

Introduction

This report aims to evaluate the performance of vector space model based Information Retrieval (IR) with different term manipulation strategies.

Our standard test collection is from CAMA (Communications of the Association for Computing Machinery), it includes a document collection D , a query set Q , and a set of 'golden' standard for each query. The task of our IR system is finding the top 10 most relevant documents to a given query.

The Vector Space Model

The vector space model maps each document $d \in D$ to a normalized vector $\vec{v}(d) \in V$. Each dimension i of V corresponds to a unique word w_i in the document collection. We treat query $q \in Q$ as a special document, and use the same map to map it into a vector $\vec{v}(q) \in V$. The similarity between a documents d and a query q is calculated by the inner product:

$$(\vec{v}(d), \vec{v}(q)) = \sum_i v_i(d)v_i(q)$$

Let's make some notation. The number of documents in D is denoted by $\|D\|$. Using tf_i to denote the number of times w_i appears in d , and idf_i to denote the number of documents that contains w_i .

Here, we consider three different document-to-vector maps:

1. **binary**: $\tilde{v}_i(d) = \begin{cases} 1, & w_i \text{ appears in } d \\ 0, & \text{else} \end{cases}$
2. **tf**: $\tilde{v}_i(d) = tf_i$
3. **tfidf**: $\tilde{v}_i(d) = \frac{tf_i}{idf_i}$, where $idf_i = \log \frac{\|D\|}{df_i}$ is called inverted document frequency.

For all three cases vector need to be normalized: $v_i(d) = \tilde{v}_i(d)/\|\tilde{v}\|$. There are two more pre-processing options in our model:

1. **-p**: Ignore all the words in the 'stop list'
2. **-s**: Use the same index for different words with the same word stem

Implementation

My work is completing the `Retrieve` class in `my_retriever.py` file. The most important function `forQuery()` takes a specific query as the input argument. This function returns the index of the top-ten most relevant documents, by using the model which implements one of the three document-to-vector maps `binary`, `tf` and `tfidf`. Two preprocessing choices `stemming` and `stoplist` can be

selected before calculating the similarity between documents and queries.

In order to increase efficiency,

- The coordinates are stored in a two-dimension float numpy array. Each row in this array is a document vector, and the columns represent the dimensions. This array is used to compute similarity for all queries.
- Indexing the keys of index from 0 and store it into `self.words`, so that truncate the coordinate array before inner production.

Result

Table 1 The Evaluation of IR System with Different Term Manipulation Strategies

weight	-s	-p	Real Retrial	precision	recall	f-measure
binary			44	0.07	0.06	0.06
	✓		82	0.13	0.10	0.11
		✓	58	0.09	0.07	0.08
	✓	✓	105	0.16	0.13	0.15
tf			50	0.08	0.06	0.07
	✓		107	0.17	0.13	0.15
		✓	72	0.11	0.09	0.10
	✓	✓	123	0.19	0.15	0.17
tfidf			128	0.20	0.16	0.18
	✓		134	0.21	0.17	0.19
		✓	171	0.27	0.21	0.24
	✓	✓	176	0.28	0.22	0.25

The record running time for each retrieval command is less than 1 second.

Discussion

Use of stop list and stemming

From the experimental result, the use of `stoplist` and `stemming` increases the accuracy of the retrieval result.

As Rijsbergen (1979) states, the most useful words for retrieval are neither the most frequent terms in a document nor the least ones. The most common word in either document or query should be something like 'and', 'what'. These words by themselves don't give any information about the document or query's topic, especially our queries might be a descriptive paragraph. In this case, we don't expect that their presence or not will contribute to the computation of similarity. When putting

this kind of word into a stop list and ignoring them, the problem solves.

At the same time, we think that the topic represented by a word is often associated only with the word stem of the word. For example, there are several 'farmer' in a document and 'farm' occurs in a query, so they are probably related. With the help of stemming, 'farmer' and 'farm' will use the same index, and contribute to similarity. However, we can not judge the semantic similarity of words by stemming word simply. For example, if there is a lot of 'farm' in a document, and 'agriculture' in query, they are probably related.

Choice of document to vector mapping

Compare among the performance of the three document-to-vector maps: $\text{binary} < \text{tf} < \text{tfidf}$.

The reason why performance of tf is better than binary is that tf includes the information not only of whether a word appears or not, but also of the number of times it appears. Intuitively, assume there are two equally long documents, 'farm' occurs 10 times in the first document, while only one 'farm' in the second document, then the first document is more likely to be related to a query which contains 'farm'. However, binary maps the components of both articles corresponding to 'farm' to the same value. This is why binary does not retrieval information well, although sometimes the documents are short.

When tf maps the document, all terms are considered equally important. However, the importance of different words is different (Manning, Raghavan and Schtze, 2008). Assume there are two keywords in a query 'AI' and 'AlphaGo', and the system retrived two document. 'AI' occurs in the first document 9 times while 'AlphaGo' occurs once. In the second document, there are 9 'AlphaGo' and one 'AI'. If we tf maps the documents, we get the same similarity score. However, the second document is more likely related to our query because it has more number of rarer word 'Alpha'. Therefore, the use of IDF increase the value of a rare term.

Conclusion

The optimal retrieval result is given by the tfidf model with stoplist and stemming, and its f-measure is 0.25.

Reference

- [1] Christopher, DM., Raghavan, P. and Schtze, H. (2008). *Intro to Information Retrieval*. Cambridge University Press.
- [2] Rijsbergen, V. (1979). [online] Dcs.gla.ac.uk. Available at: <http://www.dcs.gla.ac.uk/Keith/Preface.html>.