Q&A

What is qrel.txt?

This file contains manual relevance judgments for our set of queries and documents. See `Treceval info.pdf` for information about the file format and how to use it with the `treceval` utility for generating results.

What should my engine output?

Output a file that includes the top results for each query. See `Treceval info.pdf` for the file format.

What queries do I use?

Your engine should operate in "batch mode", processing many queries in one go. Use the queries found in `queryfile.txt` . This file resembles XML, but you'll need to write your own parser for it because it's not well-formed XML. The <num> field represents the query number (you'll need this for your output). Use only the <title> for each query (ignoring "Topic:"; <desc> and <narr> should be discarded.

Can I modify my P1?

Yes, feel free to modify your P1 to fit the needs of your P2, and correct any errors that will effect the operation of your P2.

How do you decide if the "phrase terms are common"?

It's up to you to pick a reasonable value.

How do you use the positional (proximity) index?

Since the proximity index contains the positions of each term within the document, it can serve some of the same functions as the phrase index, but with additional capabilities. Let's say you have a query "Milwaukee Brewing Company", and all three terms only appear in 2 documents: A and B.

A: "In Milwaukee, there is a long history of brewing. It is also known for being the company headquarters of Koss."

B: "Earning were announced on Monday for Milwaukee Brewing. The company boasted increased profits."

Clearly, A is not talking about the brewery, while B is. Neither has a match in the phrase index. In the single

term and stem index, they would receive the same score. In the positional index, you can use heuristics such as "all the query terms must appear within 5 terms of each other" to condition matches in the positional index. In this example, B would match this rule, and A would not. Think about what other types of heuristics you could use to better improve retrieval results, and try them out with your engine!

Instructions for preparing the code submission (Project 2)

Your program needs to be run from a shell interface (e.g. terminal, command prompt) and it needs to accept arguments. The arguments are inputs, outputs, and the configuration to your code. Please prepare your main program according to the following (including the name of the module and the arguments). Please note that the order of the arguments is important. Your code needs to run regardless of the underlying platform (if you have trouble achieving this, let us know). Please also include all the resources that your code uses in your submission.

building the index

```
build [trec-files-directory-path] [index-type] [output-dir]
```

Builds the index, accepts 3 arguments

- `[trec-files-directory-path]` the directory containing the raw documents (e.g. fr940104.0)
- `[index-type]` can be one of the following: "single", "stem", "phrase", "positional"
- `[output-dir]` the directory where your index and lexicon files will be written (please make your program create it if doesn't exist).

example command for building the index:

```
python build.py /tmp/Mini-Trec/BigSample/ phrase /tmp/my-indexes/
```

query processing (report 1, static)

```
query_static [index-directory-path] [query-file-path] [retrieval-model] [index-type] [r
```

Static query processing. Runs the specified retrieval model on the specified index type. Accepts 5 arguments.

- `[index-directory-path]` takes the path to the directory where you store your index files (the [output] of the "build index" step).
- `[query-file-path]` path to the query file

- `[retrieval-model]` can one of the following: "cosine", "bm25", "lm"
- `[index-type]` one of the following: "single", "stem"
- `[results-file]` is the path to the results file, this file will be run with trec_eval to get the performance of your system. Please have your program create directories if necessary.

example command for query processing:

```
python query.py ./indexes/my-indexes/ ./data/queryfile.txt bm25 stem ./results/results-
```

query processing (report 2, dynamic):

```
query_dynamic [index-directory-path] [query-file-path] [results-file]
```

Same as static query processing, except that you need to dynamically change the index you are querying based on the thresholds (please carefully read assignment instructions). Takes 3 arguments.

- `[index-directory-path]` takes the path to the directory where you store your index files (the [output] of the "build index" step).
- `[query-file-path]` path to the query file
- `[results-file]` is the path to the results file, this file will be run with trec_eval to get the performance of your system. Please have your program create directories if necessary.

Example command:

```
python query_dynamic.py ./indexes/my-indexes/ ./data/queryfile.txt ./results/results-dy
```