Design and Engineering of Intelligent Information Systems: HW3

Alok Kothari

October 27, 2014

1 Task

The tasks consists of implementing a simple IR system by use of similarity measures between document and query. The baseline uses Cosine Similarity and white space tokenization.

2 Error Analysis

In this section, I will analyze the errors of the basic IR system, which uses a white-space tokenizer and CosineSimilarity score as a similarity measure, between the query and the document to rank all the documents. The analysis is performed by looking at the tokenization and cosine scores of the highest rated document and the actual relevant document. I will try to analyze queries which represent the diverse errors of the system.

I will use the errors to motivate better tokenizers and stemmers in my own experiments.

2.1 Summary of errors

Type of Error	Example Queries
Tokenization/Normalization	1, 2, 5, 7,11,12, 19
Lemmatization	8, 16
Bag of Words/Stopword overlap	1, 2, 3, 14
Lack of semantics judgment inherent in CosineScore	2, 3, 6, 15, 16
In correct POS	3,12
Synonymy	9, 12, 13
Length issues	17,19
Case Issues	3,5
In correct relevant judgment	20

Query ID 1: The approach returned Document 3, as highest rated document (0.63). The high ranking was because without removing stop words, there was a high similarity between this document and the query, as there is a high overlap on stop words. This what I call error due to 'bag-of-words' overlap. The relevant document received Rank 2. I also noticed that 'Pompeii' in the document was not matched to 'Pompeii' in the query as it was not stripped off the punctuation. The question asks which volcano destroyed Pompeii, while answer documents contains this indirectly.

Query ID 2: In this case as well, tokenization like 'Jordan–one' caused the mismatch of 'Jordan' in the document and query. Thus, normalization of terms would be helpful again. Matched only at a bag-of-words level, the highest ranked document only had words similar as query, and did not provide the actual numerical answer ('largest crowd?'). Thus the semantics of the question was lost. While the relevant document got Rank 2 again (close in score), the fact that it DID have the numerical answer, but that did not give it an extra score.

Query ID 3: Almost similar reason as Q2. More word matching between query and document that was the Rank 1 document leading to high score. Missed the semantics of the question – which required a 'year' as an answer. In addition relevant document had answered the question in past tense 'purchased' which did not match 'purchase' from the question. Since the "In" is in uppercase in Question and lower case in relevant document, one more potential match is missed.

Query ID 5: The most relevant document has 0 score, as it has no match with query words due to bad normalization of words and no lower-casing. Also "Sorrow of China", has "sorrow" in uppercase, this will be a problem even if we get the tokenization correct. Also 'China's' is single token. Top ranked document while has a higher match also because of stop words.

Query ID: 6: ‘four minutes’ in query as v.s ‘four-minutes’ in relevant document.: bad normalization, no stemming/lemmatization. Actual answer word: ‘Roger Bannister’ not present in the query, hence low score.

Query ID 8: ‘biting’ in relevant document v.s. ‘bit’ in query. Stemming/lemmatization would have solved this issue.

Query ID 9: spacecraft v.s spaceship – both are almost synonyms, however this is not captured by cosine similarity. Also Missing the fact that answer required any general spaceship as v.s. manned spaceship.

Query ID 11: ‘Devils’ v.s ‘Devil’s’. bad tokenization. **Query ID 12**: Mendocino Tree is the tallest redwood in the world. as v.s ‘tall’ in document.

Query ID 13: Synonymy issue ‘deep’ in query v.s ‘depth’ in relevant document.

Query ID 14: The relevant document was just edged out because ‘the’ appeared twice in top ranked document!

Query ID 15: The cosine similarity cannot give a higher score to the place, which was hinted by ‘coldest temperature recorded in’ in the document.

Query ID 19 : This query presents a rather interesting normalization problem. The low match is because ‘New Jersey’ is represented as ‘N.J.’ in the document. Its hard to overcome such a problem! The length of the actual answer is also very long affecting its score, while the actual answer is a small part of the document. Note though the frequency vectors are length normalized, the small part of answer is still down weighted as normalizing denominator is high.

Query ID 20: I contend that the relevant document here does not provide the actual answer to the question, while the actual answer is ranked 1, as desired. “Keystone Resort is the largest ski resort in Summit County located in Keystone Colorado.” IS the right answer to the question: “What is the Keystone State?”.

3 General data flow and Design

Type System I used the given type system. The two types utilized are ‘Document’ and ‘Token’. ‘Document’ attributes are qid, relevance, token list, document text. ‘Token’ attributes are text and frequency.

Data Flow Pipeline It consists of three parts.

1. DocumentReader: Reads the documents and puts them in the CAS. Here each line is a document and it is processed accordingly.
2. DocumentVectorAnnotator: Creates the term frequency vector for the query and the documents. At this stage the term vector and document vector frequencies are raw integers. This also contains a given naive tokenizer (plain white space tokenizer). This where other tokenizers may be implemented.
3. RetrievalEvaluator: This component performs a number of tasks
   (a) Creation of normalized frequency vectors: This done by L1 AND L2 Normalization of the term frequency vectors obtained at previous stage.
   (b) Computing Similarity Scores: Computes the similarity between document and query using the normalized frequency vectors created. Some similarity scorers may not necessarily require the frequency vectors as they are set based. (e.g. Dice and Jaccard).
   (c) Computing the Rank of documents based on the similarity scores and thus computing the MRR metric.
   (d) Output writing to 'report.txt' in appropriate format.

4 Experiments

Based on the error analysis a few obvious changes to improve the score would be

1. Better tokenization and stripping of punctuation.
2. Removing stop words.
3. Using a Lemmatizer.

We will not be able to deal with synonym errors or errors which result from the fact that pure word overlap documents get higher ranks while missing a semantic understanding of the query. These will require sophisticated implementations, which perhaps make use of things like Latent Semantic Indexing, Semantic Parsing etc.

4.1 Tokenization Approaches

I have implemented tokenizers which gradually increase in sophistication to asses the contributing value of each of new feature. Here is a representative summary of all the experiments I did.

1. T0 : Basic White Space Tokenizer.
2. T0A1 : Basic White Space AND punctuation Tokenizer.
3. T0A2 : Basic White Space and punctuation Tokenizer AND lowercasing.
4. T1: Tokenizer which splits on white space, punctuation, lowercases AND removes stop words.
5. T2: Tokenizer (Stanford) which splits on white space, punctuation, lemmatizes, lowercases (without removing stop words).
6. T3: Tokenizer (Stanford) which splits on white space, punctuation, lemmatizes, lowercases (AND removes stop words).

4.2 Similarity Approaches

1. S0: Cosine Similarity.
2. S1: Jaccard Similarity: Done at a set level, without regard to frequency of the term. $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$, where A, B are query term set and document term set.
3. S2: Dice Similarity: Done at a set level, without regard to frequency of the term. $D(A,B) = \frac{2|A \cap B|}{|A|+|B|}$, where A, B are query term set and document term set.

Note that **T0,S0** is the baseline approach implemented in Task 1.

4.3 Experiment Tabulation

Line Num	Approach	MRR
1	T0 S0	0.4374
2	T0 S1	0.4416
3	T0 S2	0.6041
4	T0A1 S0	0.5125
5	T0A1 S1	0.4833
6	T0A1 S2	0.6458
7	T0A2 S0	0.5583
8	T0A2 S1	0.4583
9	T0A2 S2	0.6458
10	T1 S0	0.5250
11	T1 S1	0.5416
12	T1 S2	0.6458
13	T2 S0	**0.6625**
14	T2 S1	0.5708
15	T2 S2	0.6458
16	T3 S0	0.5958
17	T3 S1	0.6291
18	T3 S2	0.6458

4.3.1 Analysis

The slightly better tokenization, by tokenizing and stripping punctuation results in a improvement right away for all similarity measures. (refer Lines 1,2,3 v.s 4,5,6). Then lowercasing also results in slightly better results for Cosine Similarity metric. (refer Lines 4 v.s. 7). e.g. of improved rank is of Query ID 3, where "In" now matches in both query and relevant document. However score improvements are also from lucky spurious matches like 's' of U.S. and 's' from 'nation's'. e.g. in Query 17. Removal of stop words surprisingly does not yield better results for Cosine similarity metric(refer Lines 7 v.s 10), however increases the score for the set-based Jaccard metric (refer Lines 8 v.s 11). The Stanford lemmatizer (without stop words) gives further gains for both Cosine Similarity and Jaccard (refer Lines 10 v.s 13 and Lines 11 v.s 14) – in fact the cosine similarity metric gives the best score in all the experiments. This means surprisingly cosine similarity was not badly affected by stop words but more by lowercasing and lemmatization (or as earlier there are lucky spurious matches). This is confirmed by the fact that removing stop words again only increases Jaccard score (refer lines

14 v.s 17) while cosine similarity actually decreases (refer Lines 13 v.s 16).