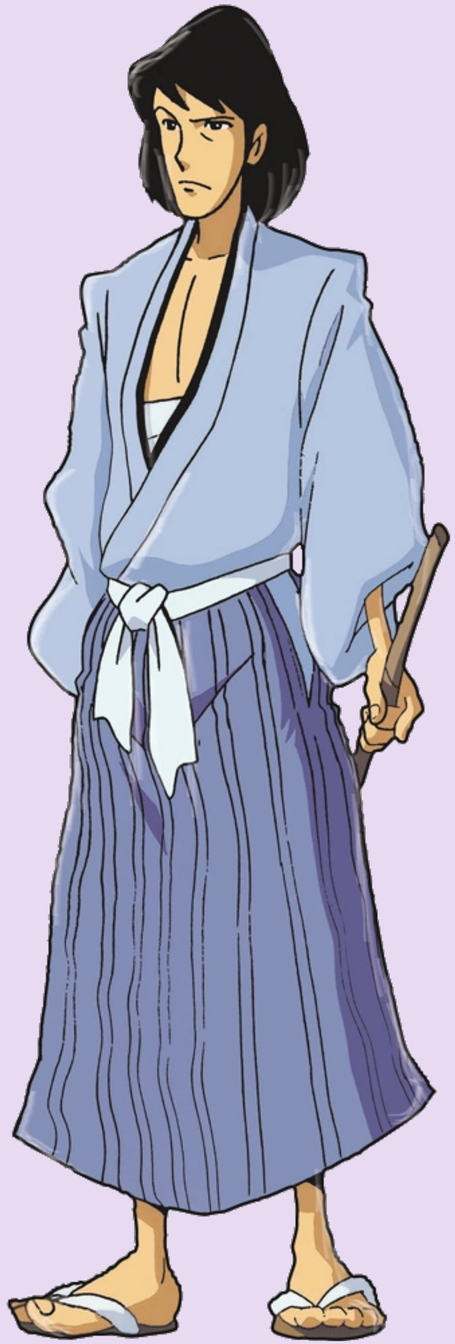
Team Goemon Ishikawa

Touché Argument Retrieval for controversial questions

Matteo Carnelos
Laura Menotti
Thomas Porro
Gianmarco Prando



Search Engines a.a. 2020/2021 - June 3rd, 2021

Introduction

The Information Retrieval (IR) system built by Team Goemon provides a solution for the Touché Task 1: Argument Retrieval for Controversial Questions. Given a question on a controversial topic, the goal is to retrieve arguments from a focused crawl of online portals.

The required IR system will parse the JSON files, index the retrieved document and produce a run on 50 different provided topics. For each topic, the system will retrieve the 1000 most relevant documents.

We adopted a two phases methodology. Firstly, we built a basic IR system with some standard processing, indexing and searching in order to have a robust and working project structure. Subsequently, we focused on improving our system's efficiency with some changes in the analyzer and in the similarity function

Related Work

We are provided with last year results therefore we used them as a baseline start for our solution.

Last year winning submissions:

- **DirichletLM Similarity**

  Probabilistic language Model like BM25, that relies on Latent Dirichlet allocation (LDA).

- **Query Expansion using synonyms**

  Query Expansion (QE) is a set of techniques that reformulate the query to improve performance of a IR system. Such techniques used can involve finding synonyms or semantic related words or fixing any spelling errors in the query formulation.

TEAM	RETRIEVAL	nDCG@5
Weiss Schnee	DPH	0.804
Prince of Persia	Multiple models	0.791
The Three Mouseketeers	DirichletLM	0.789
Swordsman (baseline)	DirichletLM	0.769
Dread Pirate Roberts	DirichletLM/Similarity-based	0.743
Prince of Persia	Multiple models	0.724
Thongor	BM25/DirichletLM	0.717
Oscar François de Jarjayes	DPH/Similarity-based	0.699
Black Knight	TF-IDF	0.692
Utena Tenjou	BM25	0.689
Arya Stark	BM25	0.662
Prince of Persia	Multiple models	0.642
Prince of Persia	Multiple models	0.641
Don Quixote	Divergence from Randomness	0.617
Aragorn	BM25	0.593
Prince of Persia	Multiple models	0.555
Aragorn	BM25	0.331
Aragorn	BM25	0.319
Aragorn	BM25	0.288
Aragorn	BM25	0.271

Methodology - Building the baseline

PARSE

- Most useful fields to assess the relevance of the documents: conclusion, text and stance.
- The parse method extracts the id and the relevant fields for the indexing and produces a ParsedDocument object.
- Since the Touché corpus is composed of JSON files, the parsing uses the Jackson library.
- Topic parsing: the XML file containing the topics are parsed and each topic is represented as a QualityQuery object with a field that corresponds to the topic title.

INDEX

- Index options for the fields: tokenized, not stored. The index keeps only document ids and term frequencies in order to minimize the space occupation.
- The Java class DebateIndexer indexes the debates documents processing a whole directory tree, since there are different files.
- For each ParsedDocument object, we create a Lucene Document object with the above-mentioned fields plus the document id, which is stored as a StringField.
- Each indexed document's id is put inside an HashMap in order to eliminate duplicates

SEARCH

- In the DebateSearcher class, after the topics have been parsed and represented as QualityQuery objects, queries are built and the most relevant document for each query are searched by means of the Lucene search method.
- After the search, the score of each document for each topic is stored in the run file following the trec convention.

Methodology

- We initially used an analyzer with off-the-shelf tokenizer and stop list provided by the Lucene library and the lowercase filter while we adopted the BM25 similarity. At this point the system was tested on last year topics and relevance judgments.
- We tested several combinations of Lucene Tokenizer, OpenNLP Tokenizer, Lucene stop, Atire stop, Terrier stop, Smart stop, no stem and OpenNLP Lemmatizer.
- We **changed the similarity function from BM25 to Dirichlet** after a major improvement in the system performance.
- **New feature: Query Expansion**
  - Query expansion method provided by Lucene after implementing the predinef filters already defined in Lucene (SynonimMapFilter and FlattenGraphFilter)
  - For the construction of the synonym list we have used the WordnetDatabase. In particular, we used the filewhich comprises the most popular synonyms of the English Language
  - Once we have parsed the file of Wordnet we just build the SynonymMap through its builder and the Object is passed to the SynonymMapFilter method.

Results

Overall performance:

- **Major difference between systems using BM25** as similarity **and** the ones using **Dirchlet**

  This behaviour is visible in all the measures, the major gap is for nDCG@5, which is the targeted measure for Touché committee.

- **Query Expansion made little difference**

  We conclude that, given the specificity of the topics, query expansion slightly helped in achieving our goal although there is not much difference between the same system with or without such technique.

Run	AP	P@10	nDCG	nDCG@5
drchLcnLcn	0.3508	0.5367	0.6593	0.7844
drchLcnLcnQE	0.3514	0.5388	0.6593	0.7810
drchLcnTrrQE	0.3312	0.4959	0.6453	0.7277
drchOpnTrr	0.3202	0.4735	0.6344	0.7184
drchLcnTrr	0.3262	0.4918	0.6399	0.7176
drchLcnSmrt	0.3221	0.5000	0.6356	0.7000
drchLcnAtr	0.3171	0.4939	0.6311	0.6859
drchOpnAtrOpn	0.3076	0.4714	0.6256	0.6657
bm25LcnLcnQE	0.2692	0.4041	0.5850	0.4887
bm25LcnTrrQE	0.2568	0.3816	0.5842	0.4745
bm25LcnLcn	0.2706	0.3592	0.5917	0.4519
bm25LcnAtr	0.2444	0.3367	0.5705	0.4244
bm25OpnAtrOpn	0.2307	0.3306	0.5493	0.4175

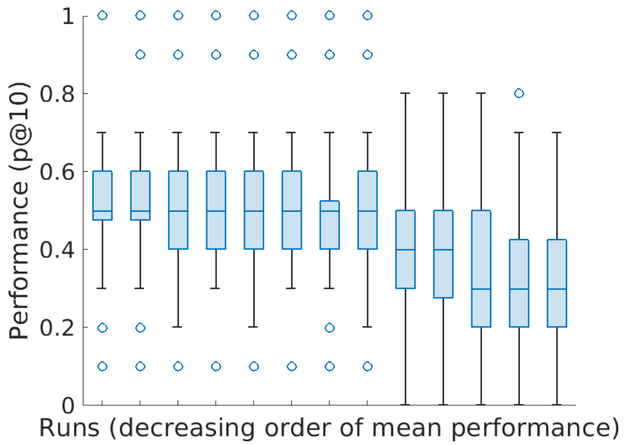
Results for 2020 topics.

Box Plots

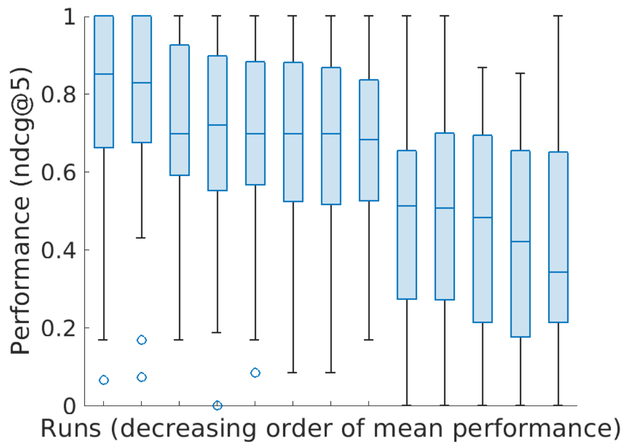
Overall slopes trend:

- **Variance of the BM25 runs is significantly bigger** than the variance of the Dirichlet runs
  - In the box plot for P@10, the variance gap between the Dirichlet and BM25 runs is clear.
  - In the box plot for nDCG@5, two of the BM25 runs have variance between 0 and 1 *(i.e. maximum range of the measure)*.
- **Presence of outliers only in Dirichlet runs**

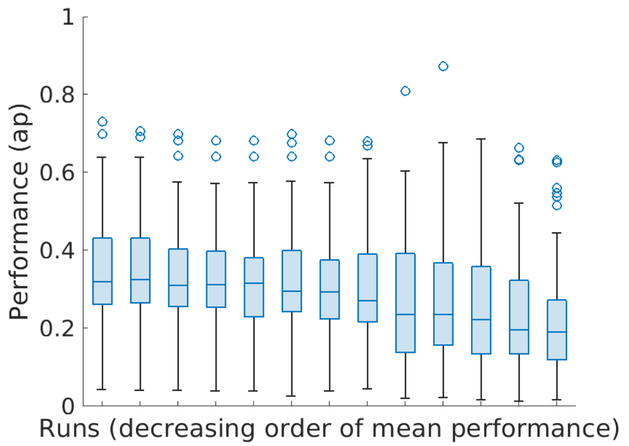
  This confirms that Dirichlet systems are stable and they all fail and succeed on the same topics.



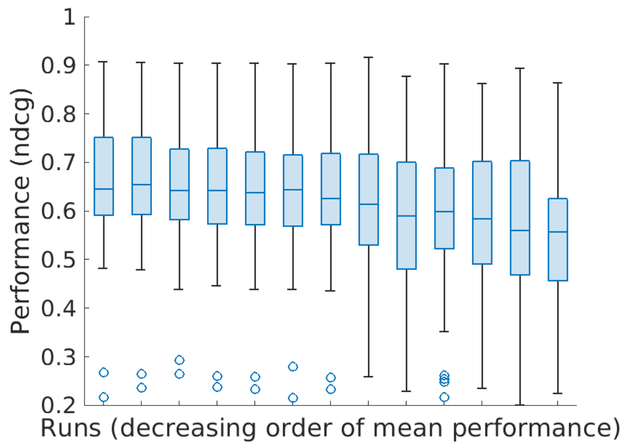
Box plot for **P@10**.



Box plot for **nDCG@5**.



Box plot for **AP**.



Box plot for **nDCG**.

Anova

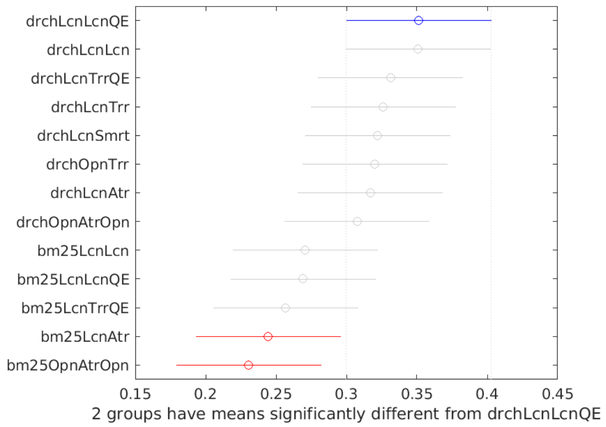
In this analysis we focused on the differences between various configuration of the systems to understand if some configurations are significantly different from others.

Overall analysis:

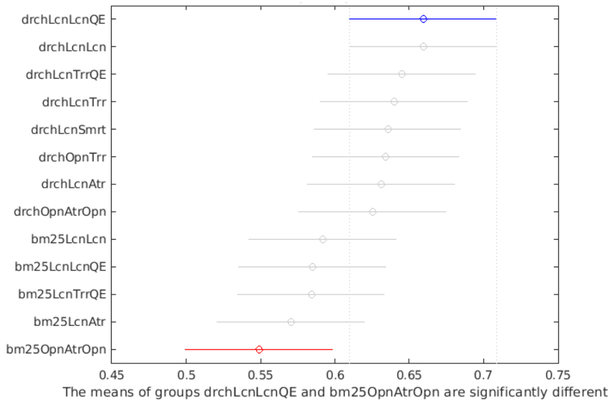
- Dirichlet and QE are a good combination for performance improvement
- Systems built with Query Expansion and BM25 as similarity tend to have slightly worse performance than the same without QE.

General measures:

- All the changes have **not provided significant improvements.**
  - We can state that our best run is not significantly different from one of our worst



Anova for **AP**.



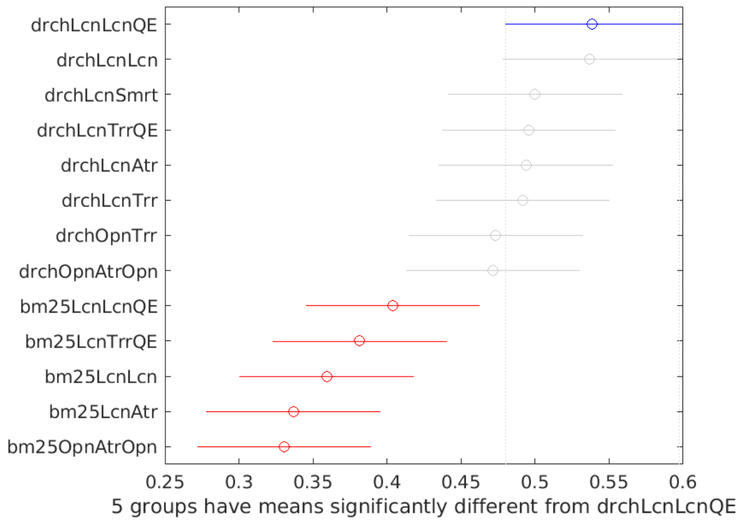
Anova for **nDCG**.

Anova

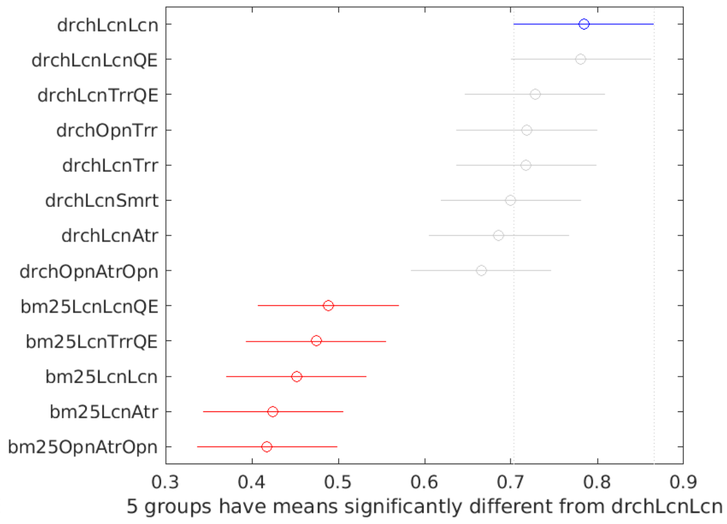
Cut measures:

If we restrict the query results taken into account for the calculation of the measures, *i.e. cut at 5 or cut at 10*, differences start to accentuate.

- All the runs with Dirichlet are significantly different from the ones with BM25.
  - This might be caused by the fact that Dirichlet provides more relevant results on the top ranked documents
- We notice that the best two runs, that use the default stop list of Lucene, performs better than others using larger lists.



Anova for **P@10**.



Anova for **nDCG@5**.

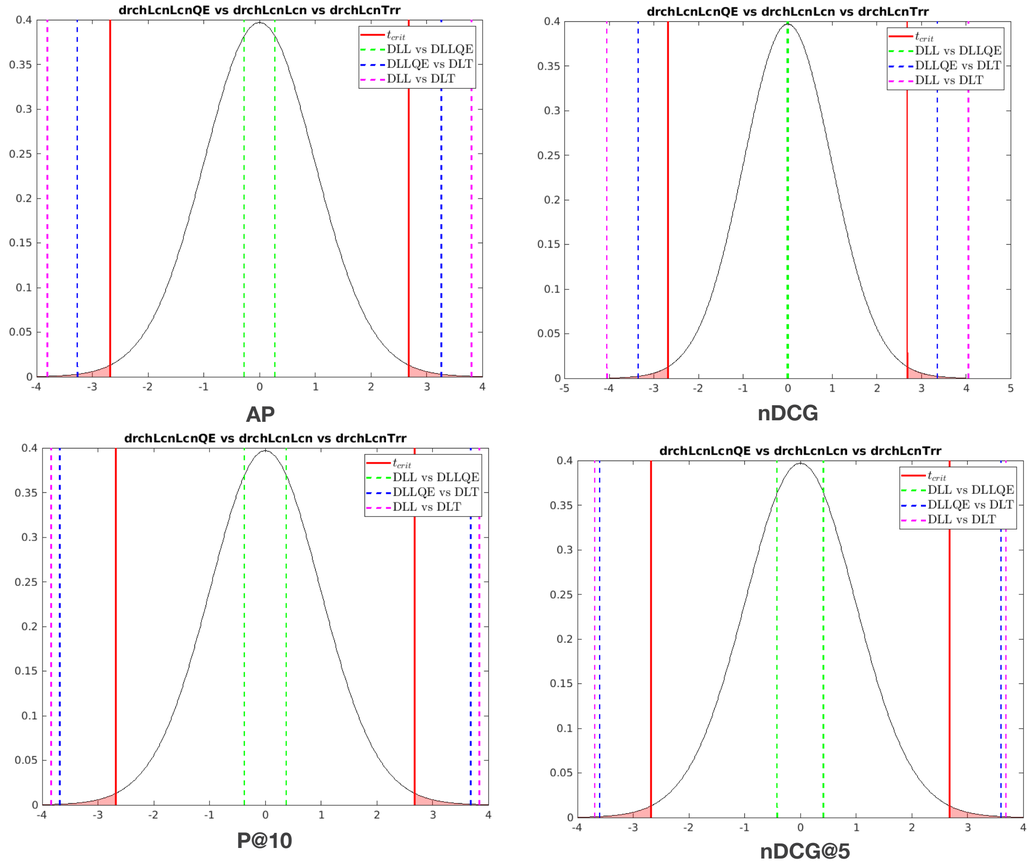
Student's T-test

We decide to analyze three runs that are not statistically different in all the ANOVA plots. This decision comes from the need to determine whether any Dirichlet system is statistically different from another of the same group.

Compared Runs	AP	P@10	nDCG	nDCG@5
drchLcnLcn vs drchLcnLcnQE	0.7840	0.7096	0.9946	0.6779
drchLcnLcnQE vs drchLcnLcnTrr	0.0020	0.0006	0.0016	0.0007
drchLcnLcn vs drchLcnTrr	0.0004	0.0004	0.0002	0.0006

Overall Analysis:

- The system using Terrier as stop list has a high probability of being statistically different from the two best-performing ones for all the four measures.
  - For all the measures the probability is less than 0,1%.
- The presence of Query Expansion did not make such difference.
  - In most measures the probability of the two systems being statistically different is greater than 68%, peaking at 99,5% for nDCG.



Failure Analysis

In order to fully understand whether Query Expansion improved our system or not we push a bit further and, as suggested, we made a failure analysis.

Topic 26:

- Query Expansion **worsen** the performance
  - The question refers to tests, but not in a broad sense, it implicitly refers to school tests. Since the word "test" can be referred to various contexts (school test, brake test, clinical test...), the introduction of synonyms can divert the meaning of the query, worsening the results (also known as **query drift**).

Topic 4:

- Query Expansion **improve** performance instead
  - The string *"corporal punishment"* is very specific and not frequently used. So, adding synonyms can widen the scope and more relevant documents can be retrieved.

- Topic ID: 26
  Topic name: Do standardized tests improve education?
  nDCG@5 for drchLcnLcn: 0.8539
  nDCG@5 for drchLcnLcnQE: 0.6992

- Topic ID: 4
  Topic name: Should corporal punishment be used in schools?
  nDCG@5 for drchLcnLcn: 0.8688
  nDCG@5 for drchLcnLcnQE: 1.0

Conclusions

Conclusions

- Statistical models works good for this type of documents.
- OpenNLP models are computationally heavy and provide very light improvements if none with this corpus.

Further Improvements

- Machine Learning techniques, e.g. Sentiment Analysis, could improve our system's performance.

  For example, knowing a-priori if a query expects pro or con arguments we can filter the results in order to better match user sentiment.

- A more precise query expansion could be done by using a bigger and more specific synonyms database.

Thank You!

Matteo Carnelos, Laura Menotti, Thomas Porro, Gianmarco Prando