COMP30018 Project 2

Jithya Nanayakkara, Student No. 342973

1. Introduction

This report explores the outcomes of implementing three add-ons to a monolingual IR system called Spydr, by attempting to identify to what extent these add-ons affect the retrieval effectiveness of the system. The retrieval effectiveness of the system will be measured using mean average precision (MAP).

The document collection used to test the system is a collection of forum threads from Ancestry.com, and the queries used are real world queries from users of the site.

2. The Spydr IR System

Spydr is a very simple IR system written in Python that is based on the vector space model and uses a TF-IDF style term weighting function. It does not do any processing beyond splitting the document to index into a collection of tokens (split according to whitespace) and stripping the tokens of mark up and punctuation.

3. Add-ons

3.1. Query Expansion Using a Thesaurus

Augmenting a query with terms related to every term in the query by using a thesaurus, is a form of query expansion. The terms in the thesaurus maybe synonyms or terms associated with the topic of a query term (Manning et al, 2008). For example, a synonym for car would be ‘automobile’ and a related term would be ‘petrol’.

It was noticed that geographical locations were present in many of the queries used to search the document collection, such as ‘jarvis winn lambert; st louis, missouri’. It was reasoned that if documents contained the terms ‘st louis’ and ‘missouri’, it was likely they would also contain the terms ‘usa’ and ‘america’, which are the different terms used to refer to the country Missouri resides in.

Therefore a thesaurus was built to expand queries with geographical locations in them. If a query contained “kyary pamyu; kyoto", it would be expanded to "kyary pamyu kyoto japan". It is hoped that this would increase precision.

A custom thesaurus was built using data obtained from the geonamescache Python library (http://pypi.python.org/pypi/geonamescache), which in turn was built using http://www.geonames.org/ as a source.

All the cities in the geonamescache JSON file were mapped to their respective countries in the thesaurus, i.e. a city name such as ‘Lahore’ would be mapped to ‘Pakistan’. Additionally, the common synonyms used to describe USA (America, United, States) and the UK (Britain, United, Kingdom, England) were manually added.

Furthermore, every state in the USA was mapped to synonyms of USA. It was also noticed that users sometimes used acronyms for US states, so these acronyms were included in the thesaurus and are mapped to the synonyms of USA and include the full name of the state. For e.g., ‘NY’ would be mapped to ‘America’, ‘United’, ‘States’, ‘New’, ‘York’. In testing of the add-on, including state acronyms improved its results very slightly.

It should be noted that all the words in the thesaurus are in lower case. While it may be questionable to expand an ambiguous query word such as ‘states’ to all the different ways of referring to USA, inspection of the queries found that the general pattern the queries follow is having names of people and places. Therefore the likelihood that those words used were not referring to a location is low. However in the thesaurus the word ‘united’ is not mapped to any other words, as it could refer to either the United Kingdom or the United States.

There are however many issues with the construction of the thesaurus. It is far from comprehensive, as it is only limited to the data provided in the geonamescache library. A more comprehensive thesaurus would have included states and provinces from different countries around the world (such as Australia for example). Also there were issues with how cities and countries with names consisting of multiple words

were handled. For example, the Federated States of Micronesia is often referred to as simply 'Micronesia', however the geonamescache data refers to it formally by its full name. Expanding a query to 'federated states of micronesia' would be adding noise to the query. While some effort was made to remove stop words such as 'Republic', 'Northern', 'Southern', 'The', 'Of', not every possible case was covered. Also when a city consists of multiple words, such as 'San Salvador', it would be split and each word would be mapped to 'El' and 'Salvador' (its country). This is not very accurate, as 'San' could belong to various cities beginning with the same name such as San Francisco. In fact, if there exists cities with the same name, the script for building the thesaurus would simply overwrite the country it should map to – so if San Francisco is the last city beginning with the word 'San' the script came across, 'San' would be mapped to the synonyms of the USA.

While query expansion using thesauri intuitively should improve precision, it is often not the case, as it has the effect of often increasing recall but significantly degrading precision (Manning et al, 2008). Its effectiveness is highly dependent on how good its thesaurus is. But this add-on was implemented in any case to see whether in this restricted context, expanding the queries would improve retrieval effectiveness.

3.2 Token Processing

Additional processing was done on the tokens generated by Spydr. Given Spydr simply strips the tokens of punctuation and mark up, there was room for additional processing. The processing used was:

- Splitting of hyphenated words into separate tokens.
- Making all the tokens lowercase.
- Stemming the tokens using the Porter algorithm.
- Removal of stop words.

Each processing component of the add-on was tested individually and found to improve performance. Both the document and query tokens were processed.

Stop words are extremely common words that have little value in selecting documents that match the user's information need (Manning et al, 2008). Therefore removing stop words is a way of increasing the precision of IR systems.

Initially, the list of stop words consisted of 174 words and were taken from http://www.ranks.nl/resources/stopwords.html However a general trend in IR systems is to have very small stop words list or have no stop word removal at all (Manning et al, 2008). By running several tests, it was found that removing stop words with the large list was more effective than not removing them. However by truncating the stop word list to twenty six words, the effectiveness of retrieval was further improved.

Lowering the case of tokens is a very common strategy to improve results. For example, if a user were to run a query for 'ferrari', it would not match all occurrences of the word 'Ferrari' due to the differing case. Similarly, the first letter of a sentence is generally capitalized in documents; therefore those words would not be matched to a query whose case does not match. There is a danger in uniformly lowering cases of tokens as there is a semantic difference between 'CAT' and 'cat' or a query for 'General Motors' which specifically refers to a company (Manning et al, 2008). While a more intelligent strategy of selecting which tokens to lower case can be implemented, such as only lowering the case of words that are at the beginning of a sentence, it is not the most practical solution as users often cannot be bothered to use proper casing (Manning et al, 2008).

Stemming is the process of chopping off the ends of words to equate related words to a base word. For example, 'caress' and 'caresses' would be stemmed to 'caresses'. The effectiveness of stemming depends on the queries performed, as it can often increase recall while reducing precision (Manning et al, 2008). However in this context, stemming was found to improve the effectiveness of retrieval in Spydr.

3.3 Pseudo Relevance Feedback

The basic idea behind relevance feedback is to implement multiple passes of information retrieval and in each pass, refine the query based on the results of the previous pass (Grossman and Frieder, 2004). Pseudo relevance feedback is an automatic way of doing so, by assuming the top k ranked documents retrieved in the first pass are the most relevant and then perform relevance feedback (Manning et al, 2008).

In this component of the add-on, the formula used

for feedback is:

$$Q' = Q + R - S$$

where:

Q' = the new query
Q = the original query
R = top k ranked relevant documents
S = the top ranked non-relevant document

Pseudo relevance feedback utilizes query term weighting to calculate the similarity with documents and the original formula, obtained from Grossman and Frieder (2008), decreases the weights of the query terms. However as query weighting isn't done in the Spydr IR system, a simplified form if it is used (as shown in the above equation), which works as so:

- The terms found in both the original query Q and top ranked k relevant documents are added to Q'.
- The terms in the top k ranked relevant documents R but not in the top ranked non-relevant document S are added to Q'.

The value of k (i.e. the number of top ranked documents deemed relevant) and the number of iterations of feedback to perform has been subject to a fair amount of research (Grossman and Frieder, 2004). The value of k used was arbitrarily chosen to be a quarter of the number of relevant documents returned and only one iteration of feedback was performed.

The danger of such a technique is that if the top ranked documents returned are not relevant or only somewhat relevant, performing feedback on those documents assuming they are relevant will further degrade the results of the query (Manning et al, 2008).

4. Results

Add-on	MAP score
Baseline (vbjdn-run01)	**0.1931**
Token Processing (vbjdn-run02)	**0.5437**
Query Expansion (vbjdn-run03)	**0.1693**
Pseudo Relevance Feedback (vbjdn-run04)	**0.1818**
Token Processing + Query Expansion (vbjdn-run05)	**0.5411**
Token Processing + Pseudo Relevance Feedback (vbjdn-run06)	**0.5427**
Query Expansion + Pseudo Relevance Feedback (vbjdn-run07)	**0.1823**
All (vbjdn-run08)	**0.4950**

5. Discussion

It can be seen that all the add-ons individually produce worse results relative to the baseline, with the exception of token processing. Combining the various add-ons failed to produce results better than only using the token processing add-on.

The results of the query expansion using a thesaurus seem to confirm research that finds its effectiveness mediocre. In this particular case, its poor results can be attributed to a thesaurus that was not constructed very well, which has been detailed in section 3.2. Another reason is that all the terms in the thesaurus are in lower case and the document index is not lower cased in Spydr, therefore augmenting the query with lower case query terms when the document collection is most likely to contain correct casing of proper nouns would result in a mismatch. Upon inspecting some of the queries, it was also noticed that some queries contained names of people that were the same name of a city. For example, ‘vicente’ is also the name of a city in El Salvador, so that query would have been incorrectly expanded to include the country name.

However, it simply may be that adding the terms simply added noise to the query and decreased the precision of the retrieval. This maybe the case as combining query expansion with the other add-ons failed to improve retrieval effectiveness.

The poor performance of the pseudo relevance feedback add-on can be attributed to its simplified implementation. If query term weighting was used, where weights were determined according to how frequently the query word occurred in the top ranked documents, it is possible a better result could have been obtained. In the current implementation, a query can become huge as it contains all the terms in the top ranked relevant documents (minus those that occurred in the top ranked non-relevant document – unless they were query terms as well). This would add noise to the system as all the terms in the query would be treated equally.

The most effective add-on by far was the token processing by itself.

6. Further Improvements

The token processing add-on can be further improved by using a more intelligent list of stop words. Rather than using a list of common stop words, a stop word list can be generated by collecting statistics for words that frequently occur across the documents in the collection. Words that both frequently occur within a document and are common to most documents across the collection can be added to the stop word list.

The document index can also be improved by adding commonly used phrases. For example, "New York" can be considered to be a single term, instead of splitting it into two terms. Queries for 'New York' would more likely retrieve documents that are more specifically about New York. Selecting which phrases to index is the difficult aspect of this technique. Typically, statistics on commonly used phrases in queries by users are collected and used. As a starting point for this document collection in particular however, it can be used for geographical locations that have multiple words.
This above technique can be used together with a positional index as well, which checks whether the position of terms in a document are compatible with the position of the terms in the query. These techniques would improve the accuracy of the documents retrieved (Manning et al, 2008).

7. References

Manning, Christopher D., Prabhakar Raghavan, and Hinrich, Schutze. 2008. *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press.

Grossman, David A., and Frieder, Ophir. 2004. *Information Retrieval: Algorithms and Heuristics,* Dordrecht, The Netherlands: Springer.