INFORMATION RETRIEVAL SYSTEM & THE BEST FITTING ALGORITHM

[TOP SECRET]

Ryan Zink
Matt Versaggi

Abstract

This report is about the design and development of the Information Retrieval system that we created, and the comparisons of different relevance computation algorithms and their effects on the results retrieved from various queries. Three different relevance computation algorithms were implemented for comparison: Cosine similarity, Okapi, and Pivoted normalization weighting. For each one, the same set of 10 queries were used and the results were manually inspected for relevance. On average, the pivoted normalization weighting algorithm retrieved the most relevant documents out of the three. Thus we conclude that in most cases, this is the most effective formula for calculating document relevance.

Information Retrieval System & The Best Fitting Algorithm

By Matt Versaggi, Ryan Zink

Introduction

For this program there were multiple things to consider as far as how the data was going to be stored and the best way to calculate the weight of a document based on a given query. We started out by determining how exactly we wanted to store the data using hash tables. In order to get the term dictionary after cycling through all documents we decided to have our Document objects contain a hash table that gets passed to each successive document which would add a word if it was the first time it has been seen through all previous documents. This data is known as the clone data in our program and was especially important when calculating the total frequency of a term in all documents. Having the clone data be passed to each document allowed us to only have to run through the documents once to get all the information we needed about the terms in a specific document and the information about that term in all the documents. We set up our IR system to easily change the way in which it weighs relevant documents using the three different methods discussed in class. This allowed us to conclude which algorithm solved our query and information needs best.

System Design and Implementation

The IR system is designed in such a way that it can go through a new file of documents once and have everything it needs in order to calculate query relevance's. Inside our main data structure, BigD, we have the four data structures. The most important ones are the hash table of terms throughout the document, the hash table of inverse document frequencies of each term and the array list of documents. The documents are never sorted in order to keep track of weights to allow for multiple queries in the same run of the program. The query object contains a hash table of the query's term weights, an array list of points where x is the index of the document in BigD's array list and y is the correlation result of that document for the query. We sorted our list of 2D points based on the y value and simply grabbed our document from the original list of documents using x as our index. The final key structure for our IR system was the Document object, which went through numerous changes throughout the design process. The biggest problem we faced was our allocation of data inside the document. We allocated a new clone for each document that used up memory extremely fast which influenced us to have a single clone that all documents use. It was also decided that for each document it should only contain a hash table of terms inside that document to save space. The document was simple after that, only having a Person object with all the original stuff like the id number, person's first and last name and so on for the document. Our last feature of the IR system allowed us to use multiple relevance calculations to determine the best algorithm for a set of given documents. Inside the query object we can change how we calculate the correlation of a document to get a better sense of what algorithm would work best for our IR system.

Query 1	CosSim	Okapi	PNW
1	0.5	1	1
2	1	1	1
3	0.5	0.5	1
4	0	0.5	1
5	0.5	0	1
6	0.5	0	1
7	1	0.5	1
8	1	0	1
9	1	0	1
10	0	1	1
Avg	**0.6**	**0.45**	**1**

Query 2	CosSim	Okapi	PNW
1	1	0.5	1
2	0.5	0.5	0.5
3	0.5	0.5	0.5
4	0.5	0.5	0.5
5	0.5	0	1
6	0.5	0	0.5
7	1	0	0.5
8	0.5	0	0.5
9	0.5	0	0.5
10	0.5	0.5	0.5
Avg	**0.6**	**0.25**	**0.6**

Query 3	CosSim	Okapi	PNW
1	1	0.5	1
2	1	0	1
3	0	0.5	1
4	0	0.5	0.5
5	0	0	1
6	0	0	0
7	0	0	0
8	0	0	1
9	0	0	1
10	0	0	0.5
Avg	**0.2**	**0.15**	**0.7**

Query 4	CosSim	Okapi	PNW
1	1	0.5	1
2	0.5	0.5	1
3	1	0.5	0.5
4	0.5	0	0.5
5	0.5	0.5	0.5
6	0.5	1	0.5
7	0.5	0	0
8	0	0	1
9	1	0	0.5
10	0.5	0	0.5
Avg	**0.6**	**0.3**	**0.6**

Query 5	CosSim	Okapi	PNW
1	1	1	1
2	1	0	1
3	1	0	1
4	1	0	0.5
5	1	0.5	0
6	0.5	0.5	1
7	1	0.5	1
8	1	1	1
9	0.5	1	1
10	1	0.5	1
Avg	**0.9**	**0.5**	**0.85**

Query 6	CosSim	Okapi	PNW
1	1	1	1
2	0	0	1
3	0	0	1
4	0.5	0	1
5	0.5	0	1
6	0	0	1
7	0	0	1
8	1	0	1
9	0	0	0.5
10	1	0	0
Avg	**0.4**	**0.1**	**0.85**

Figure 1

Query 7	CosSim	Okapi	PNW
1	1	1	1
2	1	0	1
3	0.5	0.5	0.5
4	0.5	0	1
5	1	1	1
6	1	0	1
7	1	0	0.5
8	1	0	0.5
9	1	0	0.5
10	0.5	0	0.5
Avg	**0.85**	**0.25**	**0.75**

Query 8	CosSim	Okapi	PNW
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	1	0	1
6	1	0	1
7	1	0	1
8	1	0	0
9	1	0	1
10	1	0	1
Avg	**1**	**0**	**0.5**

Query 9	CosSim	Okapi	PNW
1	1	0	1
2	1	0	1
3	1	0	1
4	1	0	1
5	1	0	1
6	0.5	0.5	1
7	0.5	0.5	0
8	0	0	0.5
9	1	0	0.5
10	0	0	0
Avg	**0.7**	**0.1**	**0.7**

Query 10	CosSim	Okapi	PNW
1	1	0.5	1
2	1	0	1
3	0.5	0.5	1
4	0.5	0	1
5	1	0	0.5
6	1	0	0
7	0.5	0	0.5
8	0	0	0.5
9	0	0	0
10	0	0.5	1
Avg	**0.55**	**0.15**	**0.65**

Figure 2

Query	CosSim	Okapi	PNW
1	0.6	0.45	1
2	0.6	0.25	0.6
3	0.2	0.15	0.7
4	0.6	0.3	0.6
5	0.9	0.5	0.85
6	0.4	0.1	0.85
7	0.85	0.25	0.75
8	1	0	0.5
9	0.7	0.1	0.7
10	0.55	0.15	0.65
Avg	**0.64**	**0.225**	**0.72**

Figure 3

Algorithm Analysis

As far as determining the best algorithm for correlation of a document we decided to use a method very similar to the precision formula discussed in class however we decided to add one more level of complexity. This method was somewhat subjective since we decided based on the document retrieved and the query provided gave an estimate on how relevant the document was towards the query. We decided to use .5 as being somewhat relevant to the query and chose 1 as truly relevant for the query. This allowed us to test each query against each algorithm to better determine the most accurate algorithm with our given data.

Figures 1 and 2 show the results of querying the data with each of the algorithms. The bottom of each query is the average precision we found using our slightly modified version of the precision method. After testing each query we came to the conclusion that the pivoted normalization weighting formula was the best choice with an average relevance precision of 72%, as shown by Figure 3. Cosine similarity came in at a close second with 64% and Okapi with 22.5%.

Query Answering

Figures 1 and 2 show the results of how relevant the documents were to the query and the provided text files labeled queryResults##.txt show the calculated relevance, the location of that document in our array list of documents and what order they came in.

Information Need Matching

The majority of time spent was determining which algorithm worked best for the basic queries provided to us. The second part of this was to come up with a query that best answered the information need provided. These information needs are located in the folder of files clearly labeled. We used more than the number of queries listed below but to get an idea of how we progressed from one query to another given the results of a query. A document received a 1 if it completely answered the information need or a 0 otherwise.

Therefore when you see in figure 4 the percentages its basically telling you how many of the ten pulled documents answered the information need. Some of the information needs were much harder to answer especially in the case of only pulling legislators for the first information need. Since our IR system only looked at utterances and not who spoke the utterance it was mostly pure luck on whether or not the document being retrieved was from a legislator. This could have been changed but we decided that changing the entire way in which we search a document would not be advantageous for a single information need. We also noted that an information need does not clearly translate into a query, which was our original thought. The queries we came up with were created through trial and error and each information need was unique in such a way that it would be extremely difficult to come up with an algorithm to translate information needs into a simple query.

Information Need #	Query Text	Accuracy (%)
1	oppose bill vaccination against death	0
	oppose vaccination too far government	10
	too far government parents	10
	oppose vaccination too far government parents diminished	**30**
2	vaccination required for every child in school	10
	support this bill require vaccination	20
	why support this bill	20
	why support this bill SB 277	**30**
3	children medical Disneyland measles	50
	Disneyland measles vaccination	**90**
4	SB 277 amendments	20
	proposed amendments	20
	current proposed committee amendments	**70**
5	cannot prevent school	20
	evidence doesn’t support findings measles	20
	science evidence vaccination	20
	vaccination doubt oppose measles	**30**

Figure 4

Analysis and Conclusion

Three different relevance computation algorithms were implemented for comparison: Cosine similarity, Okapi, and Pivoted normalization weighting. For each one, the same set of 10 queries were used and the results were manually inspected for relevance based on the following scale: 0 for irrelevant, .5 for somewhat relevant (some terms matched, however the overall information need implied by the query was not sufficiently fulfilled), and 1 for completely relevant.

While the cosSim algorithm contains no normalization parameters (all the numbers were derived from the data itself, i.e. document frequency, inverse document frequency, etc.), both Okapi and PNW do. For Okapi, there are three constants that can be adjusted based on the desired normalization effects. These are *k1*, *k2*, and *b*, which we set to 1.5, 500, and .25 (respectively). For PNW, the only normalization parameter that exists is *s*, which we set to .2.

On average, the pivoted normalization weighting algorithm retrieved the best and most relevant results out of the three algorithms. The averages (based on the 10 queries) for each were as follows: CosSim averaged ~64% relevant documents retrieved, Okapi averaged ~23%, and Pivoted normalization weighting averaged ~72%. We concluded that, in most cases, the most effective algorithm for relevance computation is PNW. Moreover, we noticed that information needs (as applied to our IRS) cannot necessarily be directly translated into a query, i.e. word for word. Often times, a shorter and more concise variant of the information need is required to correctly capture it in a query.

Further experimentation would include using varying normalization parameters for Okapi and PNW so as to observe their effects on various queries and the retrieved results.