**Information Retrieval HW4**

By: A0188307X, A0204208U, A0219963R, A0248149W

For the implemented query expansion methods, the performance metric we used are the Mean Average F2 and Mean Average Precision up to 3 recall levels (We are limited by the lack of relevance judgements provided).

OUR DATA:

Freetext without query expansion (TF-IDF Baseline)

Q1:

Average AF2: 0.01922084489114802

Average MAP: 0.004197267248018752

Q2:

Average AF2: 0.3354357798165138

Average MAP: 0.3800505050505051

Q3:

Average AF2: 0.30277600170502983

Average MAP: 0.34898798228969

**Search w query expansion** **(word2vec + wordnet)**

Q1:

Average AF2: 0.02880794034091946

Average MAP: 0.0066229599218253455

Q2:

Average AF2: 0.3338675213675214

Average MAP: 0.3796728971962617

Q3:

Average AF2: 0.3979925303454715

Average MAP: 0.47222222222222215

**Freetext w query expansion (word2vec + wordnet)**

Q1

Average AF2: 0.01922084489114802

Average MAP: 0.004197267248018752

Q2

Average AF2: 0.3354357798165138

Average MAP: 0.3800505050505051

Q3

Average AF2: 0.30277600170502983

Average MAP: 0.34898798228969

**Query Refinement Documentations**

In this assignment we attempted to implement Query Expansion with different methods to perform query refinement. In the following paragraphs, we will discuss the methods we attempted for the query expansion and discuss the reasons why we did not implement it and the reasons behind possible low performance.

We will also be treating all our Boolean queries as FreeText queries due to the seemingly better performance during our testing.

Details of the Experiments can be found in:

<https://colab.research.google.com/drive/10oYY8Ko4V4RYfER0v2R571x9Umr-9cXd>

**Query Expansion with Wordnet and Lesk (Implemented)**

In order to create “good” synonyms using Wordnet, POS tags should be provided along with the text we want to find synonyms for in order to consider the contexts around the word and its corresponding definition. Hence, we have attempted to use both the NLTK.pos\_tag() method and NLTK.wsd.lesk() method. Since the tags returned from pos\_tag method are in the NLTK format (which is different from the styles/format Wordnet is using), which requires manual conversion method to convert tags from NLTK format to Wordnet that could reduce the accuracy of the synonyms returned. On the other hand, Lesk directly returns Sysnet object that we can use to find the lemmas / synonyms of the word directly. From experiments, we found that synonyms created by Lesk generally fits better in the context compared to the synonyms created using NLTK.pos\_tag, and outperforms other models mentioned below.

**quiet phone call:**

{'hush', 'quiesce', 'quiet', 'pipe\_down', 'quieten', 'quiet\_down'}

{'telephone\_set', 'telephone', 'phone'}

{'call'}

**good grades trade scandals**

{'well', 'good'}

{'range', 'rate', 'place', 'grade', 'order', 'rank'}

{'trade', 'merchandise'}

{'outrage', 'scandal'}

**fertility treatment damages**

{'prolificacy', 'fertility', 'richness', 'rankness'}

{'treatment'}

{'wrong', 'damage', 'legal\_injury'}

**Query Expansion with Word2Vec (Implemented)**

After the experimentation with different pretrained model (which will be explained below), we think that one of the problems these pretrained models have is that they are not particularly “designed” or trained using legal documents/context. Hence, we have decided to train our own word2vec model on the provided csv to find most similar words and synonyms from a given word, by providing us similar words that appears in similar context as the word we provided. Currently we only obtain the top 5 similar words and used it in our query expansion (we do not use all of it – which would be explained in the **Improvement** Section)

Example. Quiet phone call: (Format is in [word, similarity\_score])

[[["happi",0.564520537853241],["sob",0.5491302013397217],["forev",0.5476323366165161],["lone",0.5476160049438477],["quietli",0.5444419384002686]],

[["telephon",0.841902494430542],["handphon",0.7511388063430786],["pager",0.657568097114563],["mobil",0.6022642850875854],["handset",0.5968540906906128]],

[["diall",0.4705151617527008,["ask",0.40289705991744995],["ring",0.38566291332244873],["contact",0.3798167407512665]]]

**Limitations of Word2Vec**

One of the shortcomings of the word2vec model is that it does not consider the context surrounding the word or consider the positions/pos-tagging of the word. However since the model is particularly trained on the corpus and observed the co-occurrences of words in the corpus, it could explain why word2vec model could generated more suitable synonyms compared to models below. Also, it purely considers the occurrences of the words rather than the exact meanings of the words, hence as we can see in the above example, “quiet” is similar to “happy” according to the word2vec model as the words appears in similar context with similar words.

On the other hand, due to the limitation of the size of submission and the lack of explanations in package bundling requirement (we would have to include numpy, genism and cython in order to allow our search function to load the model, which would drastically increase the size of our submission since there is no pip module in the Tembusu Cluster), we have changed bundling word2vec-related library with our submission to deploying on Heroku and serving our custom model as an API to answer our synonyms query. Considering the time required to load the model at the start for our search.py, sending requests and waiting responses would not be too slow in this case and would still fall under the time limitation of one minute.

**Improvements**

Since some of the synonyms provided by the word2vec or the wordnet model can be inaccurate, especially for word2vec model with lower similarity score, hence we have decided to take the first suggested word from both model (we would have 3 words instead 1 word for query), and perform intersections of the synonyms provided by word2vec and wordnet, if the suggested synonyms appears in both model, then we can quite confidently say that the synonym would be a good suggestion.

**Conclusion**

With the implementation of Query expansion, both MAF2 and MAP of Query Expansion with was still lower than Baseline’s score. This could be due to the short length nature of the query provided hence Lesk would not be able to be confident enough to provide the best POS tag for each token in the query, and also due to the shortcomings of word2vec as mentioned above. But it is hard to conclude whether if the performance of query expansion is indeed worse than our baseline system without any query refinement technique due to the lack of actual relevance judgements that we can compare with.

We will still keep the implementation of Query Expansion within our search engine since it still performs pretty well within the Competition framework.

**Other Methods attempted for Query Expansion**

For the following models discussed, we did not include and used in our searching pipeline as the synonyms returned by the models are already not good enough to spend more time changing the pipeline and test with MAF2 and MAP.

**Using pretrained Glove model (glove-wiki-gigaword-50):**

We attempted to use a pretrained Glove model trained on Wikipedia to create synonyms for each word in a sentence. However, the terms returned were not as good as expected, for example terms returned for grades include “kindergarten” and “pre-kindergarten”, which are considered quite unrelated to the topic especially with regards to legal documents. This could be due to the fact that the model was trained on Wikipedia articles which would affect its ability to provide us good synonyms in the legal context, while the model does not take account for the sentences / POS tag of the word and the context. The libraries requirement in order to use Glove is also large enough to stop us from any further experiments (as we either have to include the model in the submission or uses gensim.downloader to download the related files)

**Question: quiet phone call**

Synonym Expansions:

quiet --> ['calm', 'staying', 'seemed']

phone --> ['telephone', 'phones', 'cellphone']

Expanded Question: *quiet phone call phones seemed cellphone telephone staying*

**Question: good grades trade scandals**

Synonym Expansions:

good --> ['better', 'really', 'always']

grades --> ['grade', 'kindergarten', 'pre-kindergarten']

trade --> ['economic', 'export', 'commerce']

scandals --> ['scandal', 'revelations', 'troubles']

Expanded Question: *good grades trade scandals better scandal troubles always really export grade economic revelations kindergarten*

**Question: fertility treatment damages**

Synonym Expansions:

fertility --> ['reproductive', 'fetal', 'circumcision']

treatment --> ['treating', 'patients', 'treatments']

damages --> ['compensation', 'wrongful', 'unspecified']

Expanded Question: *fertility treatment damages claim treating fetal deny patients claiming wrongful circumcision compensation treatments unspecified any*

**Using pretrained BERT model (nlpaueb/legal-bert-base-uncased): (Contextual Synonym Query)**

For each word in the sentence, we will mask out the word from the sentence and use the model to predict what word should be filled in using the pretrained model that was trained using legal documents/data, to overcome the potential problems of the previous Glove model we have attempted to use. Unfortunately, it could be due to the short nature of the query, the masker was not able to “get the sense” of the context and provide good suggestions for synonyms. The results we got from the suggestions are mostly “newline” / \n character which does not help in our case. The libraries requirement in order to use BERT is also large enough to stop us from any further experiments (Include downloading the BERT model and huggingface transformer library)

**Using NLPAug:**

NLPAug is also used to experiment creating synonyms with predefined model like “wordnet” and attempting contextual synonym query with “bert-base-uncased”, along with different methods of data augmentation such as “Inserting” words vs “Substituting words”. However “Inserting” words do not provide any meaningful contribution to optimize query as it can insert common “filler” words that could potentially make other documents to be ranked higher while “substitute” method is outperformed by “wordnet with Lesk” and “word2vec” trained on the original corpus.

**Augmented text with “Insert” method using “bert-base-uncased” (5 times)**

* ['quiet a phone call', 'more good gold grades trade scandals', 'alleged fertility treatment damages civil]
* ['quiet phone conversation call', 'good for grades trade secrets scandals', 'fertility care treatment damages rights]
* ['quiet... phone call', 'good grades from trade fairs scandals', 'and fertility treatment damages fraud]
* ['the quiet phone call', 'good financial grades trade abuse scandals', 'fertility health treatment damages benefit]
* ['quiet with phone call', 'earning good grades trade competition scandals', 'see fertility claims treatment damages’]

**Example of wordnet substitution with NLPAug:**

['practiced grade trade scandals', 'good form trade dirt', 'skillful class trade scandals']