William Ibekwe
wibekwe
Homework 4 - 11791
hw4-report.pdf

Task 1.1 - Input

Task one was very self explanatory. The task around this model is to make that any question or answer model format will have similar results.

The format is as follows:

qid=1 rel=99 Classical music is dying
qid=1 rel=0 Pop music has absorbed influences from …
qid=1 rel=1 Classical music may never be the most …
qid=1 rel=0 Everybody knows classical music when …

The qid is represents the query/question of which each sentence belongs to. rel represents the type of sentence this is i.e. 99 = a question/query, 1 is a correct answer to a query and 0 is and incorrect answer.

Task 1.2 - Type System

The type system used in my homework followed the same one from what was given.

1.) Token - used to describe a single 1gram, containing the text of that gram and the frequency it occurs in a sentence.
2.) Document- this contains a parsed list of all the Tokens in a sentence using and FSArray instead of the a FSList. I did this because I wanted to make the coding of this easier since i'm not really experience in coding. The Document also wanted a rel type, the qid, as well as the entire sentence as a whole.

There was not much we could change because of the VectorSpaceRetrieval Annotator. This was taking each line of a file and treating like a document by reseting the CAS at every line.

Non - UIMA Type System

Because there were either type that were not implemented in the UIMA - Type System. I used other types to help me with the analysis if this project.

1.) **HashMap** - I used a java Hash Map to store a vector space of each Token in a Document.
2.) **Answers** - I created a data type that stored answers after all documents were parsed
3.) **Triplet** - I created this data type to combine three like objects for sorting and printing purposes.

Task 1.3 - Analysis Engine

DocumentReader - This Annotator simply parsed all the content in a file and through made them into Documents.

DocumentVectorAnnotator - I used this annotator to parse the the sentence in to Tokens that will be used in the retrieval architecture.

RetrievalEvaluator - Once all the text has been parsed into tokens and the vector spaces, we can now run our analysis. We start by putting this code through bag of words comparison and cosine similarity algorithms then from there we run them through **BONUS** similarity algorithms.

Task 2 - Error Analysis

The following is a list of error analysis steps I took to getting to the current homework version.

1. Correctly extract bag of words feature vector from the input text collection
    1. A couple of things that I noticed here in bag of words comparison is that only worked well for sets that had the same amount of tokens as the query/ question. In order to improve the retrieval technique and the mrr a better algorithm is needed. One that will interpret sentences not amount of token matched but by the similarities.
2. Compute the cosine similarity between two sentences in the text collection
    1. It was hard to keep track for the frequency of the word from just putting them in List so I initiated a Hash Map that would allow me to enter these words more freely. Here is I got my bag of words running.
3. Compute the Mean Reciprocal Rank (metric) for all sentences in the text collection
    1. Cosine similarity worked a lot better than Bag of word so we decided to use this a the base line. We were able get ok MRR results but now it is time to improve.
4. Design a better retrieval system that improves the MRR performance measure
    1. I notice that some words were not being recognized by other simply because of the case of the word. So I made all the words lower this way there would be a better recognition platform.
    2. I then noticed that the majority of the matches in the code and the algorithms were do to the matching of the stop words. When I took out all the stop words using the stop words list provided. The MRR reduced. This behavior was expected because the matching was mostly based on stop words.
    3. From there I wanted to stem the token so that all the root word would be init place instead of the tenses. This produced a better out put that I thing was acceptable.
5. Improve the efficiency of the program by doing error analysis of retrieval system
    1. My algorithm was working very slow especially because I have multiple algorithms running at the same time. Also it was hard to sort groups of items that I wanted to keep together. What I ended up to is creating my Triplet class that grouped three things together and sorted on one of them to give me good result. The items that I used in the Triplet class was the Score(Double), qid that that answer belongs to(Integer), and whether or not that answer was correct (Integer). This help me become more efficient because I did not have to keep static variable about each answer, which in turn gave me faster runtimes

BONUS

For the bonus I implemented 3 more algorithms Jaccard, Dice Coefficient, and Hamming Distance. Jaccard and Cosine Coefficient seemed to work the best our of all four Algorithms.

```
Cosine Similarity Measurement
**********************************************************
Score: 0.6123724356957946   rank: 1  rel: 1  qid: 1  sent2
Score: 0.5163977794943222   rank: 2  rel: 0  qid: 1  sent3
Score: 0.3849001794597505   rank: 3  rel: 0  qid: 1  sent1
Score: 0.4629100498862757   rank: 1  rel: 1  qid: 2  sent3
Score: 0.0                  rank: 2  rel: 0  qid: 2  sent1
Score: 0.0                  rank: 3  rel: 0  qid: 2  sent2
Score: 0.5303300858899106   rank: 1  rel: 0  qid: 3  sent2
Score: 0.4472135954999579   rank: 2  rel: 1  qid: 3  sent1
Score: 0.22360679774997896  rank: 3  rel: 0  qid: 3  sent3
Score: 0.1543033499620919   rank: 1  rel: 1  qid: 4  sent2
Score: 0.1543033499620919   rank: 2  rel: 0  qid: 4  sent3
Score: 0.14433756729740646  rank: 3  rel: 0  qid: 4  sent1
Score: 0.2857142857142857   rank: 1  rel: 0  qid: 5  sent2
Score: 0.2182178902359924   rank: 2  rel: 1  qid: 5  sent3
Score: 0.1690308509457033   rank: 3  rel: 0  qid: 5  sent1
(MRR) Mean Reciprocal Rank: 0.8

Jaccard Coefficient Measurement
**********************************************************
Score: 0.3333333333333333   rank: 1  rel: 1  qid: 1  sent2
Score: 0.3333333333333333   rank: 2  rel: 0  qid: 1  sent3
Score: 0.125                rank: 3  rel: 0  qid: 1  sent1
Score: 0.3                  rank: 1  rel: 1  qid: 2  sent3
Score: 0.0                  rank: 2  rel: 0  qid: 2  sent1
Score: 0.0                  rank: 3  rel: 0  qid: 2  sent2
Score: 0.2857142857142857   rank: 1  rel: 1  qid: 3  sent1
Score: 0.2857142857142857   rank: 2  rel: 0  qid: 3  sent2
Score: 0.125                rank: 3  rel: 0  qid: 3  sent3
Score: 0.1                  rank: 1  rel: 0  qid: 4  sent1
Score: 0.08333333333333333  rank: 2  rel: 1  qid: 4  sent2
Score: 0.08333333333333333  rank: 3  rel: 0  qid: 4  sent3
Score: 0.16666666666666666  rank: 1  rel: 0  qid: 5  sent2
Score: 0.1111111111111111   rank: 2  rel: 1  qid: 5  sent3
Score: 0.09090909090909091  rank: 3  rel: 0  qid: 5  sent1
(MRR) Mean Reciprocal Rank: 0.8
```

Dice Coefficient Measurement
****
Score: 0.45454545454545453 rank: 1 rel: 0 qid: 1 sent3
Score: 0.43333333333333335 rank: 2 rel: 1 qid: 1 sent2
Score: 0.2716049382716049 rank: 3 rel: 0 qid: 1 sent1
Score: 0.43956043956043955 rank: 1 rel: 1 qid: 2 sent3
Score: 0.3142857142857143 rank: 2 rel: 0 qid: 2 sent1
Score: 0.20253164556962025 rank: 3 rel: 0 qid: 2 sent2
Score: 0.5806451612903226 rank: 1 rel: 0 qid: 3 sent2
Score: 0.5714285714285714 rank: 2 rel: 1 qid: 3 sent1
Score: 0.5185185185185185 rank: 3 rel: 0 qid: 3 sent3
Score: 0.5052631578947369 rank: 1 rel: 0 qid: 4 sent3
Score: 0.4470588235294118 rank: 2 rel: 0 qid: 4 sent1
Score: 0.41304347826086957 rank: 3 rel: 1 qid: 4 sent2
Score: 0.4675324675324675 rank: 1 rel: 0 qid: 5 sent2
Score: 0.4482758620689655 rank: 2 rel: 1 qid: 5 sent3
Score: 0.4117647058823529 rank: 3 rel: 0 qid: 5 sent1
(MRR) Mean Reciprocal Rank: 0.5666666666666667

Hamming Distance Measurement
****
Score: 72.0 rank: 1 rel: 0 qid: 1 sent1
Score: 47.0 rank: 2 rel: 0 qid: 1 sent3
Score: 36.0 rank: 3 rel: 1 qid: 1 sent2
Score: 66.0 rank: 1 rel: 0 qid: 2 sent2
Score: 60.0 rank: 2 rel: 1 qid: 2 sent3
Score: 46.0 rank: 3 rel: 0 qid: 2 sent1
Score: 55.0 rank: 1 rel: 0 qid: 3 sent2
Score: 32.0 rank: 2 rel: 0 qid: 3 sent3
Score: 31.0 rank: 3 rel: 1 qid: 3 sent1
Score: 57.0 rank: 1 rel: 0 qid: 4 sent1
Score: 54.0 rank: 2 rel: 1 qid: 4 sent2
Score: 51.0 rank: 3 rel: 0 qid: 4 sent3
Score: 67.0 rank: 1 rel: 0 qid: 5 sent2
Score: 41.0 rank: 2 rel: 1 qid: 5 sent3
Score: 39.0 rank: 3 rel: 0 qid: 5 sent1
(MRR) Mean Reciprocal Rank: 0.4333333333333333

Total time taken: 0.901