Community Questions Answering - Question Comment Similarity with the use of Unsupervised ranking model

(4 study points project work on Text-Based Information Retrieval)
by,
Marimuthu Ananthavelu (r0652832) Monika Filipčiková (r0683254)
KU Leuven.

Summary

This project demonstrates the use of the Unsupervised model for ranking the comments with respect to a Question in the Community Questions Answering forums. The model which is used in this task is a Vector Space Model. The text pre-processing of the questions and comments is done with the help of available libraries for stop words, punctuations marks, Stemming. The relevancy between each question and all the comments were referred as vectors in Vector space and their similarities were calculated with the help of Cosine scoring. The evaluation of model is done on the development set, compared with the baseline results and they are discussed. The output prediction file is generated for the 'test_input.xml' and is submitted for evaluation. The model is developed and evaluated using Python 3.5 version.

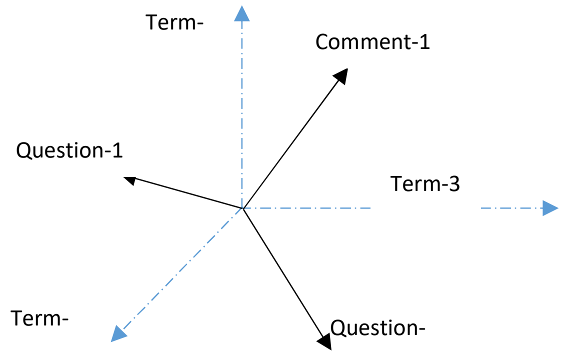
1. *Introduction to the Project task:*

Community Questions Answering forums are helpful to know the information a user wants to know. The comments in this task are the different answers for a specific question which is presented in the file with the .xml formatting. The objective of this task is to rank the very relevant answers to a question with a high score and less to the least relevant answers.

2. *Approach:*

The intuition for choosing the Vector Space Model is because of the nature of Questions and Answers forum's in which the appearance of terms occurs. A sample Question may have an information need in the form of, *'Where can I have the best coffee?', which* can have a relevant answer as: *'The best coffee is in Block-200A'.* This terms relevancy help to find the similarity between each question and comment pair.

The Vector Space model considers all the information needs and the relevant documents (comments in our case) in the Vector space. The chosen ranking model uses the same approach for finding the similarity between a question and a comment pair in each Thread in our task. An example of Vector space model representation shall be below:



Whereas,

- Terms are axes of the space,
- Comments are vectors in this space.

Fig-1: Vector space model representation

In our task, we presented the model where the Question and all the comments were represented as Vectors. These vectors are weighted and normalised to each question and document pair and the dot product of those are used for calculating the Cosine score as follows:

1. For each question, the following are calculated:

Wordset	Question					
	Term-Frequency	Term-weight frequency	Document-Frequency	Inverse document frequency	Weight	***Normalized weights***
Term-1						
Term-2						
…						

Whereas,

Question = Mix of Question Category, Subject, and Body

Wordset = Words which all are part of a single Thread (Question + all the comments in a Thread)

term Frequency (tf-raw) = Number of times a term appears in a Question which produces the count Vector for each query or comments (document in the Information retrieval context) of the vocabulary size in a Thread.

This implies a comment with 5 occurrences of the term is more relevant than a comment with 1 occurrence of the term.

term weight frequency (tf-weight) = Converting the raw number of terms into weights in logarithmic scale as below:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Calculating the weights for a raw number of term frequencies reduces the linearity to sub-linear in. For an example,

0(tf)→ 0(tf-weight), 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.

document-frequency (df)= Number of comments a term appears. The notion behind calculating this is, rare terms are more informative than frequent terms in a comment. Thus, giving high weights to the terms which are rare and the fewer weights for frequent terms (an inverse measure of informativeness).

inverse document frequency (idf) = Converted values of comments frequency using the total number of comments on a logarithmic scale.

idf = log (N / log (df))

Whereas,

N – Total number of comments

This implies, if a word appears in every comment, does have a weight of ‘0’ due to the reason of non-discriminatory value between the comments.

tf-idf weight term for query = They are the product of tf-weight and idf-weight terms.

Normalisation = The product of tf-idf are normalised at the end.

$$\boxed{\mathrm{w}_{t,d} = \log(1+\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)}$$

2. For Comments, in a similar way, they are calculated for each comment (answer) as follows;

Wordset	Comment*			
	Term-Frequency	Term-weight frequency	Weight	***Normalized weights***
Term-1				
Term-2				
…				

***There is no idf as it does not impact/not require for scoring for a comment.**

The end dot product of the normalised weights between each question and comment pair can be calculated before they are summed to find the Cosine Score.

$$\boxed{\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i}$$

which provides the similarity of the two vectors i.e. between the question (q) and comment (d) pair.

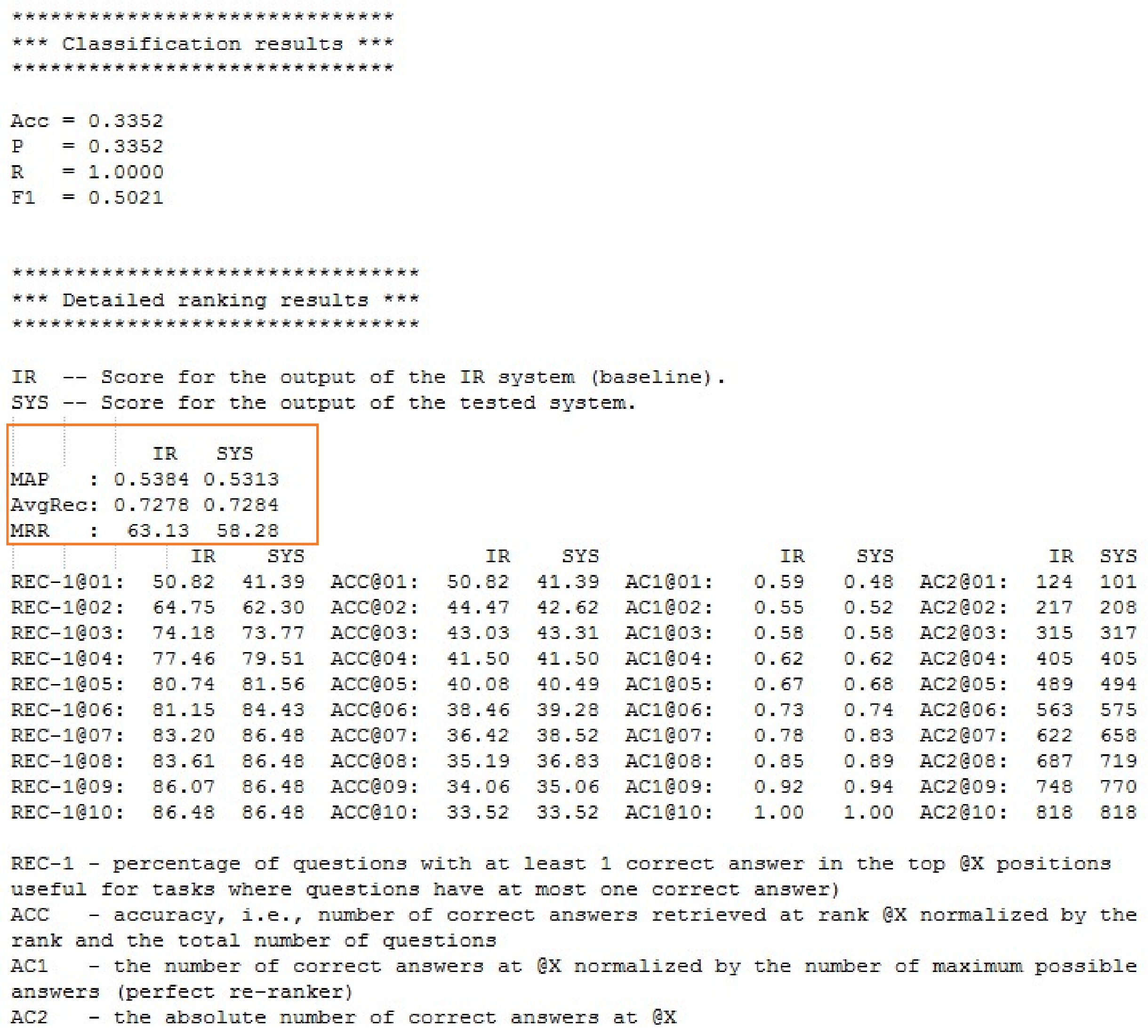
3. Model results discussion:

The scores generated from the model on development set have been evaluated with the given relevancy file for the development set. These results provide us information about how good the model is to perform for an unknown dataset. i.e. test_input.xml data with given question and comments. The predicted output file has been presented for the same as an enclosure.

The following criteria were considered as important for efficient implementation of functions and completeness:

1. The Question subject, Question body and the question Category were merged into a text as single question representing the Query vector. We merged it because we want to have at least some information about the question, in case question body is missing. This also avoids having an empty query (which will make Vector Space model make scores as Zero).
2. The questions and comments are preprocessed, because we want to remove the punctuation marks, stop words, and have a stemmed words collections within terms index.
3. The model ranks the comments as per the Cosine score for each question and comment pair. e.g. the higher the cosine score, higher the rank it is.

On the development set, the results are compared in the following way:



```
******************************
*** Classification results ***
******************************

Acc = 0.3352
P   = 0.3352
R   = 1.0000
F1  = 0.5021


********************************
*** Detailed ranking results ***
********************************

IR  -- Score for the output of the IR system (baseline).
SYS -- Score for the output of the tested system.

           IR   SYS
MAP   : 0.5384 0.5313
AvgRec: 0.7278 0.7284
MRR   :  63.13  58.28
              IR    SYS                IR    SYS                 IR    SYS              IR  SYS
REC-1@01:  50.82  41.39  ACC@01:  50.82  41.39  AC1@01:   0.59   0.48  AC2@01:  124  101
REC-1@02:  64.75  62.30  ACC@02:  44.47  42.62  AC1@02:   0.55   0.52  AC2@02:  217  208
REC-1@03:  74.18  73.77  ACC@03:  43.03  43.31  AC1@03:   0.58   0.58  AC2@03:  315  317
REC-1@04:  77.46  79.51  ACC@04:  41.50  41.50  AC1@04:   0.62   0.62  AC2@04:  405  405
REC-1@05:  80.74  81.56  ACC@05:  40.08  40.49  AC1@05:   0.67   0.68  AC2@05:  489  494
REC-1@06:  81.15  84.43  ACC@06:  38.46  39.28  AC1@06:   0.73   0.74  AC2@06:  563  575
REC-1@07:  83.20  86.48  ACC@07:  36.42  38.52  AC1@07:   0.78   0.83  AC2@07:  622  658
REC-1@08:  83.61  86.48  ACC@08:  35.19  36.83  AC1@08:   0.85   0.89  AC2@08:  687  719
REC-1@09:  86.07  86.48  ACC@09:  34.06  35.06  AC1@09:   0.92   0.94  AC2@09:  748  770
REC-1@10:  86.48  86.48  ACC@10:  33.52  33.52  AC1@10:   1.00   1.00  AC2@10:  818  818

REC-1 - percentage of questions with at least 1 correct answer in the top @X positions
useful for tasks where questions have at most one correct answer)
ACC   - accuracy, i.e., number of correct answers retrieved at rank @X normalized by the
rank and the total number of questions
AC1   - the number of correct answers at @X normalized by the number of maximum possible
answers (perfect re-ranker)
AC2   - the absolute number of correct answers at @X
```

3.1. *Functionality:*

The model is built in computing all the required information for finding the similarity using Cosine score. These functions are built into the model, which does not make use of any inbuilt library available in Python for doing the similar tasks. The functionalities are tested by creating small subsets of given training examples and in whole for ensuring the correct implementation

3.2. *Algorithmic Efficiency:*

Time Efficiency:

The implemented model comes with less or nil redundant code without making use of any inbuilt library in Python. This is also more scalable for a huge number of questions and comments with no additional efforts in implementation. This is clearly being seen and evident by having run the model overall 'test_input.xml' which took **about 10 seconds computing time.**

The time efficiency of the implemented model is O(N) whereas N – number of threads (question and comments). This helps to be more scalable with large data for getting fair ranking implementations.

Space Efficiency:

Though the terms index per thread may create the sparse matrices in places where there are completely irrelevant comments, it is likely that majority of the comments bring the similar words with respect to the question which makes all the weights of comments more competitive in overall scoring.

3.3. *Performance:*

The performance of the Vector Space model has been compared with other relevant ranking models for a similar application as below.

The performance of the model has been compared with two other SIMILAR work in implementing with the use of Vector spaces.

a) A Joint Model for Answer Sentence Ranking and Answer Extraction - Md Arafat Sultan, Vittorio Castelli‡ Radu Florian, Institute of Cognitive Science and Department of Computer Science, University of Colorado, Boulder, CO, IBM T.J. Watson Research Center, Yorktown Heights, NY. (3)
b) Finding Similar Questions in Large Question and Answer Archives - Jiwoon Jeon, W. Bruce Croft and Joon Ho Lee Center for Intelligent Information Retrieval, Computer Science Department University of Massachusetts, Amherst, MA 01003 (4)

The comparison of results is tabulated as below:

Methods descriptions	MAP Value
Joint Probabilistic Model (a)	0.84
With Language Model (b)	0.17
Vector Space Model	0.53

From the above comparison, the Implemented Vector Space Model fairly ranks the documents with the use Cosine similarities. However, the Vector space model in combination with other similar ranking models is likely to improve the performance.

Advantages of Vector Space Model:

- Vector space model uses simple computational framework for ranking
- It uses the concept of terms similarity between a Question and Comment for ranking the relevancy between the question and comment pair.

Improvements possibilities with the model:

- The semantics of natural language texts are not considered for ranking.
- Vector space model can be combined with other probabilistic models for any further improvement in the ranking process.

Conclusion:

Retrieving information from the Internet or any data source with relevancy score is difficult and time-consuming especially if such information is unstructured. With several algorithms and techniques in the information retrieval continues to be challenging in having providing good results, in this task, the Vector Space Model is expected to provide a reasonable accuracy in scoring the comments for a question. Given the simplicity of the constructed model based on the impact of terms relevancy in question and comments with cosine score, provide an indication about how best a comment scored very relevant or less relevant.

With the calculation of cosine measure, it was then used to compute the similarity measure and to determine the angle between comments vector and the question vector in a multi-dimensional space with questions and comments are points or vectors in this space. It is found that it is easier to retrieve data or information based on their similarity measures with relevancy score and produces a better ranking for the given comments.

4. Enclosures:

- Output file named 'unsupervised_rank_marimuthu_ananthavelu.txt' for the given 'test_input.xml'.
- Model

  - setup.py

  - vsm.py

  - processing.py

  - processing_word.py

5. References:

1. Course lectures
2. Manning, C., Raghaven, P. & Schütze, H. (2009). Introduction to Information Retrieval. Cambridge University Press. Chapter 18 on Matrix decompositions and latent semantic indexing.
3. http://www.aclweb.org/anthology/Q/Q16/Q16-1009.pdf
4. http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1137&context=cs_faculty_pubs