A0136070R

E0005572@u.nus.edu

BONUS

Describe the information related to the query expansion techniques you have implemented. You may include tables / diagrams in this document.

I implemented the following query expansion methods:

1. Manual thesaurus-based method with WordNet

2. Automatic thesaurus-based method with term-term co-occurrence values

3. Semi-automatic thesaurus-based method with WordNet + term-term co-occurrence filter

Method (1) is as follows:

- For each lemma in the query, get the synsets from WordNet

- Get the lemmas for these synsets and remove duplicates

- Add all these lemmas to the original query

- In my opinion, we should restrict the number / percentage of new lemmas to be added, but

I have not experimented to get the best number because I am currently not able to assess

the effectiveness of my system

- However, this number / percentage ought to be small, so as to balance expansion and drifting

Method (2) is as follows:

-

Method (3) is an upgrade of the manual thesaurus-based method, inspired by method (2),

because in method (2), there are many parameters for us to decide whether a new term should

be part of the query expansion

Method(3) is as follows:

- Follow the first 2 steps of method (1)

- Before adding the synset lemmas blindly, fetch the nltk.Text of the top k relevant documents

from the index

- k ought to be small (in my program, I did not set k)

- This means that 1 round of retrieval should be done before expanding the query

- Get the union of terms across the top k documents that co-occur with each query lemma

- Intersect the co-occurred terms with the synonyms from WordNet

- Add the intersection minus (terms in the original query) to the original query

- As before, we need to do the minus-ing because terms in the original query might re-appear in the

synonym / co-occurrence sets across different documents that are relevant.

The increase in runtime by the query expansion methods are dependent on the various parameters inside

each of these methods.

For method (1), the expected runtime increases linearly with the total number of terms in the

expanded query. Let O(x) be the time complexity of getting the cosine similarities between 1 term

and every document in the collection in sorted order, where x is some variable that models.

the complexity of my retrieval pipeline (i.e. boolean retrieval then vector space retrieval).

Then for an original query with j terms and a query expansion of k terms, the time complexity of the

retrieval process is O((j+k)\*x).

For method (2) and (3), there is a need to first retrieve the most relevant documents for the original

query before expanding the query. (Otherwise, we do not know which documents to get the co-occurrence

values from.)

As such, using the same set of variables defined above, the time complexity of method (2) and (3) are

O(j\*x + (j+k)\*x + y) where y is a constant that differentiates between method (2) and (3), since

these two methods process co-occurrence differently.

The increase in space complexity by the query expansion methods are as follows:

For method (1), we need to store WordNet. The WordNet downloaded from the Princeton University website

is about 40 MB.

For method (2), we need to store either the word\_tokenize-ed text or the nltk.Text version of every

document. The nltk.Text of 100 documents is already occupying 7 MB. The expected total additional

space usage would be similar to that of the original CSV file (about 700 MB).

For method (3), since both WordNet and word\_tokenize-ed / nltk.Text resources are used, the increase in

space usage is the sum of additional space usages by method (1) and (2).

In conclusion, query expansion increases the complexity of the retrieval. The more the expected precision / robustness of the query expansion method, the more the resources used. My personal inclination for query expansion would be the following:

- For long queries, skip query expansion (need to define what is "long")

- Should further limit the total number of terms in the query expansion (e.g. 1 expansion term per query term)

- Should use a more niche thesaurus (i.e. relevant to law) depending on the domain of the collection