11791-hw3

rgoutam

October 2014

1 Error Analysis

I manually did ana error analysis on the first 10 queries and classified errors into classes. The table below show these classes for each of the 10 queries.

qid	Error Class
1	word normalization, bag-of-words representation
2	bag-of-words representation
3	word normalization
4	word normalization
5	word normalization
6	word normalization
7	word normalization
8	word normalization
9	synonym
10	correct retrieval

The description of the error classes are below :

- word normalization : Words in the query and document are not normalized. Hence, even if two words are essentially the same, the surface form is not the same and hence, they are treated as two different words. For example, "state?" and "state." represent the same word "state" but surface forms are different.
- bag-of-words : more words in the query matched the document (which was not relevant) than a document (which was relevant)
- synonym : word in query is a synonym of the word in document. For example, "spaceship" and "spacecraft"

2 System Description

The system consists of 3 main components -

1. DocumentReader : This component reads the text file, processes it line by line, splits it based on tab and puts each column in the CAS.

2. DocumentVectorAnnotator : This component creates the term frequency vector for query and document. This component has different functions for tokenizing and normalizing words.

3. Retrieval Evaluator : The retrieval evaluator component uses different similarity measures to compute the similarity between a query and a document. It also computes the rank of documents to compute the MRR metric.

The system is a linear pipeline and each component is a part of the pipeline. The order of the components are DocumentReader, DocumentVectorAnnotator and Retrieval Evaluator.

3 Algorithm implemented

3.1 Tokenization

From my error analysis, word normalization was the biggest reason of error in retrieving documents. Hence, I improved the basic white-space tokenization algorithm by splitting words not just on white space but on any non-word character. This splits the words based on punctuations, symbols as well. I have also used the Stanford lemmatizer to get the lemma of the word, so that different forms/tenses of the same word get normalized. Furthermore, I have used the given stopword list to filter out stopwords and insert only content words in the term vector.

3.2 Similarity Measures

Cosine Similarity measure is something that I had to implement for task 1. It does not perform very well and has its own drawbacks. I have also implemented dice similarity measure to compute the similarity between two vectors. The dice similarity is defined as

$$\frac{2.|A.B|}{|A|^2 + |B|^2}$$

3.3 Best configuration

Using my custom tokenization algorithm and Dice similarity, I was able to get a MRR of approximately 0.655.