SCORING: UNDERSTANDING THE MATH OF ELASTICSEARCH FOR USE IN DATA MINING

Ogbonnaya Chimdimma Ngwu, Kelsey DeJesus-Banos, and Luke Wiskowski
Department of Computer Engineering and
Department of Computer Sciences and Cybersecurity,
Florida Institute of Technology, Melbourne, FL, USA.
ongwu2014@my.fit.edu, kdejesusbanos2013@my.fit.edu, and lwiskowski2014@my.fit.edu

May 2, 2016.

1.0 ABSTRACT

Elasticsearch is a search engine used in most databases. As a search engine, it must be able to analyze data, index document correctly (either implicitly or explicitly), and sort documents based on order of importance. The sorting mechanism of elasticsearch (relevant scoring) applies three techniques in its algorithm. These techniques are explored and their behaviors at extremities were closely monitored and modeled to better understand if and how they can be applied to large data set to select a smaller relevant subset of data (data mining). Certain factors of relevant scoring were modeled and observed to disclose critical values or thresholds where they begin to affect the weight of the term negatively. A statistical point estimator was used to estimate the value of term frequency for a given "n" occurrences of a term in a particular document. This point estimator can then be used in data mining techniques to evaluate the expected value of term frequency impact on a given data set. In most cases, a query does not have single term. A multiterm query has multiple terms connected by either Boolean operator or a white space or punctuation or all of the aforementioned. In such cases, the use of a vector space model is demonstrated where the score of a document used in its ranking is an n-dimensional vector whose numeric values are the weights of each term in the query with n-terms. The relevance of the documents is based on their similarity to the query vector calculated based on the angle between the vector representation of the document and the query vector.

2.0 INTRODUCTION

Elasticsearch is an open source search engine written in java and based on Apache's Lucene[1]. It is designed to be a scalable, distributive, and near real-time capable search engine. As a multitenant system (an instance of the software run on a server and serves multiple clients), a node refers to a running instance of elasticsearch, and creating multiple nodes will result in a cluster[2].

The first version of elasticsearch was released by Shay Banon in February 2010. Since its releases in 2010, it has undergone 20 major updates starting from the earliest version 0.4.0 (no longer supported) to the latest version 2.3.2. All versions of the search engine before 1.4.5 are no longer supported where as versions 1.4.5 unto 2.2.2 are still supported[2].

Elasticsearch as a distributed software allows indices to be divided into multiple shards with each shard being capable of having zero or more replicas. Every node (an instance of the software) hosts one or more of these shard, thereby acting as an operation delegate. It provides the features of Lucene through Java API and JSON and supports percolation and faceting (allows users to search by using multiple filters)[4].

For every index established on a running node, mapping is used to define document types and the structure of the index included, and how its fields are store and indexed[5]. Elasticsearch supports both explicit and implicit mapping. Servers of elasticsearch are capable of detecting document types and indexing them correctly if no mapping was provided explicitly. Although implicit mapping may be handy, explicit mapping allows users to create complex document types and specify how they should be indexed.

As a search engine, elasticsearch must be able to retrieve data from originally indexed document by using keywords provided per query. This is done by combining keywords with Boolean operators to find matching documents to the query using a Boolean model. The selected documents must then be sorted based on their relevance score and presented to the user. The relevance of a document is based on three factors: term frequency (frequency of the term in the field of interest), inverse document frequency (frequency of the term in the whole index), and field-length norm (the structure and nature of the field and its length).

3.0 RELEVANCE SCORING: THE MECHANISM AND ESTIMATES

Elasticsearch being adapted from Apache's Lucene implores the Boolean model in finding relevant data[1]. Although these data (which may be documents in most case) must be sorted and presented to the user based on their importance, the first step involves matching certain data with the keywords provided by the user based on the Boolean operator initiated.

These Boolean operators may be provided by the user explicitly during a query. In a case that no Boolean operator was provided, Elasticsearch applies the OR Boolean operator by default. The Boolean operator that may be applied include: "AND", "OR", and "NOT". The Boolean model of Lucene applies these operators to find every matching document (data) to that query. The following query, "understanding AND math AND (data OR mining) AND Elasticsearch" will return all documents that has all the keywords understanding, math, Elasticsearch and either data or mining.

When all matching documents have been selected from the pool, they must be sorted and ranked based on their decreasing order of relevance[3]. Certain documents are more important than others. This is the same thing you see when you do a google search: the search engine returns documents matching the query keyword arranged in their decreasing order of importance (relevance). Relevance is based on the "weight" of each keyword that appeared in the document. The relevance of a document in elasticsearch depends on three factors already described briefly above: term frequency, inverse document frequency, and field-length norm.

3.1 Term Frequency (TF)

Term frequency refers to the number of times a specific keyword appeared in the document. The weight of the keyword is directly proportional to the frequency of the keyword[5]. Hence, the weight of each term is based on how often it occurred in the

document. The term frequency is mathematically represented as follows:

$$termFreq\ (t\ in\ doc) = \sqrt{frequency}$$

The term frequency for a term "t" that has at least one occurrence in a document "doc" is the square root of the number of occurrence. Term frequency is 0 if the term did not occur in the document and 1 if it occurs in the document only once. Due to the property of the square root function, the weight of each term may be projected visually.

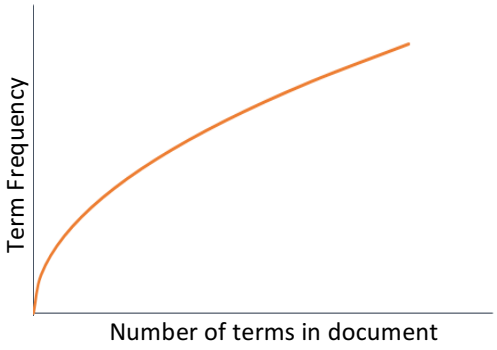

Figure 1: Growth of term frequency.

The weight of a keyword as a result of its term frequency as shown above does not grow exponentially, making it easy to calculate the term frequency for large occurrences. However, it also important to explore what happens at the extrema for the functions of term frequency.

$$\lim_{t\to\infty} \sqrt{t} = \sqrt{\infty} = \infty$$

From above, we see that as the number of occurrence of the term goes to infinity, the term frequency also goes to infinity. Now, we devise our solution to use the area under the curve of the term frequency by integrating portion enclosed by the curve.

$$\int_0^{\infty} \sqrt{t}\ dt = \lim_{n\to\infty} \int_0^{n} \sqrt{t}\ dt$$

The above improper integral does not converge. Hence, we may assume the weight of a term with such number of occurrences to be the maximum weight attainable in that document.

In some cases, the term frequency for a particular term may not be the factor of interest. Disabling term frequency by setting "index_options" to "docs", the number of occurrences is not counted and the only information we have regarding the term is either its presence or absence in the document.

3.1.0 Statistical Point Estimation of TF

Since at extrema, the term frequency of any given term as its frequency goes to infinity also goes to infinity, it is necessary to establish an estimation technique for the weighting factor. The estimated variable (TF in this case) is represented as theta (θ) and a point estimator, theta hat ($\hat{\theta}$) is used to make an estimate for the term frequency[2]. The point estimator is biased by some values. This bias is calculated as the mean square error (MSE) of the estimator. It is important that to note that the MSE of the point estimator reduces as the number of occurrences increase due to the reduction in variance from the actual TF. For a given "n" occurrences of a term in a document, the point estimator and the estimated expected value of the point estimator is given as:

$$\hat{\theta} = \int_0^{n} \sqrt{t}\ dt$$

$$E(\hat{\theta}) = E\left[\int_0^{n} \sqrt{t}\ dt\right] = \int_0^{n} t\sqrt{t}\ dt$$

The bias, B of the point estimator is given as:

$$B(\hat{\theta}) = E(\hat{\theta}) - E(\theta)$$
$$= E\left[\int_0^n \sqrt{t}\, dt\right] - E(\theta)$$
$$= \int_0^n t\sqrt{t}\, dt - E(\theta)$$

The MSE of the point estimator is expressed as the sum of variance of the estimator from the actual TF and the square of the bias.

$$MSE(\hat{\theta}) = V(\hat{\theta}) + B^2$$
$$= E(\hat{\theta}^2) - \left[E(\hat{\theta})\right]^2 + B^2$$
$$= \int_0^n t^2\sqrt{t}\, dt - \left(\int_0^n t\sqrt{t}\, dt\right)^2 + \int_0^n t\sqrt{t}\, dt - E(\theta)$$

3.2 Inverse Document Frequency (IDF)

The inverse document frequency refers to the number of occurrences of a term in a documents selected using the Boolean model. Unlike term frequency, the IDF is inversely proportional to the logarithm of the number of documents containing the term and directly proportional to the number of documents in the collection[2]. Due to this inverse relationship, words with high document frequency will have lower weights than words with lower document frequency. This ensures that words that are commonly used like "the" or "and" does have high relevance compared to less used words like "mining". The IDF is represented mathematically as follows:

$$IDF(t) = 1 + \log\left(\frac{numDocs}{docFreq + 1}\right)$$

The IDF of a term t is one plus the logarithm of the number of documents in the collection, divided by the number of document that contain the term. The IDF of any term is always greater than or equal to 1. Since the logarithm operator is undefined for zero, a numeric one is added to the document frequency to ensure the function is not broken. In the case that document frequency is 0 the lowest IDF simply depends on the number of documents in the collection. Due to the property of the logarithmic function, the IDF of each term based on number of documents and document frequency may be projected visually. The graph below illustrates the relationship between IDF and the number of documents and document frequency for a term. It assumes one document for simplicity, However, as the number of documents increases, the only change observed is the intercept on x-axis which also increase.

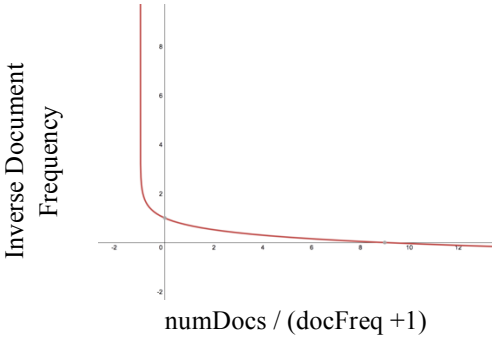

Figure2: Growth of IDF

As always, the extremas of these functions are always a point of interest. Since the number of documents in a collection doesn't change overtime, a constant value, c, is assigned to it and document frequency is represented as x, the only variable in this case.

$$\lim_{x\to\infty}\left\{1 + \log\left(\frac{c}{x+1}\right)\right\}$$
$$= \lim_{x\to\infty}(1) + \lim_{x\to\infty}\left\{\log\left(\frac{c}{x+1}\right)\right\}$$

$$= 1 + \lim_{x\to\infty}(c) + \lim_{x\to\infty}\left\{\log\left(\frac{1}{x+1}\right)\right\}$$

$$= 1 + c + (-\infty) = -\infty$$

As the document frequency of the term increases, the weight of the term based on its IDF decreases to negative infinity. As shown earlier, weight of the term can be calculated to get a reasonable value by integrating area partitioned below the curve.

$$\int_0^{\infty} 1 + \log\left(\frac{c}{x+1}\right)\, dx$$

$$= \lim_{n\to\infty} \int_0^{n} 1 + \log\left(\frac{c}{x+1}\right)\, dx$$

The above improper integral does not converge. Hence, assume the weight of a term with such document frequency to be the minimum weight attainable in that overall indexed document collection.

The intercept on the x-axis is the critical point or threshold of the IDF. After this point, an increase in document frequency results in a negative weight for the term. The threshold can be calculated as follows:

$$T_c = (numDoc \times 10) - 1$$

The threshold function has a linear relationship with the number of documents. Hence, it goes to infinity as number of documents in the collection goes to such extremas, and since it is not a closed function, it does not converge when integrated from 0 to ∞.

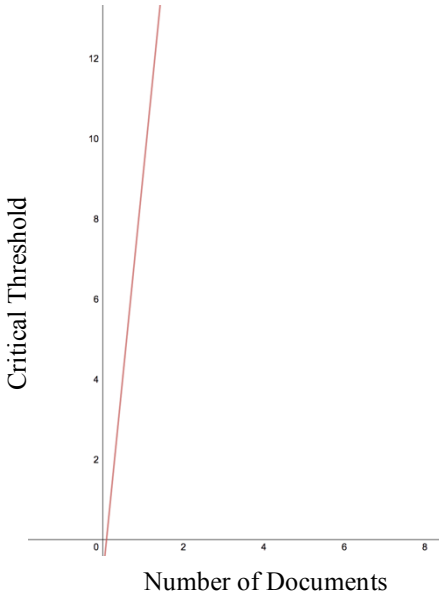

Figure 3: Graph of Critical Threshold

3.3 Field-length Norm

Field refers to the location of the term. Terms may be found at different parts of the document during the search[1]. The fields may include title, or the heading, or the body of the document. The field-length refers to the size of the body. Usually, titles and headings have smaller field-length than bodies of a particular document[2].

Field-length norm is inversely related to the weight. Terms that appear in shorter fields like title, heading have higher weight since the content of such field has more impact on the document than the large body fields. The field-length norm (denoted as norm(d)) is represented mathematically as follows:

$$norm(d) = \frac{1}{\sqrt{fieldFrequency}}$$

Field frequency refers to the number of times a term appeared in a particular field of interest. The field frequency of a term that appeared twice in the title of a document is just 2. The field-length norm is the inverse square root of the frequency of the term in

that particular field. Norm of a term appearing once in the field is just 1 and it is undefined for zero frequencies. The lowest number of weight attainable by a term through its field length norm is always greater than 0. As the number of of occurrences of a term in a field increases, the norm decreases. Using the property of the square root function, field length norm of each term based on its field frequencies may be projected visually.

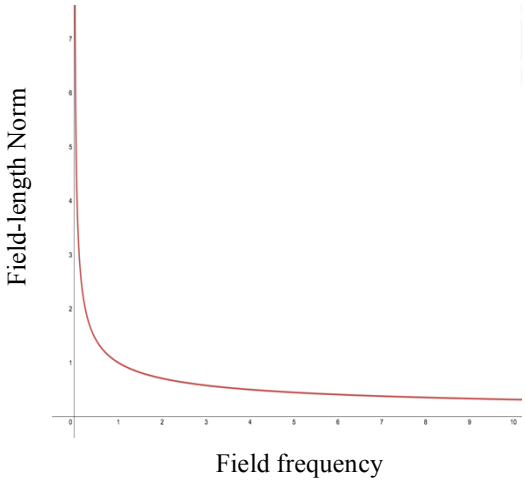

Figure 4: Growth of Field-length Norm

The extremities are always a point of interest. The field frequency being the variable in this case is represented by x.

$$\lim_{n\to\infty}\left(\frac{1}{\sqrt{x}}\right)=\frac{1}{\infty}=0$$

As the number of occurrence of a term in a field increases, the norm decreases rapidly and goes to 0 as field frequency goes to ∞. Hence, norm values most certainly falls between the range: $0 < norm(d) < \infty$.

4.0 THE VECTOR SPACE MODEL

Previous considered the case of a single term in the query statement. In contrast, most cases a query statement would have multiple terms either separated by a Boolean operator or a punctuation such as a comma or a semicolon. These kinds of queries are referred to as *multiterm queries*. The vector space model is used in comparing the multiterm term query to matching documents[5]. Hence, there exist only one single score that wholly represent the extent to which a particular document matches the search term. This single score is in the form of a vector, a one-dimensional array with numeric values.

The numeric values in the vector returned to represent the score of the document are the weight of each term in the query separately calculated using the methods described in previous session (term frequency, inverse document frequency, and field-length norm). The vector (2, 0, 1, 7) represents the score of a document matching a multiterm query (four terms in this case) in which the first, second, third, and fourth terms have weights 2, 0, 1, and 7 respectively.

Consider a four term query, “let time constraint”. The term “let” is commonly used and will certainly have a low weight, and for the case of practicality, 2 is assumed to be its weight. The weight of the term “time” is assumed to be 4 since it is also used frequently. The term “constraint” is hard to encounter and the weight is assumed to be 7 in this case. The three dimensional vector [(2,4,7)] is represented as a three dimensional line starting at point (0,0,0) and ending at point (2,4,7).

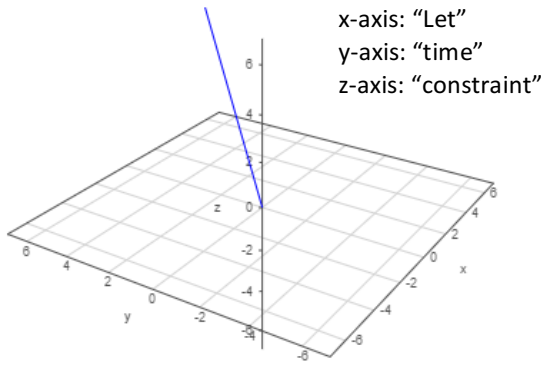

Figure 5: 3-dimensional model of query vector (2,4,7)

Assuming there were three documents (with only one statement per document, for simplicity) matching the query. The sample documents and their terms are shown below.

1. *Let* every *time* be nice.
2. The *time* has a *constraint.*
3. *Let* the *time constraint* be 0.

For each document, a similar vector model representing its score by weight of each term can be plotted

1. [Let, time, _____]: (2,4,0)
2. [__, time, constraint]: (0,4,7)
3. [Let, time, constraint]: (2,4,7)

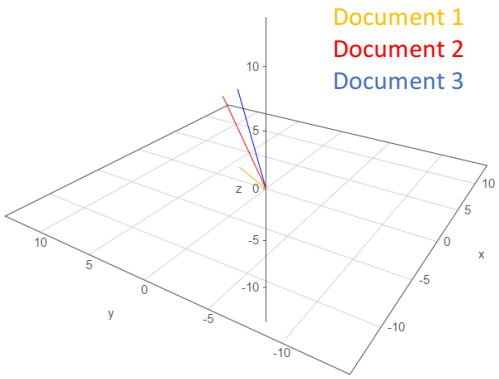

Figure 6: 3-dimensional model of the documents scores.

These vectors shown above in the model can be compared to assign a relevance score by calculating the angle between the document vector and the query vector using cosine similarity. A large angle lower score and hence the lowest relevance. In the sample query and matches used here, document 1 has the largest angle and hence the lowest relevance whereas document 3 has the same angle as the query and hence has the highest relevance. The relevance scoring of these vectors based on cosine similarity is shown mathematically below[5]:

$$a \cdot b = \|a\|\|b\|cos\theta \quad \text{(Euclidean dot product)}$$

$$similarity = cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

$$= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where A_i and B_i are components of vectors A and B.

5.0 APPLICATION TO DATA MINING

As an interdisciplinary field of computer science, data mining combines artificial intelligence, machine learning, database systems, and statistical methods in exploring patterns in voluminous data sets through complex computational processes. The major task in a data mining project is extracting important information from a given data set and representing the information in an easily comprehensible structure for further use. While data extraction is the major step in data mining, the process of extraction is just as important as the the extracted data itself[2]. A traditional data mining process involves six tasks: anomaly detection (finding errors that should be investigated), dependency modelling (figuring out how each variable is related to other variables in the data set), clustering (unveiling the structure and grouping dynamics of the data set),

classification (associating a known structure to the discovered structure of the data set), regression (modeling the data set with a function that results in the smallest error), and summarization (representing the data in an understandable structure).

The major application of the concepts and mechanisms introduced in previous sessions is in the data extraction phase of data mining. How do we know parts of the data set that is important? How do we know some parts are more relevant than other parts? How do we determine the general impact of a single datum in the whole data set? By combining factors of the scoring system of elasticsearch, the relevant data in a data set can be extracted for use in structured summary and yet an understandable data, which is the sole aim of data mining[4].

To use this scoring system in data mining, three important factors must be taken into consideration: the volume of the data set, the rate of change of information in the data set, and the variety of the data set [2]. The size of the data is important since frequency of term changes depending on number of documents in the set. For a changing data set, the weight for each term is expected to change per mutation and the diversity of the data set is also quite important.

6.0 CONCLUSION

Every time you use a speech recognition software, a facial recognition software, or a biometrics machine, it is important to understand that in each case data processing occurs. In all these scenarios and more, the data taken as input is matched to to certain data pre-entered in the computer to either produce valid match or an invalid match in a case where it doesn't exist. When a database is queried or search engine is used to find matching documents to keywords, it simply applies certain basic Boolean models to select all matching documents to the query. These selected documents are then sorted based on their relevance by a scoring system. Understanding this scoring system is important in order to know how to make documents published online or any other database to have a high rank based on the expected search keywords.

REFERENCES

[1] C. Gormley, and Z. Tong. *Elasticsearch: The Definitive Guide:* O'Reilly Media, Inc. 2014. Print.

[2] M. S. Divya, and S. K. Goyal. *Elasticsearch: An Advanced and Quick Search Technique to Handle Voluminous Data*. Compusoft, vol. 2, no. 6. 171-175. 2013

[3] M. White. *Critical Success Factors for Enterprise Search.* Business Information Review. Vol. 32, no. 2, 110 - 118. 2015.

[4] O. Baysal, R. Holmes, and M. W. Godfrey. *Developer Dashboards: The need for qualitative analytics*. IEEE Software, 30(4): 46-52, 2013.

[5] R. R. Coifman, S. Lafnon, A. B. Lee, M. Maggionii, B. Nadler, F. Warner, and S. W. Zucker. *Geometric Diffusions as a Tool for Harmonic Analysis and Structure Definition of Data: Diffusion Maps.* Proceedings of National Academy of Science, USA. 102 no. 21, 7426 – 7431. 2005.