Weekly Progress

27.04. - 04.05.2018

Spell-checking

Implemented spell-checking for queries

- in class “framework.querying.GenericQueryGenerator.java”
- using Lucene.search.spell.SpellChecker
- public static String correctSpelling(queryString){...}

Spell-checking

public static String correctSpelling(queryString){...}

- Splits input query into subStrings that are one word each
- Checks every word for spelling errors
- Corrects every misspelled word
- Connects all corrected subStrings to a full queryString and returns this String

Spell-checking

Problems:

- Which dictionary is best to use?

```
// To index a field of a user index:
spellchecker.indexDictionary(new LuceneDictionary(my_lucene_reader, a_field));
// To index a file containing words:
spellchecker.indexDictionary(new PlainTextDictionary(new File("myfile.txt")));
```

- Current dictionary: plaintext words.txt
  - ➢ weird spelling-error correction
  - ➢ E.g.: “daeth” to “doeth”

TODO:

- Frontend: Tell user that his input was corrected/altered

Suggesting queries

Implemented first steps to suggest most debated conclusion

- In class framework.retrieval.SampleLuceneRetriever.java
- Code currently in retrieveArguments(query){...}
  - can be put into separate function
- “filters” retrieved arguments to only include ones with the most debated conclusion

Suggesting queries

- Takes retrieved Arguments Set<Arguments>, converts it into a List, then into a Map<String, Integer> to count the most debated conclusion
  - list.get(i).conclusion()

- Removes every argument whose *conclusion* is not the same as the most debated one
  - Convert into Set<Arguments> and return this as the retrieved Arguments

- Comparing conclusion strings:
  - String.equals()
  - getLevenshteinDistance() < k

Suggesting queries

TODO:

- Migrate code into separate function
  - Filter before retrieval takes place?

- Search for *k* most debated conclusions
  - Currently just the top most debated one

- Don’t suggest conclusion if the user’s conclusion is already well worded?

- Frontend: Tell User that his query was altered and give the option to go back to his original query

Suggesting queries

TODO:
Front end :

- ***Negate*** function
  - give user the option to Negate the suggested query
  - will switch between pros and cons

- ***Not what I meant*** function
  - Will iterate to next most valued conclusion
  - Will have 10 most debated conclusion related to the query

- ***Search for*** function
  - Will give the user to search for the result for his own query