Prototype-1 Report

Github repository: https://github.com/ba1044/DataScience_FinalProject

Prototype1 release: https://github.com/ba1044/DataScience_FinalProject/releases/tag/Prototype1

Method Description:

Frequent Bigram:

1. Inspect all paragraphs for frequent bigrams. Take top 10 bigrams (term1, term2) with the highest bigram score; Create a index field called “bigram” for each paragraph contains those 10 bigrams.
   Meanwhile, also index the full text of each paragraph
2. bigram score = p(term1, term2) / (p(term1) * p(term2)).
3. Combine queries:
   a. BM25 section-path query against "text" field
   b. Using frequent bigrams in the section path query (each bigram as single term): BM25 bigram query against "bigram" field

Query Expansion:

1. Index the full text of each paragraph
2. First feedback run with BM25 page_name(section_path) query
3. Use custom stopword list to get top 5 terms to expand query.(10 terms will cause too much noise)
4. Second run with BM25 expanded query.

Heading Weight Variation

Using base Index just the full text of each paragraph.

BM25 Query of just the page name:BM25 Query of just the page name: for just the page name extracting page id and page name.

BM25 Query of just the lowest heading: Extracted the lowest heading from section path.

BM25 Query of the interior headings: Extracted all the interior heading of section path.

Word-Embedding:

Using baseline index and BM25 section path query to retrieve a candidate set (top 100); then reranking the candidate set as following

Define query vector as the average of word vectors of all query terms

- Define document vector as average of word vectors of all document terms
- Ranking by cosine similarity of of query and document vector for each query term, identifying closest document term (by word vector cosine similarity)

The results of word embedding are not so good.

Weighted Features: the following methods are fully described in the last section of the repository's readme. In each method, a weighted combination of features (trained using RankLib) is used, in addition to BM25, to reweight queries:

- **average_query**: The query is broken up into multiple query terms, and a paragraph's score is equal to the average of these queries against the paragraph's text (using BM25).
- **split_sections**: The query is split up by section path and each token is treated as a separate feature (score again comes from BM25 against paragraph text)
- **mixtures**: Each paragraph has an associated distribution over entities. Paragraphs are scored according to their distribution over entities with the top 100 paragraph's distributions over entities.
- **lm_dirichlet**: Score paragraphs according to Lucene's LMDirichletSimilarity.
- l**m_mercer**: Score paragraphs according to Lucene's LMJelinekMercerSimilarity
- **entity_similarit**y: A paragraph's score is expressed as the average similarity between each of the query's tokens and the paragraph's entities. The Jaccard and JaroWinkler string similarities are used.
- **combined**: Uses a combination of features from split_sections, lm_dirichlet, and entity_similarity that appeared to be the most effective after training.

Location of Required Files on the Cluster Server:
The old index is at: /home/jsc57/programs/jsr-lucene-variant/myindex
The new index (when it is finished) is at: /home/jsc57/group_project/DataScience_FinalProject/index
The --spotlight_folder directory is located at: /home/jsc57/jsr-lucene-project/spotlight_server
The --graph_database file is located at: /home/jsc57/jsr-lucene-project/graph_database.db

Code Contribution:

Jordan: Implemented argument parser (in Main) and everything in the Kotlin directory. Also added documentation (including readme). Did entity-linking, graph walks, and training with RankLib. Implemented methods described in "Weighted Features" in the section above.

Bindu: Implemented heading weights variation and word embedding. Updated details in documentation under heading weight section and word embedding.

Kaixin: Implemented Frequent Bigram variation and Query Expansion. Also some Project utilities method and part of the indexing function.

Evaluation Results on bechmarkY1train:

Method	MAP	MRR	Rprec	Author
BM25Query-(Just the page name)	0.0822	0.4054	0.1393	Bindu
BM25Query(Just the lowest heading)	0.0585	0.0802	0.0426	Bindu
BM25Query (interior heading	0.0167	0.0266	0.0102	Bindu
WordEmbedding	0.0046	0.0389	0.0107	Bindu
average_query	0.1038	0.1479	0.0769	Jordan
entity_similarity	0.1044	0.1486	0.0779	Jordan
lm_dirichlet	0.1078	0.1526	0.0816	Jordan
lm_mercer	0.1064	0.1505	0.0798	Jordan
mixtures	0.1044	0.1493	0.0778	Jordan
split_sections	0.1094	0.1532	0.0816	Jordan
combined	0.1183	0.1649	0.0891	Jordan
FrequentBigram(Page Name)	0.0488	0.2060	0.0832	Kaixin
FrequentBigram(Section_path)	0.0325	0.0488	0.0179	Kaixin
Query Expansion(Page Name)	0.0734	0.3333	0.1279	Kaixin
Query Expansion(Section_path)	0.0822	0.1175	0.0612	Kaixin