Results and Inference Obtained from a Small Implementation of the DF-ICF- The Modified TF-IDF

Vidya Kamath
Assistant Professor
Dept of Computer Science and Engineering
Srinivas School of Engineering, Mangalore
Affiliated to Visvesvaraya Technological University, Belgaum
Karnataka, India

ABSTRACT
DF-ICF is an algorithm designed by modifying the well known TF-IDF, for the purpose of improving the performance and reliability. The work mainly presents the validation of this new algorithm. The algorithm has been implemented with Hadoop using Cloudera, VMware and WampServer in order to conduct experiments. It also presents the results of an experiment conducted on the algorithm. Finally, the performance of the algorithm is predicted based on assumptions by comparing it with that of the TF-IDF. Overall it was found out that DF-ICF is actually better than TF-IDF.

General Terms
Page Ranking Algorithm, Performance, Validation

Keywords
TF-IDF, DF-ICF, Cosine Similarity, Document, Term, Corpus

1. INTRODUCTION
Term Frequency-Inverse Document Frequency- is a well known algorithm which is used to find out the importance of a term within a given document. It is mostly used in massive applications and its performance hence is very important. Document Frequency- Inverse Corpus Frequency is a modification of the TF-IDF where the center of focus is shifted to documents instead of terms. The term found out to be important in a less cherished document obviously does not retain its importance. Hence DF-ICF is designed with an aim to find out the importance of the document even before you apply your TF-IDF, so that you need not waste your time in calculating the TF-IDF of terms in unwanted documents.

The algorithm DF-ICF was first proposed in [1]. Although the algorithm was a mere proposal without any strong ground, its worthiness was yet quite visible. The work on the DF-ICF was kept on the flow and finally a small implementation has been successfully made and the results obtained prove the relevance of the algorithm.

This work is entirely dedicated to exhibit the results and inference of the DF-ICF. The main contributions of the work can be summarized as follows

- Validation of the DF-ICF algorithm based on cosine similarity.
- The implementation details of the DF-ICF using Big Data Platforms for the purpose of this experiment.
- The findings of this experiment with the algorithm by taking different input values.
- The inference on performance of DF-ICF based on the ground of relative assumptions along with a comparison of the algorithm with the TF-IDF.

2. VALIDATING THE DF-ICF
The TF-IDF has a unique property. The TF-IDF associates a number with the documents you are dealing with and hence, you can use these numbers as vectors. And moreover, these vectors follow the cosine similarity. Cosine similarity is one of the similarities metric which depends upon envisioning user preferences as points in space [7].

A small example set had been taken to find out the relationship of the TF-IDF with the cosine similarity. A trial was made to find out whether DF-ICF has a similar relation with the cosine similarity. And it was found out that it does. Since DF-ICF is a modification of TF-IDF, the algorithm can be proved to be valid if it has similar properties as that of TF-IDF. This is what was observed.

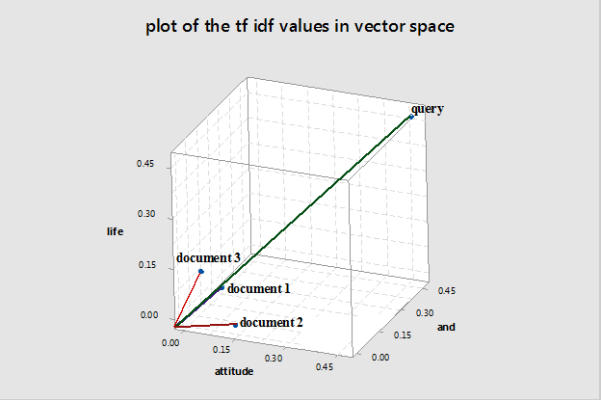

Fig 1: Graphical view of the findings for TF-IDF

Fig 1 shows the plot of TF-IDF for documents and a query and Fig 2 plots the DF-ICF for corpus with a consideration. Refer appendix at the end for details. The document 1 was major candidate which matched the query and it is visible from Fig 1. The minimum is the angle between to vectors, the maximum is their similarity

Similarly the corpus 3 matches best with the consideration and this is visible in Fig 2 as well. This proves that DF-ICF also follows the properties of TF-IDF and hence establishes its validity.

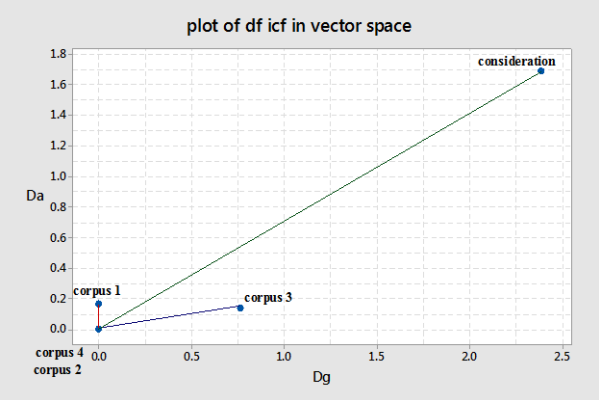

Fig 2: Graphical view of the findings for DF-ICF

3. IMPLEMENTING THE DF-ICF

The algorithm was implemented using 3 softwares- namely the VMware (to create a virtual machine), Cloudera (to use the Hadoop) and Wamp Server (to create websites as inputs to DF-ICF). These softwares are easily available and easy to use. The coding was done in java since it is the language supported by the Hadoop.

The system requirements for successfully running the algorithm was as follows

- Microsoft Windows 7 or higher.
- 64 bit Operating System
- Minimum of 8 GB server
- Support of virtualization

First the coding of TF-IDF was done in the Cloudera which used the Hadoop Map/Reduce for parallelization of the run according to guidelines given in [4]. The Cloudera in turn runs within the VMware as it is a separate OS. Once the TF-IDF is successfully implemented, the DF-ICF was coded using same theory. To create inputs for the DF-ICF which needs corpuses, web sites were created using the Wamp Server. The web sites were considered as collection of corpuses. And each of the corpus contained one or more documents in them. Once the algorithms were successfully deployed, this experiment was ready to be conducted.

4. A SMALL EXPERIMENT

In order to find out whether the algorithm is reliable, an experiment was conducted. The process is as follows

- Take 5 sets of documents as inputs in order to start your experiment.
- Find out the outcome by applying only TF-IDF on the input data set.
- Find out the outcome by applying the DF-ICF before TF-IDF is applied.
- Compare the result to check if there is any improvement.

Table 1: Outcome of the Experiment

RUNS	No of documents present	No of documents which passed the filter
Run 1	10	5
Run 2	20	9
Run 3	25	18
Run 4	30	21
Run 5	40	30

The results so obtained by doing the above process, is as shown in Table 1.

It is clear that while using DF-ICF, the TF-IDF has to be calculated only on those documents which have passed the filter. But when TF-IDF alone is taken into consideration, the number of documents considered is equal to number of documents present since TF-IDF is calculated on all the documents. The Fig 3 below shows the outcome of Table 1 graphically.

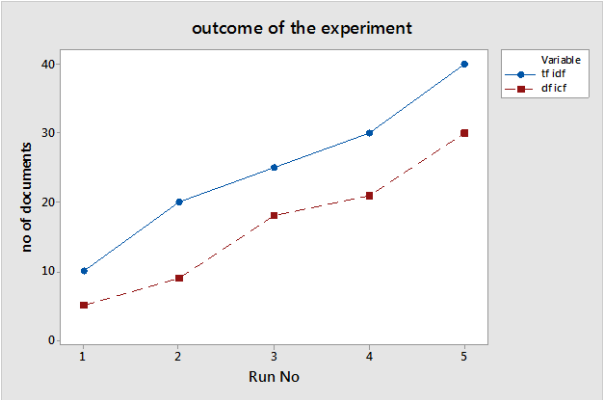

Fig 3: Outcome of the Experiment

For ideal case, keeping the number of documents constant, if you take the total number of documents to be N then TF-IDF has to be calculated for the entire set N while DF-ICF always eliminates a few documents and hence lies below the TF-IDF region as shown in Fig 4 below.

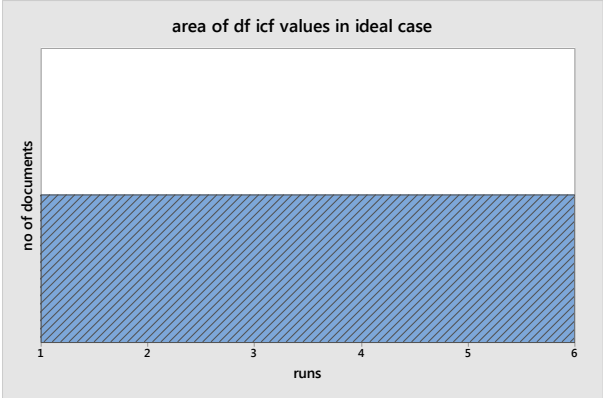

Fig 4: Ideal case

This shows that the more you save your time without wasting on unnecessary calculations on unreliable documents, the more reliable will be your algorithm. This in turn means that the DF-ICF is reliable enough when compared to the TF-IDF.

5. PERFORMANCE EVALUATION

Reliability alone does not make the algorithm worth enough. Time which has more value than gold or platinum in today's era, also counts on while deciding about your proposals. The reason why it was chosen to do the performance estimations of the DF-ICF; taking time as the metric focused.

5.1 Assumption

Assume that you are considering 100 documents. And each of the documents has 100 words/ terms within them. The TF-IDF and DF-ICF are similar and hence their time of calculation will be same. Let us assume it to be 0.1 seconds for now.

Using TF-IDF alone you need to calculate the TF-IDF weight on every document and every term in it. The time taken will hence be as follows:

Total time taken = (time for TF-IDF calculation X Number of documents X number of terms)

= (0.1 s X 100 X 100)

=1000 s

Now consider the scenario when you use DF-ICF. Let x be the number of documents which have passed the filter. You can note that x is always less than the total number of documents.

Let x= 99 (only one document eliminated taking the value next to the worst case)

Total time taken = time for calculating DF-ICF for all documents + time for calculating TF-IDF for filtered documents only.

= (time for calculating DF-ICF X number of documents) + (Time for calculating TF-IDF X Number of terms X Number of filtered documents.)

= (0.1s X 100) + (0.1 s X 100 X 99)

= 10 s + 990 s = 1000 s

Note that the time taken is yet the same and not more than that for TF-IDF even though we have used extra algorithm. More over; this is the worst case. Taking average cases, the time taken will be as shown in Table 2 below.

Table 2: Time comparison between the algorithms

x	Time using TF-IDF alone	Time using DF-ICF and TF-IDF
99	1000 s	1000 s
80	1000 s	810 s
70	1000 s	710 s
60	1000 s	610 s
50	1000 s	510 s
40	1000 s	410 s
30	1000 s	310 s
20	1000 s	210 s

It is clear from the Table 2 that DF-ICF performs better than the TF-IDF while the time consumed is on addition lower than that of consumed by TF-IDF. This is more clearly visible in the Fig 5 shown below.

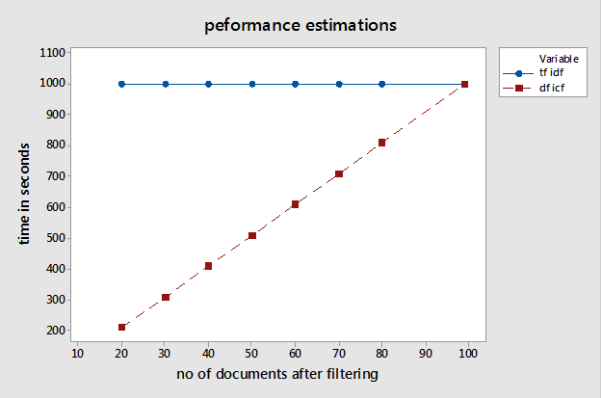

Fig 5: Performance Estimations based on time

This proves that the algorithm not only reliable but also faster than the TF-IDF, while it comes to performance.

6. CONCLUSIONS

The DF-ICF is a successful modification of the TF-IDF. Its validity has been proved on the basis of cosine similarity. The algorithm was also successfully implemented in a small scale. The result of experiment conducted on the algorithm proves its relevance. The performance of the algorithm is also proved to be better as compared to the TF-IDF.

All these infer that the algorithm is suggestible to be deployed on large scale basis. In the Big Data era, even a second saved can make revolutions and hence DF-ICF can prove to be very useful.

7. APPENDIX

7.1 The TF-IDF and Cosine Similarity Calculations

Procedure: Take the example documents given in Table 3 and find out TF-IDF of the terms in the documents.

Now taking an example query- *Life and attitude*, find the TF-IDF of the query. When you find cosine similarities between the query and the 3 documents, the query matches most with document 1, which is obvious as it contains all the three terms. Table 4 gives the cosine similarity values of query with documents.

Table 3 : Three Example Documents

Document 1	Choosing to be positive and having a grateful attitude is going to determine how you are going to live your life.
Document 2	Attitude is a little thing that makes a big difference.
Document 3	Life is ten percent what you make it and ninety percent how you take it.

Table 4 : Cosine Similarities for query with all documents

	Document 1	Document 2	Document 3
Cosine Similarity	1	0.57735026919	0.816496580928

Table 5: Cosine Similarities for the 'Consideration' with the 4 Corpus.

	Corpus 1	Corpus 2	Corpus 3	Corpus 4
Cosine Similarity	0.57866605	0	0.905973287	0

Table 6: Example Corpus details

Da	The game of life is a game of everlasting learning.		
Db	The unexamined life is not worth living.		
Dc	Never stop learning.		
Dd	Choosing to be positive and having a grateful attitude is going to determine how you are going to live your life.		
De	Attitude is a little thing that makes a big difference.		
Df	Life is ten percent what you make it and ninety percent how you take it.		
Dg	Better late than never.		
Number of views in the corpus and its documents			
Corpus 1= 50	Corpus 2= 150	Corpus 3= 80	Corpus 4= 10
Da= 5	Dd= 40	Da= 20	Db=10
Db=10	De=60	Df=35	Dg=0
Dc=35	Df=50	Dg=45	
Corpus 1	**Corpus 2**	**Corpus 3**	**Corpus 4**
Da, Db, Dc	Dd, De, Df	Da, Df, Dg	Db, Dg

7.2 The DF-ICF and Cosine Similarity Calculations

Procedure: To find the DF-ICF, take the example corpus shown in Table 6. Again as done before, calculate the DF-ICF for documents within corpus. Taking an example consideration which includes documents Da and Dg, find the DF-ICF of the consideration. Now find cosine similarities of consideration with all corpuses. Table 5 gives the values obtained. The most matched is corpus 3 which is obvious.

8. REFERENCES

[1] Puneet Goswami, Vidya Kamath, "The DF-ICF algorithm- Modified TF-IDF", International Journal of Computer Applications, Volume 93, No 13, May 2014.

[2] SALTON G, BUCKLEY C. Term-weighting approaches in automatic text retrieval [J]. Information Processing and Management, 1988, PP513 - 523.

[3] SALTON G, CLEMENT T Y. On the construction of effective vocabularies for information retrieval[C].Proceedings of the 1973

[4] Bin Li, Yuan Guoyong- " Improvement of tf-idf for Hadoop Framework" The 2nd International Conference on Computer Application and System Modeling (2012)

[5] Moty Fania, John David Miller- White paper- "Mining Big Data in the Enterprise for Better Business Intelligence", Intel July 2012

[6] Puneet Goswami, Vidya Kamath-"Big Data- Driving force for innovation and Value Recreation", IJARCSSE Volume 4, Issue 3-March 2014.

[7] Jana Vembunarayanan, " TF-IDF and cosine simililarity", Seeking Wisdom, Oct 2013

[8] "Making data Analytics Work- three Key Challenges. McKinsey and Company. IDC Digital Universe Study, sponsored by EMC, June 2011

[9] Stamatis Karnouskos-" Big data analytics for Smart Grid Cities" . Eurescom mess@ge 1- 2013.

[10] LiThomas H Davenport, Jill Dyche- "Big Data in Big Companies" International Institute for Analytics, may 2013

[11] M. Santhanakumar , C. Christopher Columbus- "A modified frequency based term weighting approach for information retrieval " , Int. J. Chem. Sci.: 14(1), 2016, 449-457 ISSN 0972-768X.

[12] Liu Zhenyan, Meng Dan, Wang Weiping, Zhang Chunxia - " A Supervised Parameter Estimation Method of LDA" , 17th Asia-Specific Web Conference, APWeb 2015, China, 2015 proceedings- Springer .

[13] Chengzhi Zhang, Huilin Wang, Yao Liu , and Hongjiao "Document Clustering Description Extraction and Its Application" , XuW. Li and D. Mollá-Aliod (Eds.): ICCPOL 2009, LNAI 5459, pp. 370–377, 2009, Springer.