Web Mining and Information Retrieval

Programming Assignment 1— Vector Space Model and Rocchi Relevance Feedback

B01501085 莊東諺

0. My Vector Space Model

In my implementation of VSM, I use Python and its feature to create my VSM search engine. There are two python file in my program: `dict.py` contains some preprocess and all data like TF and IDF implementation; `main.py` is the main process of query. The table below explains my main parameters and data structures.

```
Dict.py
      Variables
      vocab:        save all unique terms in (vocab_id_1, vocab_id_2) format.
      vocabAll:     a dict, key is term in unicode representation and value is its
                    id(from vocab_all file).
      vocabID:      a dict, key is term's id and value is its unicode representation.
      rawtermfreq:  a two dimension dict, give an key set [term][docID], it will return
                    this term's frequency in the document docID.
      nq:           given a key term, returns its frequency in all document collection
      N:            number of documents in this collection.
      doclist:      key is docID, value is this document's path.
      bytelenght:   key is docID, value is this document's length in byte.
      aval:         average byte length of the collection

      Methods
      build():      initialize all requiste data from model/ dir
      query2term(query): given a chinese string query, split it, remove stopwords and
                    generate all unigram and bigram, retrun a list of terms
      idf(term): compute this term's Inverse Term Frequency
      tf(term, docID): compute this term's Term Frequency in the document docID
      normal(vec):    normalization of the vector
```

The procedure of my VSM
Preprocess and build the VSM
—> parse the query.xml, extract all data(title, narrtive, question, concept) to generate the query vector
—> computer document's TFIDF vector and query's TFIDF vector
—> compute query and every documents dot value, return top 100 result

1. My Rocchio Relevance Feedback

In my feedback strategy, I simplely take top 5 ranked document as "relevant document". I count every term's "n * idf", where "n" is how many times the term appear in these relevant documents. I pick top 10 terms and add them into the query. After query expansion, I use this new query to recount the score of all documents in the collection.

2. Experiments

No any preprocess, only use "concept" to query: **0.49043**
Use this formula

$$score(D,Q) = \sum \left(\frac{1+\log(1+\log(f)) * df(n)}{1 - c + c * b / avgl} * \frac{qf * 1}{1} \right)$$

- After I use every element (title, narrative, question, concept) to query, MAP value only increase by **0.04135**, still under the weak-base line
- Use different log base, but MAP value is still bad(**0.54027**)
- Use **Okapi/BM25** to query, MAP value increase rapidly to 0.70741, but still bad
- Use both unigram and bigram to query, increase MAP value to **0.73110**
- Implement Feedback and use parameter k1 = 1.2, b = 0.75, reached my best MAP value **0.74349**
- **My final weighting formular:**
  - wq = term frequency in query / normalization * IDF of term
  - wd = raw term frequency * (k1 + 1) / (raw term frequency + k1 *(1-b+b*byte length / average byte length))

---

3. Discussion and Observation

- The base of log is not important.
- The more query term used, the higher MAP value. I think because it provides more information to improve the result.
- With my feedback strategy, it has a slight increment in MAP value. In my submission, the feedback increse about 0.003.
- The performance of bigram and unigram is better than bigram only.

I have learned that in VSM, it is really important to parameter tuning, it can affect the performance significantly. Unfortunately, my parameter can not push my rank further due to the execution speed and implementation. On the last day of this assignment, I suddenly realize a faster way to reduce my execution time, but there is no more submission avalible.