTML representation in the **ChatNoir Requests** section. Additionally, the document is cleaned from HTML tags. Now every query (both original and expanded) has a list of result documents. In the next section, **Scoring**, each of these documents is augmented with arbitrarily many scores, rewarding documents relevant for the comparison with higher scores. In the last step, **Reranking**, all of these scores are collected and merged into one (allowing the weighting of each score) and ranking the list of result documents accordingly.

#### 3.2. Query Expansion

To create a broad mass of queries, we utilize multiple query expansion approaches and combine them differently to maximize the recall value. We include the original query in all requests.

Our first query expansion method is called *en mass*. This approach iteratively replaces one of the entities or features in the query with their most similar word. We retrieve similar words with a Word2Vec model that is provided by the Python library “Gensim” [12]. This model was trained on a Wikipedia corpus from 2014 and a newswire text data corpus called “Gigaword” [13]. Entities are the nouns or brand names that the user wants to compare or retrieve more information about. Features are the words that are used to compare the entities under a given aspect. To extract these features and entities we use the NLP library spaCy [14]. The *en mass* approach returns as many queries as there are entities and features in the query.

The second approach is called *Bilbo Baggins*. This is the same query expansion method described by Abye et al. from the Touché task in 2020 [5].

Our third query expansion utilizes a masked language model and is called *masked model expansion*. Similar to the approach by Akiki and Potthast [9], we use a masked language model to replace masked words with a similar (context fitting) word. We use the query and replace all entities and features one by one with a mask token. This method generates as much queries as there are entities and features in the query. The used masked model is the “roberta-base” model [15]. We chose this model since RoBERTa seems like an improved version of BERT [15] which was used by Akiki and Potthast in [9].

We also tried to fine-tune a small BERT model on the args.me corpus [16]. However, this fine-tuned model produced significantly worse results and this approach was therefore discarded due to lack of time.

Our last approach is the *text generation* approach, which utilizes a text generation model to provide more words for a given query. The idea behind this approach is to generate text from the query and utilize the nouns from the generated text as new queries. For this query expansion approach we work with the GPT-2 model [17]. The text is generated with a top-p sampling method, which is one of the recommended sampling methods to generate fluent text with GPT-2 models [18]. To generate the text we use the original query and append the feature word from the original query after the query to start a new sentence. This method can be seen in Table 1, here the feature word “better” is directly appended after the query. All words that are written in italics in the “Generated text” row, are used to generate text with the GPT-2 model. The idea behind this change is to restrict the generated text. Sentences shall generate text that better fits the comparative queries by including the aspect (feature) of the comparative query in the text generation.

Original query	Which is better, Canon or Nikon?
Generated text	Which is <i>better</i> , Canon or Nikon? <i>Better</i> pictures are taken by a good camera lens. In comparison [...]
Expanded query	canon nikon pictures camera lens comparison

**Table 1**  
Example: text generation expansion

All query expansion methods that request similar words for a given word (*bilbo baggins*, *masked model expansion* and *en mass*) use the two most similar words for a given word (not just the most similar word). This means that the number of created queries for *masked model expansion* and *en mass* doubles. Similarly, the *text generation expansion* generates two texts and

therefore expands the original query into two queries.

All expanded queries are cleaned after the query expansion process by removing words within the queries that are too similar to each other.

### **3.3. ChatNoir & Cleaning**

All queries that were created during the query expansion are subsequently forwarded to the ChatNoir search engine. Mostly, 100 documents are requested, but as shown later, we also experimented with different amounts for this value. The queries to ChatNoir are executed using the ClueWeb12 dataset, which is a large dataset of English web pages collected in 2012 [19]. Almost all documents returned by ChatNoir contain a query-specific score, a page rank, and a spam rank. Furthermore, all returned documents are uniquely identified with a UUID (universally unique identifier).

After our application receives the documents from the ChatNoir endpoint, the documents are cleaned by removing HTML tags. No further content extraction or cleaning is done.

### **3.4. Scoring**

The goal of the scoring steps in our pipeline architecture is to assign scores to documents respecting their relevance to the query and their argumentative quality. In this section, each approach that we have implemented is described. The resulting scores are used in the "reranking" pipeline step described in 3.1 to sort the documents into their final order of relevance and quality.

#### **3.4.1. Simple Term Count**

One of the simplest ranking approaches is to assume that documents containing certain terms might be more relevant. In the context of this work, the main idea is that there might be certain words that are used often in argumentative texts and rarely in not argumentative texts. Thereby simply counting occurrences of these words could help to find relevant documents. Examples for such words are "evidence", "that shows", "versus" and "in comparison to".

The "Simple Term Count" pipeline step accepts a list of terms as a txt file. For each document, the pipeline step receives it checks for every term in the list if it occurs in the cleaned text of the document. The score assigned to the document is the number of terms from the list that occur in the document's cleaned text.

#### **3.4.2. Query Term Count**

Another implemented manual feature-engineering approach is based on the counting of relevant query terms in each document. The main idea is that the relevancy of a document increases with a higher number of words that also appear in the query. It can be expected that the user is especially interested in comparisons that include all the terms he searched for. The score, therefore, rewards documents that contain all of the relevant query terms in an equally distributed manner. The pipeline step described in the following tries to calculate a document score that reflects these thoughts.

Its first step is the identification of terms that are deemed *relevant* for the query. For this, a similar method as for the query expansion is employed, which extracts query terms based on their part-of-speech tags. The terms are always extracted from the original query, even if sub-queries were generated by the query expansion step.

In the following,  $T$  denotes the set of relevant query terms. For each retrieved document, all  $t \in T$  are then counted, respectively yielding the number of occurrences  $n_t$ . Based on this, the score is calculated as follows:

$$x_{\text{doc}} = \begin{cases} 0, & \exists t \in T | n_t = 0 \\ \sum_{t \in T} 1 - \frac{1}{b \cdot n_t + 1} & \text{else} \end{cases}$$

The score attains zero if not every term occurs at least once in the document. Otherwise, the shown sum is calculated, whose summands converge to 1 for increasing  $n_t$ . This has the effect of not favoring documents only due to the excessive existence of some terms and it makes the score resistant to the length of a document. The score is higher for documents where the query term occurrences are uniformly distributed than for such where only one or a few terms occur very often.

The factor  $b$  (with  $b \geq 0$ ) is another parameter that determines how big the influence of increasing values of  $n_t$  is: Greater  $b$  yields a curve that is steeper for small  $n_t$ , and will more early be close to 1. A smaller  $b$  on the other hand will make the score not grow as fast for increasing  $n_t$ .

### 3.4.3. Classifier

To measure the argumentative quality of texts we trained a text classifier and built a pipeline step for assigning scores using Tensorflow text classifiers. The classifier is trained on a combined dataset which contains documents from the "askreddit" Q&A forum on the social media platform Reddit and additionally documents from the IBM Debater dataset[20].

On the "askreddit" forum, posts can be tagged with the flair *serious*. These posts are strictly moderated and only serious comments are allowed. Jokes, puns, and inappropriate comments are supposed to be deleted. The goal of the classifier during the training phase is to distinguish between posts from two pools. The first pool contains comments from *normal* askreddit posts. The second pool contains comments from *serious* askreddit posts and additional arguments from the IBM Debater dataset. This pool should be recognized as arguments of high quality.

The comments and their corresponding question posts are mapped to vectors by calculating the mean of their content's word2vec values. For examining the validness of these resulting vectors, we compared the serious and normal comments with their respective post. By reducing the vectors with PCA, we could verify that the vectors for *serious* comments are closer to their corresponding post than the comments from the *normal* category. This supports the hypothesis that *serious* comments are thematically closer to the question they are answering. Therefore, a classification based on word vectors can be a valid approach. Nevertheless, due to the limited time frame, we decided to use a classifier that only takes the comment as an input disregarding the original post the comment is attached to.

The classifier we trained is a Tensorflow model, utilizing text vectorization and one-dimensional

convolutional layers. It is trained with the binary-cross-entropy loss function. The dataset is split into training and validation data at an 80%/20% ratio.

On the validation set, the classifier achieves an accuracy of 83%. State-of-the-art argument classifiers also achieve accuracy of roughly 80% [21]. It has to be noted that the difference in the datasets is quite substantial and due to the unique properties of the Reddit dataset, the trained classifier might work on a simpler domain than other argument classifiers.

#### 3.4.4. Targer

For another approach of measuring the "argumentativeness" of a document, the neural argument miner TARGER is used. For this every document's cleaned text is sent to the public available API<sup>3</sup>. It returns a list of words, either tagged as claim, premise, or not part of an argument with the confidence of said tag. The assumption for this approach is the more of the text is an argument (or part of one) the better. So we try to measure the argumentative density by calculating the value  $x_{\text{doc}}$ :

$$x_{\text{doc}} = \frac{\text{tags}_{\text{arg}}}{\text{tags}_{\text{all}}}$$

While  $\text{tags}_{\text{arg}}$  is the number of words tagged as an argument,  $\text{tags}_{\text{all}}$  denotes all tags (words) in the document. It is configurable what counts as an  $\text{tag}_{\text{arg}}$  (claim, premise, or both) and what minimal confidence a tag needs to be counted as an argument.

There are multiple models available at the public TARGER API. The model mainly used here<sup>4</sup> yields argument/non-argument useful ratios roughly between 10% and 80%.

### 3.5. Reranking

The reranking step of the pipeline sorts all documents by their assigned scores. Each score value is normalized before sorting. A configurable weighting of each scoring method is possible. To sort each document a final, normalized and weighted score is assigned using the following formula where  $S$  is a list of scores and  $w$  is a configured weight for the scoring method:

$$\text{score}_{\text{doc}} = \sum_{\beta \in S} \frac{\beta}{\beta_{\text{max}}} \times w$$

## 4. Evaluation & Results

### 4.1. Evaluation Method

To gain valuable insights into the characteristics of our implemented pipeline steps, we created a baseline configuration to compare them with. It only uses ChatNoir and does not employ any modifications, neither on the original query nor on the ranking that is returned by it. This also corresponds with the baseline utilized in [4], whose results we could thereby reproduce.

Following the idea of an evaluation-driven development, our system allows us to test and fine-tune various approaches. By comparing them with the baseline, we could quickly decide

---

<sup>3</sup><https://demo.webis.de/targer-api/apidocs/>

<sup>4</sup>Essays model, fasttext embeddings. See [https://demo.webis.de/targer-api/apidocs/#/Argument%20tagging/post\\_targer\\_api\\_tag\\_essays\\_fasttext](https://demo.webis.de/targer-api/apidocs/#/Argument%20tagging/post_targer_api_tag_essays_fasttext)

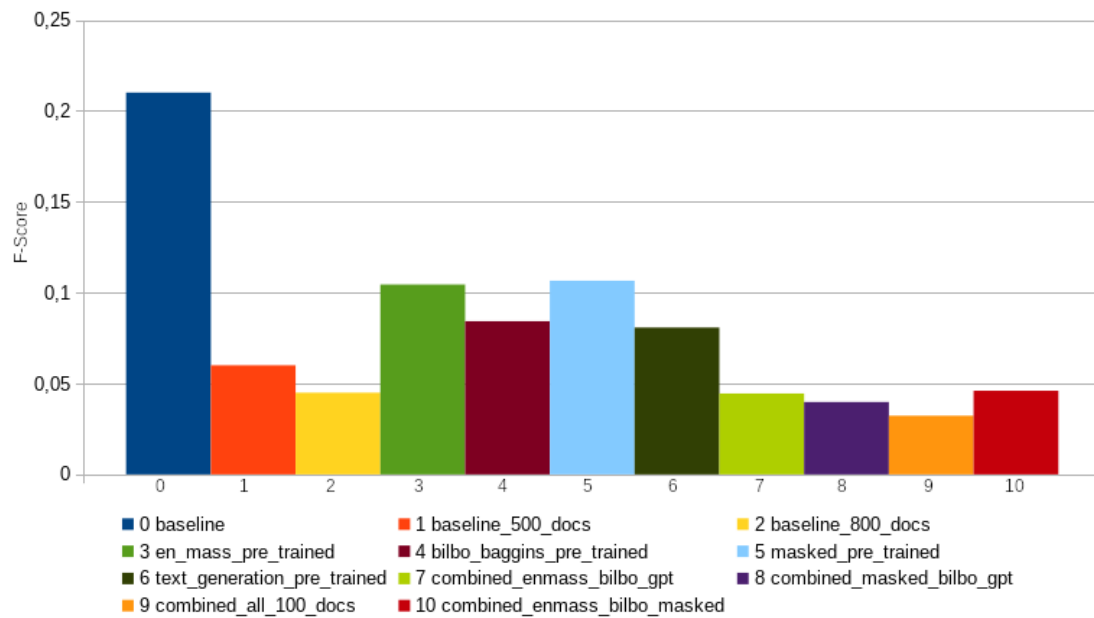
which parameters work better than others. The evaluation of every configuration was based on the topics and judged documents from the Touché Lab 2020 [4].

For determining the ideal values for the reranking weights, a simple hyperparameter search is used. While, in theory, each part of the pipeline could benefit from this kind of optimization method, the reranking step is particularly suited for it: The serialization of the internally used data model allows re-running this step multiple times, as the reranking itself runs in a significantly shorter time than the other pipeline steps. Therefore a grid search is implemented, which simply iterates over a given set of combinations for the reranking weights. For a given pipeline configuration, this allows determining weight values that maximize the NDCG@5.

## 4.2. Results

### 4.2.1. Query Expansion

An overview of our query expansion methods and their f-score values is shown in Figure 2. The f-scores were created by requesting 100 documents per query from ChatNoir.



**Figure 2:** F-score comparison of query expansion methods

The query expansion mode “0 baseline” in Figure 2 represents the f-score of the default query with 100 requested documents. Since expansion methods like *en mass* generate up to 8 queries, which result in ca. 800 documents that are retrieved, we also include “1 baseline\_500\_docs” and “2 baseline\_800\_docs”, which request 500 respectively 800 documents for the original query. This allows us to better compare our expansion methods with the baseline, since the different query expansion modes create different amounts of queries and therefore retrieve different amounts of documents. We see that with more documents, the f-score decreases. This does



not necessarily mean that the results are not relevant, but could also be an indicator that the ratio of evaluated to un-evaluated documents decreases with more documents. Therefore, we included the baseline query with more than 100 documents to better compare the f-score with our other approaches to query expansion that also retrieve more than 100 documents - since the *en mass* approach retrieves ca. 800 documents.

It is observable that all query expansion methods perform better than the baseline with 500 and 800 documents. None of these approaches is better than the baseline with 100 documents. This is probably due to the fact, that the document judgments were done over the baseline query.

To further increase the number of retrieved documents and therefore increasing the number of possibly relevant documents, we tried to combine multiple query expansion modes. The combination of all expansion methods and requesting 100 documents per query, results in the worst F-score with a score of ca. 0.03. This is why we decided to only combine three query expansion approaches. The query expansion approaches with the highest f-score are a combination of *bilbo baggins*, *en mass* and *text generation* and also a combination of *bilbo baggins*, *en mass* and *masked model expansion*. These approaches are shown in Figure 2 as approach 7 and 10. The main difference between these two approaches is the deterministic state of the query. While approach 7 is deterministic, the usage of a GPT-based model makes the generated query nondeterministic.

#### 4.2.2. Reranking

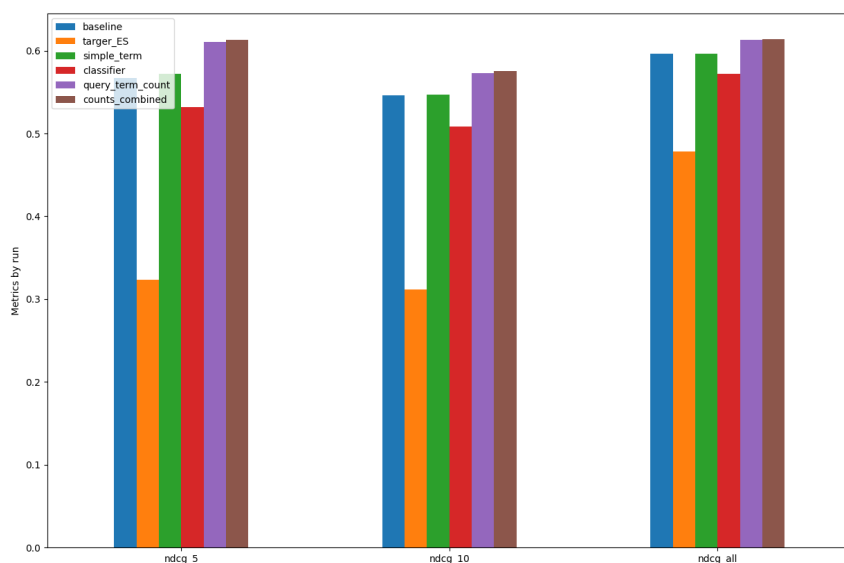
We tested the reranking approaches on the standard baseline with 100 documents (blue bar) from ChatNoir to measure which algorithm would yield the best NDCGs. In Figure 3 the comparison of different implementations is shown, namely TARGER, the Tensorflow classifier, the simple term counting, and the counting of relevant query terms.

The ES in the TARGER label stands for the used Essays model. However, even though some topics got a better ranking with the usage of TARGER, the hyper-parameter search revealed for the maximization of the NDCG that it is best set to 0, meaning it has no positive influence on the average NDCG with any weighting. Also, the classifier based on the IBM/Reddit dataset seems to be unable to improve the NDCG of the baseline retrieval. The simple term count doesn't seem to add much to the NDCG, positive or negative.

A well-performing approach is counting the relevant query terms in each document. Furthermore, it can be seen that the combination of the simple term count and the query term count seems to be slightly better than the latter alone, but in the result displayed in Figure 3 the simple term count only has a weighting of 0.05, meaning that it has nearly no influence.

#### 4.2.3. Combined Approaches

We tried bringing the best of our approaches together, using the query term counting reranking approach on the results of the query expansion, which yielded the results in Figure 4. The bars annotated with `_ignore` are evaluated with the ignore-option. This option ignores all unjudged documents when calculating the NDCG, meaning that we do not assume unjudged documents as irrelevant, which is normally the case. We did not have a lot of judgements for



**Figure 3:** Average NDCG@X of the different reranking approaches (on the baseline)

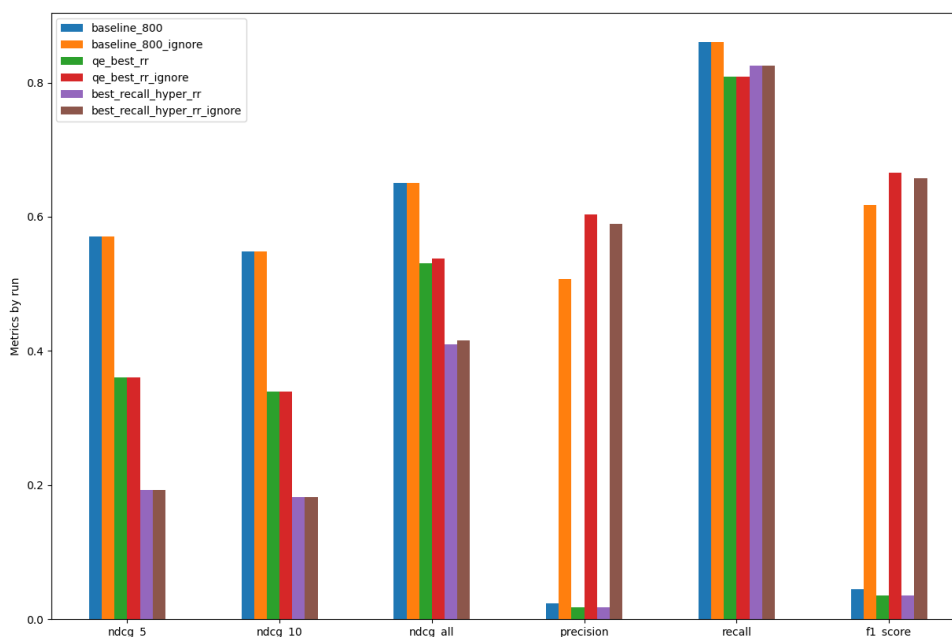
the documents, most of the documents we retrieved were unjudged. Since a lot of unjudged documents in `ndcg@5/10/..` could either mean that our reranking methods perform bad, or that we found lots of new potential relevant but unjudged documents. So we wanted to ignore all unjudged documents, to only calculate the NDCG over judged documents. This allows us to better compare the `ndcg` values of the reranking approaches among each other.

The best retrieval method combined with the query expansion (`combined_masked_bilbo_gpt`) yields no better NDCG than simply querying the baseline. Our second combination approach is utilizing the query expansion `combined_enmass_bilbo_masked`, which has the highest recall. The results of this approach are then annotated with reranking scores and the hyper-parameter search in then run on it. It revealed that the scores of TARGER and the count of simple terms do not contribute to a better NDCG (best NDCG with a weighting of 0). While the classifier does rank relevant documents higher (weight: 1.1), but not as well as the query term count (weight: 2.5).

The precision and the F-score of the approaches with the ignore-option are higher than the ones of the baseline. Since the recall of our query expansion is not as high as the `baseline_800`, this means that the query expansion retrieves more unjudged documents.

We decided to submit multiple runs to TIRA, but not all combinations of our approaches, only the one we deemed worth. The submitted runs to TIRA are:

- run\_1:** using `combined_masked_bilbo_gpt` query expansion and query term counts reranking
- run\_2:** the `counts_combined` reranking (combining both methods that count terms in the documents)
- run\_3:** using `combined_enmass_bilbo_masked` query expansion and a optimized weighting on the reranking



**Figure 4:** Average metrics of all topics with the combined query expansion and best reranking approach (qe\_best\_rr) compared to the baseline (baseline\_800) and the best recall with query expansion and a hyperparameter search on the weighting of all reranking scores (best\_recall\_hyper\_rr)

## 5. Discussion & Conclusion

In this project, we tried to evaluate many different approaches to both query expansion and ranking documents. Our aim was to build a modular and highly configurable system that provides the necessary tools to help with evaluation-driven experimentation. The results show that we found approaches that were able to outperform the baseline. Especially simple approaches like query term counting have proven to be promising.

But there are certain limitations to our work. Due to the focus on many approaches, we could not investigate them in depth. Focusing on a small number of approaches might help in improving our research insights further. The quality of our evaluation is also limited by the provided judgments we compared to. They only contain a very limited amount of documents per topic.

In the future, multiple steps could be taken to further improve our research. Main content extraction could be applied to the documents and might improve the performance of the scoring pipeline steps by removing ads and irrelevant content. Our TARGER implementation can also be improved by applying a more specific algorithm than just counting words to the arguments of the text. Additionally due to the limited amount of provided queries our hyperparameter search might lead to overfitting on the domain. This might need to be mitigated in the future.

The query expansion could be improved by working more on the entity and feature extraction since some expanded queries suffer from falsely extracted entities and features. To further improve the F-score, one could experiment more with the parameters of the specific expansion

modes, like the number of returned queries for each approach.

A promising approach to reranking seems to be the counting of relevant query terms in the result document itself, improving the NDCG w.r.t the baseline by around 5% (NDCG@5). It can be assumed that the underlying formula (cf. Section 3.4.2) favors texts which focus on the comparison of two or more entities. However, it does not include any qualitative measurement of the included arguments. This leaves room for further improvement: It could be imagined to combine the resulting score with another score rewarding good arguments in the text.

Lastly, the implemented reranking approach imposes certain limitations, as it is only a linear combination of the calculated document scores. Combining them in a more complex way could improve the performance as well.

To conclude, some approaches could be found that improve the retrieval performance in comparison to the baseline. The rather simple approach of counting relevant query terms in each document, combined with ChatNoir scores seems to be promising. The provided evaluation system might be used in the future to further incorporate and assess the viability of more complex approaches.

## References

- [1] J. Bevendorff, B. Stein, M. Hagen, M. Potthast, Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl, in: L. Azzopardi, A. Hanbury, G. Pasi, B. Piwowarski (Eds.), *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2018.
- [2] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2021: Argument Retrieval, in: D. Hiemstra, M.-F. Moens, J. Mothe, R. Perego, M. Potthast, F. Sebastiani (Eds.), *Advances in Information Retrieval. 43rd European Conference on IR Research (ECIR 2021)*, volume 12036 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2021, pp. 574–582. URL: [https://link.springer.com/chapter/10.1007/978-3-030-72240-1\\_67](https://link.springer.com/chapter/10.1007/978-3-030-72240-1_67). doi:10.1007/978-3-030-72240-1\_67.
- [3] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA integrated research architecture, in: N. Ferro, C. Peters (Eds.), *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*, volume 41 of *The Information Retrieval Series*, Springer, 2019, pp. 123–160. URL: [https://doi.org/10.1007/978-3-030-22948-1\\_5](https://doi.org/10.1007/978-3-030-22948-1_5). doi:10.1007/978-3-030-22948-1\_5.
- [4] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2020: Argument Retrieval, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névél (Eds.), *Working Notes Papers of the CLEF 2020 Evaluation Labs*, volume 2696 of *CEUR Workshop Proceedings*, 2020. URL: <http://ceur-ws.org/Vol-2696/>.
- [5] T. Abye, T. Sager, A. J. Triebel, An open-domain web search engine for answering comparative questions, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névél (Eds.), *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, Thessaloniki, Greece,

- September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: [http://ceur-ws.org/Vol-2696/paper\\_130.pdf](http://ceur-ws.org/Vol-2696/paper_130.pdf).
- [6] M. Schildwächter, A. Bondarenko, J. Zenker, M. Hagen, C. Biemann, A. Panchenko, Answering comparative questions: Better than ten-blue-links?, in: L. Azzopardi, M. Halvey, I. Ruthven, H. Joho, V. Murdock, P. Qvarfordt (Eds.), *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval, CHIIR 2019, Glasgow, Scotland, UK, March 10-14, 2019*, ACM, 2019, pp. 361–365. URL: <https://doi.org/10.1145/3295750.3298916>. doi:10.1145/3295750.3298916.
- [7] C. Gormley, Z. Tong, *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*, " O'Reilly Media, Inc.", 2015.
- [8] A. N. Chernodub, O. Oliynyk, P. Heidenreich, A. Bondarenko, M. Hagen, C. Biemann, A. Panchenko, TARGER: neural argument mining at your fingertips, in: M. R. Costajussà, E. Alfonseca (Eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, Association for Computational Linguistics, 2019, pp. 195–200. URL: <https://doi.org/10.18653/v1/p19-3031>. doi:10.18653/v1/p19-3031.
- [9] C. Akiki, M. Potthast, Exploring argument retrieval with transformers, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: [http://ceur-ws.org/Vol-2696/paper\\_241.pdf](http://ceur-ws.org/Vol-2696/paper_241.pdf).
- [10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Huggingface's transformers: State-of-the-art natural language processing, *CoRR abs/1910.03771* (2019). URL: <http://arxiv.org/abs/1910.03771>. arXiv:1910.03771.
- [11] Y. Ajjour, H. Wachsmuth, J. Kiesel, M. Potthast, M. Hagen, B. Stein, args.me corpus, 2020. URL: <https://doi.org/10.5281/zenodo.4139439>. doi:10.5281/zenodo.4139439.
- [12] R. Řehůřek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, in: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, ELRA, Valletta, Malta, 2010, pp. 45–50. <http://is.muni.cz/publication/884893/en>.
- [13] R. Parker, D. Graff, J. Kong, K. Chen, K. Maeda, *English Gigaword Fifth Edition*, Philadelphia: Linguistic Data Consortium, 2011. doi:10.35111/wk4f-qt80.
- [14] M. Honnibal, I. Montani, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, 2017. To appear.
- [15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, *CoRR abs/1907.11692* (2019). URL: <http://arxiv.org/abs/1907.11692>. arXiv:1907.11692.
- [16] H. Wachsmuth, M. Potthast, K. Al-Khatib, Y. Ajjour, J. Puschmann, J. Qu, J. Dorsch, V. Morari, J. Bevendorff, B. Stein, Building an argument search engine for the web, in: *Proceedings of the 4th Workshop on Argument Mining*, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 49–59. URL: <https://www.aclweb.org/anthology/W17-5106>. doi:10.18653/v1/W17-5106.
- [17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners (2019).

- [18] P. von Platen, How to generate text: using different decoding methods for language generation with transformers, 2020. URL: <https://huggingface.co/blog/how-to-generate>, accessed: 26.02.2021.
- [19] J. Callan, M. Hoy, C. Yoo, L. Zhao, Clueweb12 web dataset, 2012.
- [20] IBM, Ibm debater dataset, 2014. URL: [https://www.research.ibm.com/haifa/dept/vst/debating\\_data.shtml](https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml).
- [21] A. Toledo, S. Gretz, E. Cohen-Karlik, R. Friedman, E. Venezian, D. Lahav, M. Jacovi, R. Aharonov, N. Slonim, Automatic argument quality assessment – new datasets and methods, 2019. [arXiv:1909.01007](https://arxiv.org/abs/1909.01007).