Non-factoid Question-Answering

Information Retrieval project

Michael Simaan and Fadi Yousef

Abstract

This paper explains thoroughly the methods that were used in building the Non-factoid QA system for the IR course project at the University of Haifa, given by Dr. Haggai Roitman. The project is combined of questions and their answers extracted from the Yahoo! database, nfL6. Each question has a set of relevant answers previously determined. Using Lucene 7.3.1, and python's nltk we implemented an information retrieval program, which includes query expansion, document retrieval, passage retrieval, re-rank and more to reach good results.

1. Introduction

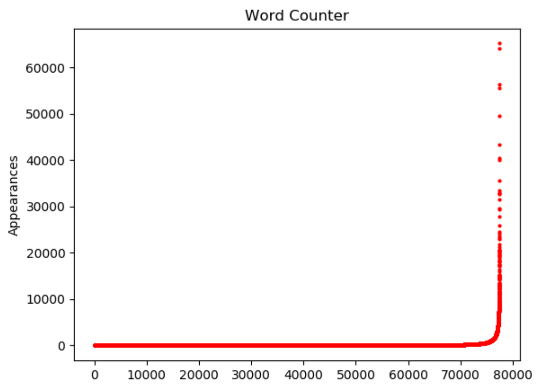
Question Answering (QA) systems aim at locating answers to natural language questions in large document collections. In this project, we were given the "Yahoo Non-Factoid Question Dataset" nfL6, taken from the CIIR website. The project is combined of around 500 thousand documents, and close to 85 thousand questions extracted from the Yahoo! database. Each question has a set of relevant answers previously decided. The objective is to retrieve 5 documents, having retrieved at least one relevant document. Using the knowledge from the IR course at the University of Haifa, in addition to the information from multiple IR research papers which revolve around Okapi BM25, passage retrieval and non-factoid question answering, we could implement a system which takes advantage of multiple utilities to reach the objective.
We've noticed that for non-factoid QA simple techniques such as stemming, tokenization and stop-words removal seem to be more effective than more sophisticated techniques such as word2vec query expansion and word2vec re-rank.
It's important to highlight that we kept a benchmark of 15 questions with their best output so far, and always compared the output of new features to that benchmark. Our approach was splitting the project into three major parts, which are data set-up, query processing, and document retrieval and re-rank.

2. Data Set-up

As our project revolved around QA, and the questions in the corpus refer to countless subjects, our first purpose was to reduce the size of the information to a set that is more manageable and better accessible.

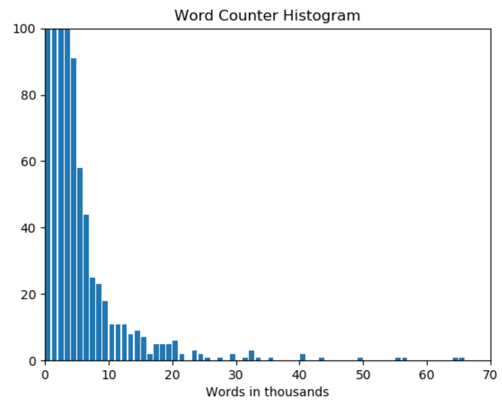
2.1. Corpus Filtering

After including the original corpus in the project, we took each document and filtered out the python's nltk set of common stop-words. The python's nltk stop-words list includes the most frequently used English words.
We also removed multiple types of ASCII punctuation, which we considered more harmful than helpful for corpus searching, such as commas, apostrophes, parentheses and more.
Furthermore, the original corpus included terms such as:
"actor/actress", "college/university" and "science/inventions". Whether the separation was a slash, backward slash, hyphen or other, it was replaced with a space to take advantage of the full text of the corpus.
Moreover, up until this point we had a corpus that was punctuation filtered, and stop-words filtered. However, after we created a word counter for the whole corpus, we discovered that words such as "know", "think", "people" and "like", appear in the corpus more than 20 thousand times. In the worst case, that would've been 20,000 different documents. After deep-thought most of these words were filtered-out to make place for other terms that will point us more carefully towards the relevant answers, and not randomly into thousands of difference results. Figure 1 below represents the number of different words in the corpus, with a counter to their appearances. You'll notice that most words are near 1 – 10,000 appearances, while a lot of words that appear more than that. We considered those terms false-positives.



2.2. Corpus Stemming and Tokenization

Once we got ourselves a filtered corpus, improving passage retrieval as mentioned in Jorg Tiedemann's paper on "Improving Passage Retrieval in Question Answering using NLP" can be done with stemming the corpus. We preferred to use the SnowballStemmer from the nltk library over the PorterStemmer as it is commonly accepted that the SnowballStemmer is an improvement of PorterStemmer. We also tokenized the filtered corpus using nltk's word tokenizer.



2.3 Analyzing and Indexing

As previously mentioned, we used two corpuses, the given corpus, and the filtered corpus. Our search is over the filtered corpus, while the documents were retrieved from the full corpus. We also used Lucene's English Analyzer to parse both corpuses, and used the default indexer for indexing the documents.

3. Query Processing

Query processing is the major part in document retrieval, as it provides the system with the relevant documents it will be reranking and outputting the results from. This step is crucial for retrieving relevant documents and should be done very carefully. Our first step after receiving the question is removing all kinds of punctuation from it except for dots and keeping only English words and numbers.

3.1. Spell-correct

To implement a precise spell-corrector, we had to create a dictionary of words. To do so, we took the extra mile, by which we extracted all the different corpus words, checked their correctness against enchant library's English dictionary, and only saved the correctly-spelled words. In addition, we added to the corpus of correct words the names of all the countries, places, first names and last names, which we've downloaded from an online github project. The script for spell-correct finds the closest words for the one given in case it is not a part of the built dictionary. We'd like to highlight that we have not corrected words that are capitalized or begin with a capital letter as they are considered names of places or people.

3.2. Syntax Tagging

To be able to dive deeper into syntactic analysis (explained in section 3.6.1.), we used nltk's position tagging to tag the words in our query. "V" was used for verbs, "J" for adjectives, "N" for nouns, and "O" for other. This is later used for query boost.

3.3. Acronyms

When dealing with queries that include words such as U.S, U.K or N.C.I.S, we noticed that the corpus usage for such words isn't homogeneous. Each query that included such a term was expanded with the same term, but without the dots.

3.4. English Verbal-Phrases

We download a list of verbal phrases and their synonyms to help with query expansion. Terms such as "get-up", "get-down", "move-away", and more can be easily removed due to filtering stop-words and frequent words (explained in section 3.5.). Queries which include verbal phrases were expanded with the synonym.

3.5. Filter Words and Stemming

After adding the verbal phrases synonyms, we filter out words that are not necessary. Such as stop-words and frequent words that have been completely removed from the filtered corpus, as they will not be found even if they were searched for.
Furthermore, since the search is done over the filtered corpus, and since the filtered corpus is stemmed, we also stemmed the filtered query using SnowballStemmer.

3.6. Query Expansion

Query expansion is one of the features that we have tried multiple ways to implement, yet at the end found out that the best way to do so, is using pseudo relevance feedback. We use Okapi's BM25 similarity, with smoothing parameters $b = 0$, and $k_1 = 0.6$ to retrieve the top 50 documents. The original Okapi BM25 similarity score equation is as follows:

$$score(D, Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i, D) \cdot (k_1 + 1)}{TF(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avg\ dl}\right)}$$

We set $b = 0$ to disregard the extra score that the length of the document gives, and retrieve only relevant ones with extra emphasis on the IDF. The BM25 final score equation used is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i, D) \cdot 1.6}{TF(q_i, D) + 0.6}$$

We give less emphasis on the Term Frequency (TF) as in Lecture 9, which discusses retrieval results, it was stated that it's better to give a bigger chunk of score for the IDF over the TF.

3.6.1. Below 3 Terms

Queries that include 1 or 2 terms have been expanded with the following process:

- We treat each **noun** in the query as an entire query. We boost it with 5, and retrieve using Okapi BM25 similarity the top 50 documents. Afterwards, we split those 50 documents to passages with windows of 50 characters and overlap ratio of 10%, then retrieve the top 30 passages taking in consideration the original document score. Finally, we create a hash of each word that appeared in the retrieved passages and a counter of how many times each word appeared, and append to the original query the word with the maximum appearances.
Note: here we take advantage of the query original tagging.

- Afterwards, we look at the original filtered stemmed query and run the same process over it, and expand with the top 3 frequent words in the retrieved passages.
Overall, we end up with at least 3 words of expansion for each query of 1 or 2 terms, and a maximum of 5 words.

3.6.2. Above 2 Terms

For queries that include 3 or more terms, we run the same process as explained above once on all the full stemmed filtered query. We boost the query, retrieve 50 documents, however the only difference is that we increase the window of passage retrieval to 60 characters, and retrieve 25 passages. The query expansion length differs from one question to another in the following way:

- If the original stemmed filtered query is 3 or 4 words, we append an extra 3.
- If it is combined of 5 or 6 words, we append an extra 2 terms.
- Queries of length 7 or more are not expanded.

3.7. Boost

Boosting the expanded query is crucial to give documents that discuss the question's topic a higher score.
The boost process we implemented is as follows:

- Original query terms are boosted with a constant 8.
- Expanded query terms are boosted with a constant 4.
- Original query sections combined of adjective then noun boosted with a constant 10. For example, "hybrid cars", "base colors" and "slow cooker".

4. Document Retrieval and Re-ranking

We retrieve the initial set of documents using Okapi BM25's similarity. Once again, our search is over the filtered corpus. However, our smoothing parameters are both 0, which means the documents are scored with IDF calculation and boost only.
After retrieval of the top 50 documents, we proceed to passage retrieval. We look at a window of 50 characters, overlap of 10%, and retrieve 25 passages only. The passages are scored according to their document score, times the sum of the term frequency for each term in the query.
We then take the top 5 retrieved passages, which originate from different documents, and as has been taught in the IR course, we give the score of the passages to the documents. At the end we return to the user the top 5 documents in which we found the best matching passages.

5. Attempts

We worked on our project for over a month and had a chance to read multiple research papers, and to try out various methods for data set-up, query processing, and document retrieval and re-ranking. Here are some of the paths we attempted but did give results better than the benchmark we held.

5.1. Data Set-up

- Building the filtered corpus as passages from the original documents with 10 terms each with overlap of 1 term.
- Training a dataset using word-embedding and Word2Vec.

5.2. Query Processing

- Retrieving an original set of results for the filtered query, then expanding gradually using the hash of the maximum appearing terms, while checking the effect of each term on the retrieved documents. Example:
For filtered query "cats eat", the word "dog" might be the most appearing word. However, appending the word "dog" to the filtered query and retrieving documents using "cats eat dog" results with more than 50% different documents than the original retrieved

documents, which we believed to be drastically drafting away from the main query topic.

- Expanding the query using the trained dataset from Word2Vec, by searching for the most similar word for each term in the query.
- Reformulating questions according to the way their written. Questions starting with “Why”, boosting results with the word “Because”. Questions starting with “When”, boosting results with the word “at”, and so on.

5.3. Document Retrieval and Re-ranking

- Reranking the documents using a trained dataset from Word2Vec
- Reranking using $\sum_{t \in q \cap p} tf(t,p) * idf(t)$
- Reranking using Saliency Weighted Semantic Network.

6. Results

After running a set of 300 questions, our results accuracy was 16.6%, while the MTRR we received for the top 5 answers was 22.9%. Overall, we saw that in the field of non-factoid question answering, the maximum results that were received are between 45% - 50% accuracy.

7. Conclusion

Non-factoid question answering is not trivial. Implementing a system that answers the nfL6 Yahoo questions requires lots of fine-grain machine learning and it is difficult to reach satisfying results with the regular methods. It is commonly accepted that the nfL6 answers are not as high quality as other datasets, which made the task a bit harder. On the other hand, it is very interesting that small changes in the query expansion or the retrieval calculations can have major effects on the result. Overall, we see that the field of non-factoid question answering is still growing and needs a lot of more decoding for us, or any other programmer and searcher to reach the results they can be satisfied with.

References

- Long Short Term Memory Networks for Non-Factoid Question Answering - http://ciir-publications.cs.umass.edu/pdf/TMP-1059.pdf
- Information Retrieval: Improving Question Answering Systems by Query Reformulation and Answer Validation - https://waset.org/publications/13824/information-retrieval-improving-question-answering-systems-by-query-reformulation-and-answer-validation
- Improving Passage Retrieval in Question Answering using NLP - https://pdfs.semanticscholar.org/54c5/0418a09fb6d62ab6b17a8cbc0eb6b87e570a.pdf
- A Survey of Community Question Answering - https://arxiv.org/pdf/1705.04009.pdf
- Short Text Similarity with Word Embeddings - https://staff.fnwi.uva.nl/m.derijke/wp-content/papercite-data/pdf/kenter-short-2015.pdf
- Dr. Haggai Ruetman’s IR Lectures.