# **BONUS - A0136134N-A0190363H-A0187836L-A0188693H**

# **1. Query Expansion**

## **1.1 Implementation**

#### **1.1.1 Using Twinword Text Analysis API (discontinued)**

We use an online API to get the synonyms for the query tokens. [Twinword](https://rapidapi.com/twinword/api/twinword-text-analysis-bundle/endpoints) is an online text analysis API that provides sentiment analysis, topic tagging, lemmatizer and other services. For our use case, we will use the API to retrieve word associations.

Step 1: Tokenize the query

Step 2: For each token, retrieve the first 5 synonyms from the API and add to the tokens list

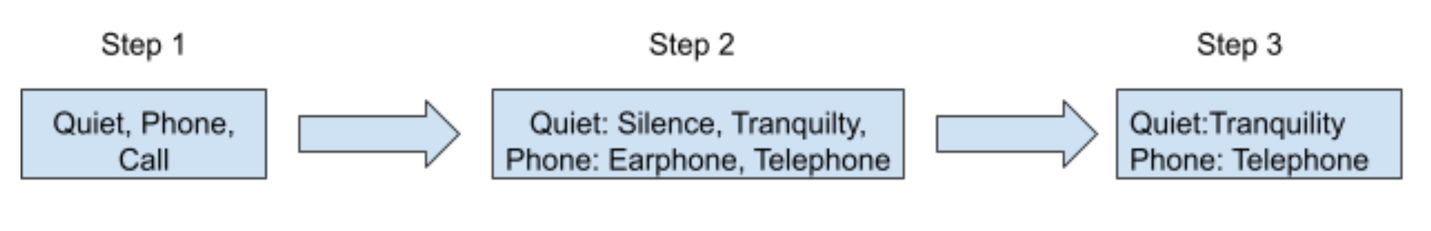
We ultimately did not use this because the results gotten from using the word association api service was very poor. This is likely because it doesn't account for Part Of Speech (POS) tagging (eg. if the word is a noun, adverb, adjective etc).

#### **1.1.2 Using Wordnet**

To obtain more accurate synonyms, we used the Wordnet library and experimented with different metrics to select the appropriate synonyms.

Step 1: Obtain words’ synonyms from Wordnet

Step 2: Filter synonyms based on metrics



## **1.2 Synonym Filtering**

For variations 1 & 2, we first perform a layer of filtering to only extract synonyms which have high enough idf. The rationale of doing this is that we wanted to prevent picking words that are too common and have too many synonyms. This is an additional filter applied on top of removing stop words.

**Variation 1:** We utilize the list of relevant documents (discontinued)

* A synonym is returned only if it exists in either of the relevant documents
* If the synonym does not appear in the relevant documents, it is a signal that the synonym is not of the right context
* Not very reliable since we may not always have relevant documents and the number of relevant documents are not many, which could make the synonyms very biased

Example

Given query term: “quiet phone call”

Given relevant documents: Doc3, Doc4, Doc 5

Question: Determine if “quiet” and “tranquility” are synonyms

Count how many times “quiet” and “silent” in Doc3, Doc4 and Doc5, which have been identified as relevant documents. If the count is more than certain constant, we assume it as relevant

**Variation 2:**  For this method, we assume we do not know which documents are relevant, so we look for co-occurrence in the entire corpus (discontinued)

* Co-occurrence rate between each term should be high
* We also took standard deviation between the scores into account
  + A synonym is ideally considered to be of the right context if it is equally likely to appear with different terms of the query
* We calculated co-occurrence rate by comparing posting list of the term in query and the

Example:

Given query term: “*quiet* phone call”

Question: determine if “*tranquility*” is a synonym of “*quiet*”.

Calculate *co-occurrence score* between “*tranquility*” and “quiet”, “*tranquility*” and “phone”, ”*tranquility*” and “call”

If co-occurrence score for all three pairs is high and co-occurrence scores are similar, then “*tranquility*” is considered a good synonym for “*quiet*”.

* For calculation of co-occurrence score, we first find the number of documents that contain both the original term and synonym
* We also calculate the difference between the intersection calculated and the length of the original query term’s posting list
  + This is to find out number of docs that contain the original query term but does not contain the synonym term
* For every document that contains both the original query term and the synonym, we give it a +1, if the document contains the original query term and not the synonym, we give it a -1.
* We then sum up the scores and we will mostly end up with a negative value
* We will then modulate these final net scores and based on the difference, we will decide if the synonym is of the right context.
* Just as variation 1, the results were very unstable and we could not find a good balance to threshold whether a synonym was relevant or not.

For the last variation, it is a much simpler approach as we do not filter the synonyms by idf.

**Variation 3:** For this method, we find synonyms that have the same context and definition as the query token

* For each query term, we determine the type of pos (part of speech) it is
  + E.g, verb, noun, etc
* We will then retrieve synonyms for each query term that have the same pos as the query term
* For the set of synonyms generated per query term, we will only include the first k synonyms where k is a threshold we set
* This particular variation seems to work best and hence we have chosen to go with this approach.

# **2. Relevance Feedback**

## **2.1 Implementation**

We perform relevance feedback using the Rocchio algorithm. There is also an option to perform pseudo relevance feedback if no relevant documents are given.

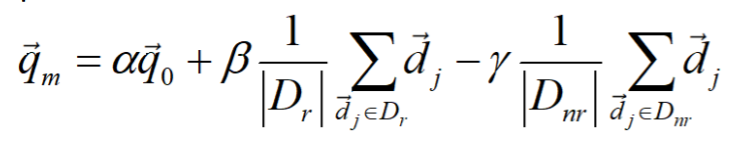


Figure 1. Rocchio formula from week 10 lecture slides where alpha, beta and gamma represent original, relevant and non-relevant query weights respectively

The algorithm is implemented as follows:

1. Obtain relevant documents from query
   1. If there are no relevant documents and pseudo relevance feedback is enabled, perform normal search and use the top 5 ranked documents as the relevant documents
2. For every relevant document, build a normalized log weight document vector
3. Build the normalized log weight query vector
4. Follow the Rocchio formula shown in Figure 1 above to add the query vector to the document vector to obtain the modified query vector.
   1. We work without non-relevant documents
   2. Thus, we set the non-relevant document weight to 0
   3. The relevant document weight is set to 0.75
   4. The original query weight is set to 1
5. The modified query vector is used in the calculation of the cosine similarity score.

### **2.2 Parameter Tuning**

The alpha and beta parameters were tuned to get a better document score. However there was no significant change to the F2, and the baseline alpha = 1 and beta = 0.75 were used.

### **2.3 Performance**

Table 1. Table of F2 scores for different scenarios using initial testing

|  |  |  |  |
| --- | --- | --- | --- |
| Query | With pseudo feedback | With normal feedback | Without Rocchio |
| Q1 | 0.0109273436296868 | 0.0109057071960298 | 0.0109057071960298 |
| Q2 | 0.373401534526854 | 0.371088861076345 | 0.371088861076345 |
| Q3 | 0.00638354631347912 | 0.0112076017176378 | 0.0112076017176378 |
| Q4 | 0.442255062944718 | 0.448412698412698 | 0.448412698412698 |
| Q5 | 0.0794887287080229 | 0.080483181350175 | 0.080483181350175 |
| Q6 | 0.182225063938619 | 0.265734265734266 | 0.265734265734266 |

As seen in Table 1, the combination of VSM with pseudo relevance feedback was only effective in improving F2 score for the highlighted query test case 2. For all other cases, there was a little reduction in F2 scores. Because we obtained the same results when we ran normal relevance feedback (only boost query vector if there are relevant documents given), it is likely that there are no relevant documents given at all in the queries. This will also explain why there was little change when doing parameter tuning on the original query and relevant document weights in [2.2 Parameter Tuning](#vhwc0i7swnm3).

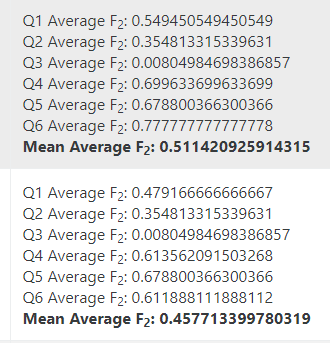


Figure 2. F2 scores after integration with other retrieval techniques

We obtained more significant results after integrating Rocchio with different retrieval techniques as shown in Figure 2 above. This indicates that the better the F2 score for baseline document ranking, the better the performance for pseudo relevance feedback. In our case, the baseline document retrieval achieves a high F2 score which is further improved with pseudo relevance feedback. Thus, we choose to include Rocchio in our implementation.

## **3. Date Relevance (discontinued)**

### **3.1 Implementation**

* Given a search year, e.g 1980, we will give documents that contain this year with a certain range an additional weightage
* We decided to look at a year range of +-4 years
* If the year found in the document is further from the original search year in terms of absolute difference, we give it a smaller weight and if nearer to the original search year, we give it a greater weight.
* We apply the logic of a normal distribution to scale the weights given to [0,1]
* We will then use this weights and combine it with the other factors
* We will then rank docs with at least a non zero date weight more important than a doc with a zero date weight, even if the overall score was better than the document with the non zero date weight
* We ultimately dropped this feature because the assumption that documents that contain the relevant years would be more important than the documents with no relevant years even if the document with no relevant years has a higher nett score might be too biased

## **4. Biword Index (discontinued)**

### **4.1 Implementation**

* We attempted to maintain a biword index to handle phrasal queries
* Biword index facilitates faster retrieval for biwords and triwords
  + To find triwords, simply merge the postings lists for the biwords that make up the triword, and check their positions to ensure that they are adjacent
* However we eventually excluded this due to the 800MB memory limit
  + Base inverted index already 784MB, insufficient space for additional biword index
* Alternative approach to use positional index was used in final submission
  + More space efficient
  + Can be generalised to to find postings lists for biwords, triwords or multiwords of any length by checking their relative positions
  + Allows for inclusion of subterms (i.e. for phrase “A B C”, we also score “A”, “B”, “C”, “A B” and “B C”) which may also be relevant to user

## **5. Proof of Concept**

### **5.1 Test query with query refinement (relevance feedback + query expansion)**

Test query: quiet phone call

After query expansion: ['call', 'phone', 'telephon', 'speech', 'sound', 'quiet', 'silenc']

Table 2. Query vector before and after query expansion and relevance feedback

|  |  |  |
| --- | --- | --- |
| Query Term | Before adjustment | After adjustment |
| call | 0.0670499076988648 | 0.10178266992454416 |
| phone | 0.2918951229604982 | 0.2918951229604982 |
| telephon | 0.2452788074423365 | 0.2758365103291561 |
| speech | 0.42615565333452 | 0.432235947638725 |
| sound | 0.30951898376452736 | 0.3203837172871688 |
| quiet | 0.5734180230670308 | 0.5790753411697126 |
| silenc | 0.4938835716354771 | 0.5002472407362438 |

Top 10 Doc ranking in descending doc score before adjustment: 2223478 2225511 6072585 2914577 6666862 2725285 2888569 2884850 4336635 2709275

Top 10 Doc ranking in descending doc score after adjustment: 2706568 2763888 2241291 6752120 2720182 2223478 2745660 2754748 2701440 2705791

We see that the document ranking changes. Let us observe the first difference in ranking. The third highest document rank is different. Using normal search, the document ranked third is 6072585.

Table 3. Document term frequency for document 6072585 and 2241291

|  |  |  |
| --- | --- | --- |
| Term | Document 6072585 tf | Document 2241291 tf |
| call | 3 | 1 |
| phone | 7 | 0 |
| telephon | 0 | 0 |
| speech | 0 | 8 |
| sound | 0 | 0 |
| quiet | 4 | 0 |
| silenc | 0 | 10 |

In Table 2 above, we see that the weightage of the call query term has increased significantly. From Table 3 above, we observe that the call term frequency for document 6072585 is greater than the term frequency for document 2241291, which causes the change in ranking.

### **5.2 Test query with date relevance**

Test query: 2016 quiet phone call

Top 10 Doc ranking in descending doc score without date authority: 2706568 2763888 2241291 6752120 2720182 2223478 2745660 2754748 2701440 2705791

Top 10 Doc ranking in descending doc score with date authority: 6554609 6146011 6019401 2770819 3744374 2701466 6525052 3797443 2768181 2763740

We see that the document rankings have changed. Now, documents that are dated 4 years to the query date are ranked higher, and come before documents that are not dated.