Report:

Gaurav’s Implementation:

* Framework that enabled you to re-rank top n documents using multiple retrieval algorithms.
* Framework for generating a result file with top n without re-rank to compare with the one generated from the re-ranked framework.
* Caching extended to all the retrieval algorithms (previously only worked on TFIDF Improved and DIRICHLET) now works on BM25 and BM25+
* Caching also extended to cache Tagme results to avoid hitting the server.
* Merged tagme and my results generation pipeline now retrieval algorithms can be used in any combination with query expansion enabled or disabled.
* Tested code on the server provided. (Installed Conda and setup testing framework)

**Generating Cache (Note: Updated in Prototype 3)**

\*\* Caching now works with tagme queries \*\*

tc\_generate\_document\_cache.py [outlines file] [paragraphs file] [no of passages to extracts from paragraph file] [use\_tagme\_enhancement]

[use\_tagme\_enhancement] : enhanced, un\_enhanced

**(top-n re-ranked)Generating trec\_eval compatible results using top n re-ranking algorithm ( Note: New in prototype 3 )**

python tc\_rerank\_document\_framework.py [outlines file] [paragraphs file] [output file] [primary\_retrieval\_algorithm] [re-ranking\_algorithm] [cache] [no\_of\_results\_to\_re-rank] [no of passages to extract] [use\_tagme\_enhancement]

The aforementoned arguments can take the following value:

[primary\_retrieval\_algorithm]: BM25, BM25+, TFIDFIMPROVED

[re-ranking\_algorithm] : DIRICHLET

[cache] : no\_cache, cache ( Note 'cache' only works if tc\_generate\_document\_cache.py is run first on same number of passages )

[no\_of\_results\_to\_re-rank] : an integer (less than no of passages being extracted)

[no of passages to extract] : an integer

[use\_tagme\_enhancement] : enhanced, un\_enhanced

Sample Run Statement:

Note: For a quick result evaluation use the un\_enhanced but if you want to use enhanced generate the cache first.

python tc\_rerank\_document\_framework.py all.test200.cbor.outlines release-v1.4.paragraphs output.top500reranked.run TFIDFIMPROVED DIRICHLET no\_cache 500 50000 un\_enhanced

**(top-n not re-ranked)Generating trec\_eval compatible results top - n without reranking ( Note: New in prototype 3 )**

\*\*\*\* Note:

This is only to compare the top n implementation as results from top n might be a little bit lower than the the entire thing this is a good way to generate a top n only results file without re ranking to compare it with the re-ranked implementation

\*\*\*\*\*

python tc\_rerank\_document\_framework.py [outlines file] [paragraphs file] [output file] [ranking\_function] [cache] [top\_n\_only] [no of passages to extract] [use\_tagme\_enhancement]

The aforementoned arguments can take the following value:

[ranking\_function]: BM25, BM25+, TFIDFIMPROVED, DIRICHLET

[cache] : no\_cache, cache ( Note 'cache' only works if tc\_generate\_document\_cache.py is run first on same number of passages )

[top\_n\_results\_only] : an integer (less than no of passages being extracted)

[no of passages to extract] : an integer

[use\_tagme\_enhancement] : enhanced, un\_enhanced

Sample Run Statement:

Note: For a quick result evaluation use the un\_enhanced but if you want to use enhanced generate the cache first.

python tc\_rerank\_document\_framework.py all.test200.cbor.outlines release-v1.4.paragraphs output.top500notreranked.run DIRICHLET no\_cache 500 50000 un\_enhanced

**Generating trec\_eval compatible results file all results (Note: Updated in Prototype 3)**

tc\_generate\_document.py [outlines file] [paragraphs file] [output file] [ranking function] [cache] [no of passages to extract] [use\_tagme\_enhancement]

The aforementoned arguments can take the following value:

[ranking function] : BM25, BM25+, TFIDFIMPROVED, DIRICHLET

[cache] : no\_cache, cache ( Note 'cache' only works if tc\_generate\_document\_cache.py is run first on same number of passages )

[no of passages to extract] : an integer

[use\_tagme\_enhancement] : enhanced, un\_enhanced