Volume : 4 | Issue : 8 | Aug 2015 • ISSN No 2277 - 8179

Research Paper

Determine Word Relevance in Document Queries Using TF-IDF

Computer Science

KEYWORDS : Query retrieval, Ad-Hoc Retrieval, Word retrieval in TF-IDE

Ashok Koujalagi | Computer Science (Network Engineering) Document Query Retrieval Concept

ABSTRACT

In this paper, we examine the results ofapplying Term Frequency Inverse DocumentFrequency (TF-IDF) to determine what words in a corpusof documents might be more favorable to use ina query. As the term implies, TF-IDFcalculates values for each word in a document throughan inverse proportion of the frequency of theword in a particular document to the percentageof documents the word appears in. Words withhigh TF-IDF numbers imply a strong relationship with the document they appearin, suggesting that if that word were to appear ina query, the document could be of interest tothe user. We provide evidence that thissimplealgorithm efficiently categorizes relevantwords that can enhance queryretrieval.

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.

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 $q = w1, w2, ..., wn$ for a sequence of words wi. Then we wish to return a subset D^* of D such that for each $d \in D^*$, we maximize the following probability:

$P(d \mid q, D)$ (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.

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 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.

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.

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 calculatepronouns, 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 fw, d is large and fw, D is small. Then log $(|D|/ fw, D)$ will be rather large, and so wd will likewise be large. This is the case we're most interested in, since words with high wd 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.

Volume : 4 | Issue : 8 | Aug 2015 • ISSN No 2277 - 8179

Therefore, when a query contains this *w*, returning a document *d* where *wd* is large will very likely satisfy the user.

Encoding TF-IDF

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

ϵi *wi, d* (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.

$wd = fw, d * \log(|D|/fw, D)$ (2),

where *fw, d* equals the number of times *w* appears in *d*, |*D*| is the size of the corpus, and *fw, 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 *fw, d*, |*D*|, and *fw, D*, the most prominent of which we'll examine.

Assume that |*D*| ~ *fw, D*, i.e. the size of the corpus is approximately equal to the frequency of *w* over *D*. If $1 < \log(|D|/fw, D) < c$ for some very small constant *c*, then *wd* will be smaller than *fw, 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,

Experiment

Data Collection and Formatting

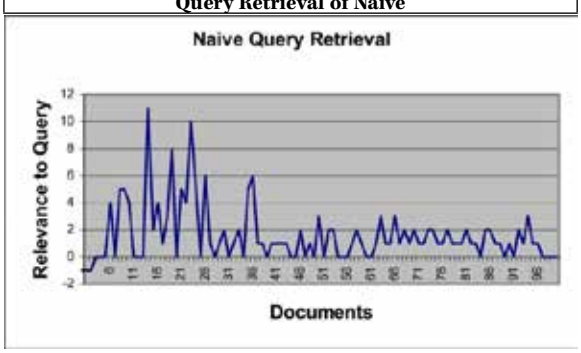
We tested our TF-IDF implementation on a collection of 1400 documents from the LDC's United Nations Parallel Text Corpus. These documents were gathered arbitrarily from a larger collection of documents from the UN's 1988 database. The documents were encoded with the SGML text format, so we decided to 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! , dv 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 naive approach.

3.2 Experimental Results

Return Pos.	Document #	Sumf , d	Sum *md*
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 . d returned by our naive algorithm for query ="the trafficking of drugs in Colombia". The high Ji., d comes mostly from long documents with plenty articles and prepositions. These documents had very low w, scores and are mostly useless to the query.

Query Retrieval of Naive



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 md, so a query containing w would likely receive document d as a retun rn value.

REFERENCE

Salton, G. & Buckley, C. (1988).Term-weighingapproache sin automatic text retrieval. InInformationProcessing & Management, 24(5):513-523. | Oren, Nir. (2002). Reexamining tf.idfbasedinformationretrieval with Genetic Programming. In ProceedingsofSAICSIT 2002,1-10. | Littman, M., &Keim, G. (1997). Cross-LanguageTextRetrieval with Three Languages. In CS-199 7-16,DukeUniversity. | Brown, Peter F. et al. (1990). A Statistical ApproachtoMachine Translation. In Computational Linguistics16(2):79-85. | Berger, A et al (2000). Bridging the LexicalChasm:Statistical Approaches to Answer Finding. In Proc.Int.Conf. Research and Development inInformationRetrieval,192-199. | Berry, Michael W. et al. (1995). Using Linear Algebrafor Intelligent Information Retrieval. SIAMReview, 37(4):177-196. |