Using TF-IDF to Determine Word Relevance in Document Queries

Juan Ramos JURAMOS@EDEN.RUTGERS.EDU
Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 08855

Abstract

In this paper, we examine the results of applying Term Frequency Inverse Document Frequency (TF-IDF) to determine what words in a corpus of documents might be more favorable to use in a query. As the term implies, TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. Words with high TF-IDF numbers imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest to the user. We provide evidence that this simple algorithm efficiently categorizes relevant words that can enhance query retrieval.

1. Introduction

Before proceeding in depth into our experiments, it is useful to describe the nature of the query retrieval problem for a corpus of documents and the different approaches used to solve it, including TF-IDF.

1.1 Query Retrieval Problem

The task of retrieving data from a user-defined query has become so common and natural in recent years that some might not give it a second thought. However, this growing use of query retrieval warrants continued research and enhancements to generate better solutions to the problem.

Informally, query retrieval can be described as the task of searching a collection of data, be that text documents, databases, networks, etc., for specific instances of that data. First, we will limit ourselves to searching a collection of English documents. The refined problem then becomes the task of searching this corpus for documents that the query retrieval system considers relevant to what the user entered as the query.

Let us describe this problem more formally. We have a set of documents D, with the user entering a query $q = w_1, w_2, \ldots, w_n$ for a sequence of words w_i. Then we wish to return a subset D^* of D such that for each $d \in D^*$, we maximize the following probability:

$$P(d \mid q, D) \quad (1)$$

(Berger & Lafferty, 1999). As the above notation suggests, numerous approaches to this problem involve probability and statistics, while others propose vector-based models to enhance the retrieval.

1.2 Algorithms for Ad-Hoc Retrieval

Let us briefly examine other approaches used for responding to queries. Intuitively, given the formal notation we present for the problem, the use of statistical methods has proven both popular and efficient in responding to the problem. (Berger & Lafferty, 1999) for example, propose a probabilistic framework that incorporates the user's mindset at the time the query was entered to enhance their approximations. They suggest that the user has a specific information need G, which is approximated as a sequence of words q in the actual query. By accounting for this noisy transformation of G into q and applying Bayes' Law to equation (1), they show good results on returning appropriate documents given q.

Vector-based methods for performing query retrieval also show good promise. (Berry, Dumais & O'Brien, 1994) suggest performing query retrieval using a popular matrix algorithm called Latent Semantic Indexing (LSI). In essence, the algorithm creates a reduced-dimensional vector space that captures an n-dimensional representation of a set of documents. When a query is entered, its numerical representation is compared the cosine-distance of other documents in the document space, and the algorithm returns documents where this distance is small. The authors' experimental results show that this algorithm is highly effective in query retrieval, even when the problem entails performing information retrieval over documents written in different languages (Littman & Keim 1997). If certain criteria are met, they suggest that the LSI approach can be extended to more than two languages.

The procedure we examine with more detail is Term Frequency Inverse Document Frequency (TF-IDF). This weighing scheme can be categorized as a statistical

procedure, though its immediate results are deterministic in nature. Though TF-IDF is a relatively old weighing scheme, it is simple and effective, making it a popular starting point for other, more recent algorithms (Salton & Buckley, 1988). In this paper, we will examine the behavior of TF-IDF over a set of English documents from the LDC's United Nations Parallel Text Corpus. The purpose of this paper is to examine the behavior, strengths, and weaknesses of TF-IDF as a starting point for future algorithms.

2. An Overview of TF-IDF

We will now examine the structure and implementation of TF-IDF for a set of documents. We will first introduce the mathematical background of the algorithm and examine its behavior relative to each variable. We then present the algorithm as we implemented it.

2.1 Mathematical Framework

We will give a quick informal explanation of TF-IDF before proceeding. Essentially, TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF-IDF numbers than common words such as articles and prepositions.

The formal procedure for implementing TF-IDF has some minor differences over all its applications, but the overall approach works as follows. Given a document collection D, a word w, and an individual document $d \in D$, we calculate

$$w_d = f_{w,d} * \log (|D|/f_{w,D}) \quad (2),$$

where $f_{w,d}$ equals the number of times w appears in d, $|D|$ is the size of the corpus, and $f_{w,D}$ equals the number of documents in which w appears in D (Salton & Buckley, 1988, Berger, et al, 2000). There are a few different situation that can occur here for each word, depending on the values of $f_{w,d}$, $|D|$, and $f_{w,D}$, the most prominent of which we'll examine.

Assume that $|D| \sim f_{w,D}$, i.e. the size of the corpus is approximately equal to the frequency of w over D. If $1 < \log (|D|/f_{w,D}) < c$ for some very small constant c, then w_d will be smaller than $f_{w,d}$ but still positive. This implies that w is relatively common over the entire corpus but still holds some importance throughout D. For example, this could be the case if TF-IDF would examine the word 'Jesus' over the New Testament. More relevant to us, this result would be expected of the word 'United' in the corpus of United Nations documents. This is also the case for extremely common words such as articles, pronouns, and prepositions, which by themselves hold no relevant meaning in a query (unless the user explicitly wants documents containing such common words). Such common words thus receive a very low TF-IDF score, rendering them essentially negligible in the search.

Finally, suppose $f_{w,d}$ is large and $f_{w,D}$ is small. Then $\log (|D|/f_{w,D})$ will be rather large, and so w_d will likewise be large. This is the case we're most interested in, since words with high w_d imply that w is an important word in d but not common in D. This w term is said to have a large discriminatory power. Therefore, when a query contains this w, returning a document d where w_d is large will very likely satisfy the user.

2.2 Encoding TF-IDF

The code for TF-IDF is elegant in its simplicity. Given a query q composed of a set of words w_i, we calculate $w_{i,d}$ for each w_i for every document $d \in D$. In the simplest way, this can be done by running through the document collection and keeping a running sum of $f_{w,d}$ and $f_{w,D}$. Once done, we can easily calculate $w_{i\,d}$ according to the mathematical framework presented before. Once all $w_{i,d}$'s are found, we return a set D^* containing documents d such that we maximize the following equation:

$$\Sigma_i w_{i,d} \quad (3).$$

Either the user or the system can arbitrarily determine the size of D^* prior to initiating the query. Also, documents are returned in a decreasing order according to equation (3).

This is the traditional method of implementing TF-IDF. We will discuss extensions of this algorithm in later sections, along with an analysis of TF-IDF according to our own results.

3. Experiment

3.1 Data Collection and Formatting

We tested our TF-IDF implementation on a collection of 1400 documents from the LDC's United Nations Parallel Text Corpus. These documents were gathered arbitrarily from a larger collection of documents from the UN's 1988 database. The documents were encoded with the SGML text format, so we decided to leave in the formatting tags to account for noisy data and to test the robustness of TF-IDF. We simulated more noise by enforcing case-sensitivity. Due to certain constraints, we had to limit the number of queries used to perform information retrieval on to 86. We calculate TF-IDF weights for these queries according to equation (3), and then return the first 100 documents that maximize equation (3). The returned documents were returned in descending order, with documents with higher weight sums appearing first. To compare our results, we also performed in parallel the brute force (and rather naïve)

method of performing query retrieval based only on the term $f_{w,\,d}$. Naturally, this latter method is intuitively flawed for the larger problem of query retrieval, since this approach would simply return documents where non-relevant words appear most (i.e. long documents with plenty of articles and prepositions that might not have any relevance to the query). We will provide evidence that TF-IDF, though relatively simple, is a big improvement over this naïve approach.

3.2 Experimental Results

Return Pos.	Document #	Sum $f_{w,\,d}$	Sum w_d
1	64	139	1.83
2	879	136	4.52
3	1037	121	2.08
4	324	107	0.91
5	710	98	7.22
6	161	93	3.95
7	1175	87	0.24
8	402	86	5.13

Table 1. First eight documents with highest $f_{w,\,d}$ returned by our naïve algorithm for query = "the trafficking of drugs in Colombia". The high $f_{w,\,d}$ comes mostly from long documents with plenty articles and prepositions. These documents had very low w_d scores and are mostly useless to the query.

As expected, the naïve, brute force approach of simply returning documents with high sum of $f_{w,\,d}$ given each query word was very inaccurate. Of the top eight documents shown in Table 1, none are relevant to the given query. This pattern is evident throughout the whole list of 344 words, which is summarized in Figure 1.

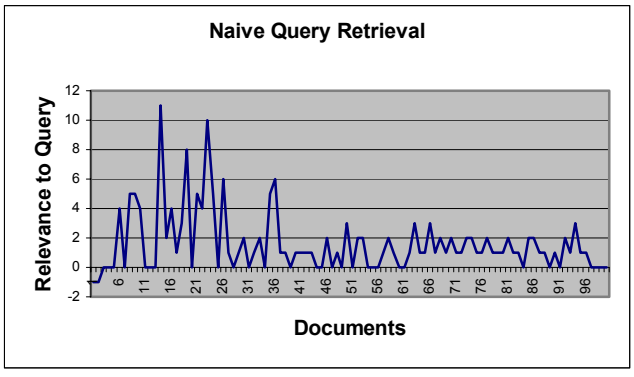

Figure 1. Results of running naïve query retrieval on our data. Notice the algorithm does not consider w_d, but rather returns documents based solely on $f_{w,d}$. Relevant documents are scattered sporadically, so simply returning the top documents as done by this algorithm returns irrelevant documents.

Let us now consider the results from running TF-IDF on our data. In this case, documents at the top of the list have a high sum of w_d, so a query containing w would likely receive document d as a return value.

Return Pos.	Document #	Sum $f_{w,\,d}$	Sum w_d
1	788	24	28.09
2	426	72	26.73
3	881	56	23.96
4	253	43	19.16
5	1007	37	16.5
6	362	29	15.42
7	520	33	12.34
8	23	58	10.79

Table 2. First eight documents with highest w_d returned by TF-IDF for query = "the trafficking of drugs in Colombia". The top document, entitled "International Campaign Against Traffic in Drugs", is intuitively relevant to the query.

As Table 2 shows, retrieval with TF-IDF returned documents highly correlate to the given query. The top two documents make frequent use of the non-article words in the query; words that are not that frequent in other documents. This gives a high sum of w_d, which in turn gives a high relevance to the document. Figure 2 shows this continuing pattern for our data.

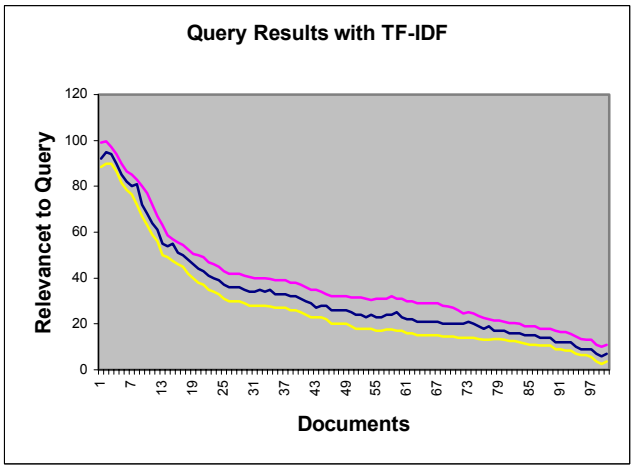

Figure 2. Results of retrieval with TF-IDF on our data. High values of w_d are concentrated on the beginning of the graph, so basing query retrieval on the top words here will likely return relevant documents. The two extra graphs indicate upper and lower bounds found by the retrieval engine.

Clearly, TF-IDF is much more powerful than its naïve counterpart. When examining the words in the query, we see that TF-IDF can find documents that make frequent

use of said words and determine if they are relevant in the document. The discriminatory power of TF-IDF allows the retrieval engine to quickly find relevant documents that are likely to satisfy the user.

4. Conclusions

4.1 Advantages and Limitations

We have seen that TF-IDF is an efficient and simple algorithm for matching words in a query to documents that are relevant to that query. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular query. If a user were to input a query for a particular topic, TF-IDF can find documents that contain relevant information on the query. Furthermore, encoding TF-IDF is straightforward, making it ideal for forming the basis for more complicated algorithms and query retrieval systems (Berger et al, 2000).

Despite its strength, TF-IDF has its limitations. In terms of synonyms, notice that TF-IDF does not make the jump to the relationship between words. Going back to (Berger & Lafferty, 1999), if the user wanted to find information about, say, the word 'priest', TF-IDF would not consider documents that might be relevant to the query but instead use the word 'reverend'. In our own experiment, TF-IDF could not equate the word 'drug' with its plural 'drugs', categorizing each instead as separate words and slightly decreasing the word's w_d value. For large document collections, this could present an escalating problem.

4.2 Further Research

Since TF-IDF is merely a staple benchmark, numerous algorithms have surfaced that take the program to the next level. (Berger et al, 2000) propose a number of these in a single paper, including a version of TF-IDF that they call Adaptive TF-IDF. This algorithm incorporates hillclimbing and gradient descent to enhance performance. They also propose an algorithm for performing TF-IDF in a cross-language retrieval setting by applying statistical translation to the benchmark TF-IDF.

Genetic algorithms have also been used to evolve programs that can match or beat TF-IDF schemes. (Oren, 2002) employs this method to evolve a large colony of individuals. Using the main ideas of genetic programming, mutation, crossover and copying, the author of the paper was able to evolve programs that performed slightly better than the common TF-IDF weighing scheme. Though the author felt the results were not considered significant, the paper shows that there is still interest in enhancing the simple TF-IDF scheme.

Examining our data, the easiest way for us to enhance TF-IDF would be to disregard case-sensitivity and equate words with their lexical derivations and synonyms. Future research might also include employing TF-IDF to performing searches in documents written in a different language than the query. Enhancing the already powerful TF-IDF algorithm would increase the success of query retrieval systems, which have quickly risen to become a key element of present global information exchange.

References

Berger, A & Lafferty, J. (1999). Information Retrieval as Statistical Translation. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval* (SIGIR'99), 222-229.

Berger, A et al (2000). Bridging the Lexical Chasm: Statistical Approaches to Answer Finding. In *Proc. Int. Conf. Research and Development in Information Retrieval*, 192-199.

Berry, Michael W. et al. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, 37(4):177-196.

Brown, Peter F. et al. (1990). A Statistical Approach to Machine Translation. In *Computational Linguistics* 16(2): 79-85.

Littman, M., & Keim, G. (1997). Cross-Language Text Retrieval with Three Languages. In *CS-1997-16*, Duke University.

Oren, Nir. (2002). Reexamining *tf.idf* based information retrieval with Genetic Programming. In *Proceedings of SAICSIT 2002*, 1-10.

Salton, G. & Buckley, C. (1988). Term-weighing approache sin automatic text retrieval. In *Information Processing & Management*, 24(5): 513-523.