Pseudo Relevance Feedback

PRF/Blind Feedback uses feedback from the original retrieval, by fetching documents similar to the ones the original top k. It is Blind/Psuedo, because the top-k are assumed to be relevant. Last part of the document includes some of the questions to be answered while implementing PRF

Assumptions:

1. Initial user query is good and representative of user's information needs
2. Initial top results are pertinent have useful term representations for term harvesting
3. Term distributions appear similar for relevant documents and this distribution is different from that of irrelevant documents
4. Query Expansion base PRF doesn't negatively impact system performance
5. We have a good suppress word list (List of words which will not be added to the query e.g. Circuit)

How to choose the terms for expansion?

1. **Pure TF-IDF - Terms and Phrases a. ES - Term Vectors**

   Get Top terms from each document/passage entity with highest TF-IDF scores

   i. Group popular terms across the spans and choose common popular terms
   ii. Can be implemented using ES termvectors api
   iii. N-Grams and Phrases aren't possible without re-indexing possibly a new text field.

   b. Online TF-IDF
      i. Fit TF-IDF on the chosen text spans (entire document or key passages or both)
      ii. Get top terms and phrases
   c. Query text can be combined with in both the situations (Treating query as a document)
   d. **Note:** Doesn't carry meanings, and hence very difficult to build query centric context to choose terms

2. **Query Expansion as Boosting**
   a. Detect Phrases, but instead of query expansion - do phrase boosting

More like this

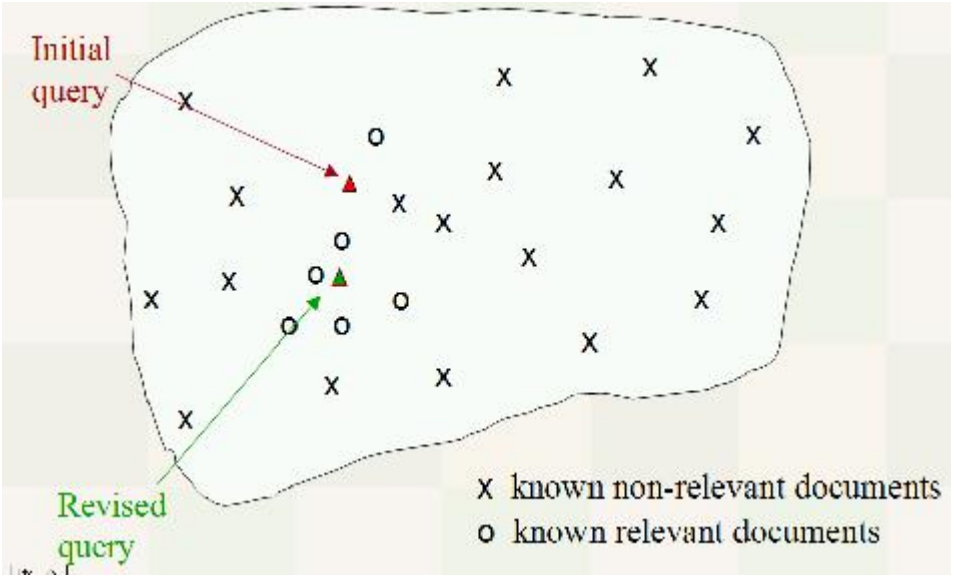
We don't choose the terms - ES does it for us.

1. *The More Like This Query finds documents that are "like" a given set of documents. In order to do so, MLT selects a set of representative terms of these input documents, forms a query using these terms, executes the query and returns the results. The ES user controls the input documents, how the terms should be selected and how the query is formed. - Taken from documentation*
3. More like this automated version of doing the TF-IDF via term vectors
4. **Like** param - Specify entities on which to do more like this. Entities could be
   a. Document Ids

b. Min-hash- Documents are hashed in a way that similar documents are more likely to produce the same hash code and are put into the same hash bucket, while dissimilar documents are more likely to be hashed into different hash buckets. This type of hashing is known as locality sensitive hashing (LSH). We get a minhash per doc and if two documents are similar then their minhashs will be similar
c. Key Passages
d. Query Text

5. **Unlike param** - The unlike parameter is used in conjunction with like in order not to select terms found in a chosen set of documents. In other words, we could ask for documents like: "Apple", but unlike: "cake crumble tree". (From documentation)
   a. More like this can be made into a dismax/boolean should query with different boost values for the document query (ids or min hash) and the text query (User'soriginal question terms). We are essentially using a combination of query and results together.
   b. Black Box as opposed to the TF-IDF methods

Semantic Retrieval - Rocchio Algorithm- DL/ML

c. 

Source NLP Staford
d. We do semantic retrieval - I.e convert everything to some vector(DRMM/ ROSS Vectors / BERT).
e. Get a query vector and vector for every
f. Do Rocchio math of generating a new query vector by moving the original vector towards the centroid of the relevant documents and away from the non relevant documents.
g. Get documents that are closest to the new query vector
h. Relevant/Irrelevant - Typically explicitly labelled, for the purpose of PRF we can assume the top 10 are relevant and 90-100 are irrelevant(?)
i. This can be implemented in combination to the above techniques, where RF (Relevance Feedback) is run instead of P (Pseudo - By assuming relevance) RF. That is we only run this algorithm when user explicitly positively or negatively interacts with the results

j. - - : Hyper parameter weights given to the query vector, centroids of relevant and non relevant documents respectively. We can control these weights

6. Find co-occurring Terms based on the original user query, i.e they use fiduciary, find out all the terms that occur along with fiduciary, and use this for expansion

PRF Questions

Background

When using ElasticSearch (ES) as the initial retriever with an index of sentences/passages/documents.

Broad Questions

1. What are some of the best ways to choose terms for query expansion?
2. What is an ideal text span/ a combination of text spans to be used for choosing expansion terms? (Since we have access to three types as mentioned in the introduction)
3. Can expansion terms based on the query terms (similar to the general query expansion techniques)? (E.g. Synonyms, Co-Occurring Terms)
4. Should we use automatic/semantic document expansion as opposed to query expansion? (Or combine both the techniques)
5. What are some of the best ways and metrics to test the performance gains?
6. Are there effective alternative testing methods for partially labelled data?

Alternatives Considered and Problems Associated with them

1. **What are some of the best ways to choose terms for query expansion?**
    1. We can consider the following alternatives (In the order of reducing complexity)
        1. Build a classifierfor choosing good terms
        2. Use Topic Modelling (LDA and other variants too) or Dominant Document Clustering
        3. Semantic/Latent Vector space based Term Expansion (Use In house trained models or OOTB SOTA language models)
        4. **Heuristics based filtering aided by TF-IDF scores (Global scores extracted using ES termvectors API or local scores using an online model on chosen text spans from the top N documents)**
2. **What is an ideal text span/ a combination of text spans to be used for choosing expansion terms?**
    1. There are four options for choosing the text -
        1. Feature snippets
        2. Passages
        3. Entire result document content
        4. Or a combination of all multiple textual spans.
3. **Should we use automatic/semantic document expansion?**
    1. Instead of doing query expansion by manually choosing terms, we can take advantage of ES's More Like This API. The input for this can be one of these alternatives
        1. Top K original result set documents
        2. MinHashof documents/passages
        3. Semantic Hashes - Based on Semantic Hash papers(This was one of the papers you have previously recommended)
    2. Rocchio Algorithm (semantic retrieval after query vector update iteration) - Proximity Based Rocchio's Model for PRF(SIGIR'12)

4. **Are there recommendations for a reasonable, practical and/or apt way to merge and/or display the result set (ER) from the Expanded Query (EQ) with the original result set (OR) from the original query (OQ)**
    1. Classic PRF - Use only the re-ranked expanded query results
    2. Train or Re-train a/the ranker to combine and sort both the result sets - E.g https://trec.nist.gov/pubs/trec13/papers/umass.novelty.hard.pdf
    3. Implement PRF-Query Expansion as Phrase Boosting
    4. User Expanded ES query, but add boosting clauses for the original terms, so as to tackle the MSM universe change problem
        1. Retrain the ~~Ranker on the new score ranges~~
    4. Present both sets of the results to the user - Possibly as relevant and related (Need Designers Input - E.g. Dice Jobs - Relevance Feedback)
    5. Is the highlighting in the result cards typically changed, to indicate the expanded terms presence?
5. **What are some of the best ways and metrics to test the performance gains?**
    1. Qualitative Assessments using SMEs (subject matter experts)
    2. Should we vet all the possible hyperparameters and experiments? Is there a recommended prioritization you can recommend?
    3. Metrics suggested : nDCG & Recall