Role of Weighting on TDM in Improvising Performance of LSA on Text Data

Sudarsun S, *Member, IEEE*, Venkatesh Prabhu G, Sathish kumar V

***Abstract*—In this paper, we show that the efficiency of LSA is significantly controlled by the choice of weighting algorithm applied. These weighting algorithms allocate relative importance to the document attributes (e.g. Keywords) based on their occurrences in the corpus. Effects of different weighting algorithms will be the central point of this paper. We experimented with various weighting algorithms to evaluate and study their effects as measured by precision and recall values. Our experiments include weighting function application on TDM (Pre-Weighting) in order to increase or decrease the relative importance of words based on their occurrence. We also evaluated the application of weighting functions on the projected query (post-weighting). Post-weighted keyword queries were projected on an LSA model built on pre-weighted TDM to obtain closely correlated keywords or a document (keyword collection). We have developed a prototype IR query projection tool which projects keyword queries on the LSA model to retrieve relevant keywords with a floating-point score.**

***Index Terms*--Information Retrieval, Weighting Functions, IDF, IWF, WIDF, NDV, LSA, SVD, Precision, Recall, TDM**

I. Introduction

IF computers are designed around their ability to process and store large sets of data, realizing their promise has required the development of accurate and efficient means for accessing that data. Information Retrieval is the science of pioneering, evaluating, and advancing methods for identifying and extracting pertinent content. In this, Information Retrieval requires mathematical simulation of an intrinsically qualitative property: relevance.

Keyword relevance is an important information retrieval method, which works by, returning words related to a projected query. Consider this example; when 'Account' is projected as a query, a user would expect documents that include terms like 'general ledger', 'accounts receivable' and 'trial balance' to be picked up as relevant. While these words are not actually synonyms of the queried term, they are perceptually relevant.

Latent Semantic Analysis (LSA) can establish keyword relevancy. LSA identifies latent structure present among the documents using Singular Value Decomposition (SVD) and establishes Word-Word binding. LSI is the modification of the vector-retrieval method that explicitly models the correlation of term usage across documents using a reduced dimensional data. Word-Word binding strongly depends on those weighting algorithms that are applied on the Term Document Matrix (TDM). These weighting algorithms assign relevance to terms based on their occurrence in the corpus and thereby increase the efficiency of information retrieval. We carried out several experiments in order to study the effects of different weighting functions.

This work was supported fully by Checktronix India Pvt. Ltd, Chennai

A. Organization of the Paper

This paper is organized as follows. In Section 2, we discuss prior works related to the effect of weight functions in LSA. In Section 3, we give an overview of LSA. In Section 4, we present various weighting functions, descriptions of each weighting function and their mathematical expression. These include Local Weighting, Global Weighting, Pre-Weighting and Post-Weighting. In Section 5, we present the effects of various weighting functions we experimented with on such attributes as corpus size, number of domains, rank, matrix density, etc. We conclude with a list of inferences.

II. Related Work

Many researchers have studied the impact of weighting on LSA. Jones [12] contributed the TF-IDF weighting function to the information retrieval domain. Dumais [11], proposed local weighting such as term frequency, logarithmic, binary weighting and global weighting such as Normal weighting, GFIDF, IDF and Entropy weighting and tested these algorithms against various corpus sizes and presented precision recall graphs to measure the performance. He concluded that IDF and Entropy seems to increase the performance by 30% while the combination of log and entropy increased the performance by 40%. Nakov et al [8] experimented with combinations of weighting algorithms proposed by Dumais and Jones in order to evaluate critically their efficacy. This enabled us to skip those algorithms with which the above researchers had already experimented and instead to compare their performance with that of alternative algorithms. In addition, we evaluated combinations of weighting functions such as IDF+NDV, IWF+NDV.

III. LSA – An Overview

Latent Semantic Analysis (LSA) [2], [3] is a statistical technique, which describes the underlying structure of texts. It is widely used in author recognition; search engines and computes similarity between texts. LSA is a fully automatic computational technique for representing the meaning of a text as a vector in a high dimensional semantic space. LSA is a two-staged process and includes learning and analysis of the indexed data. During the learning phase, LSA performs an

automatic document indexing. The first step in LSA is to construct a semantic space based on a large body of relevant text. It represents terms and documents in Term–Document vector space called Term–Document Matrix (TDM). Terms are assigned rows and Documents are assigned columns in the TDM. Each cell in the TDM represents the frequency of word **'i'** in document **'j'**. On the TDM, weighting function is applied in order to increase the efficiency of the system. This is one of the tuning parameter of the system. As a result, the cell (i, j) contents should be a better approximation of the interrelations between terms and documents (Weighting functions will be discussed in detail in the next section). The TDM is then subjected to a mathematical transformation called Singular Value Decomposition (SVD) [4], [9]. SVD is multivariate data reduction technique, which approximates high dimensional TDM to low dimensional space. SVD decomposes the TDM in to 3 matrices namely U, S and V. U and V is orthogonal matrices that contain left and right singular vectors respectively. S is diagonal matrix, which contains Singular values as the diagonal values.

$$A = U_{mxr} S_{rxr} V_{rxn}^{T} \quad (1)$$

'r' is the rank of matrix, r <= min (m, n). Here dimensionality reduction is based on 'k', k < r singular values. The best approximation of A then is:

$$A' = U_{mxk} S_{kxk} V_{kxn}^{T} \quad (2)$$

The second stage is analysis. Most often this includes a study of proximity between a pair of documents, a pair of words or between a word and a document.

A. Similarity Computation using LSA

Because in LSA, terms and documents are represented as vectors, the similarity between them can be measured by computing the cosine between the two vectors. Cosine between two vectors C & D is

$$Cos\theta = \frac{C \bullet D}{|C| \bullet |D|} \quad (3)$$

$Cos\theta$ Value ranges from –1 to +1.

Similarity between term-term, document-document and term-document can be measured using the above cosine.

B. Query Projection

A query is tokenized in to terms and their term-frequency is represented as a column vector, say 'q' consisting of 'm' terms. Then the Pseudo-vector **'Q'** for query 'q' can be constructed based on following manipulation [7].

$$Q = q'US^{-1} \quad (4)$$

After constructing the pseudo-vector, post-weighting is applied on the pseudo-vector. Similarity between query Vs term and query Vs document can then be computed.

IV. WEIGHTING FUNCTIONS

As discussed in the introduction section, weighting functions increase the efficiency of information retrieval algorithms by correspondingly varying the importance of terms based on their occurrence in the corpus. Weighting functions allocates weights to each term based on its frequency of occurrence in the corpus. Each cell in TDM is replaced with Local Weighting Function (LWF) and Global Weighting Function (GWF). The LWF approach defines weights in each document. The GWF approach defines weights to term across all the documents in the corpus. Here weighting is applied in two stages. The first stage is called the Pre-Weighting where weighting is applied on TDM before computing SVD. The later is Post-Weighting, which is applied on the pseudo vector during query projection.

A. Normal Weighting

This LWF that normalizes the document vector based on the sum of the frequencies of that particular document vector. Normal weighting scales down the column vector so that column sums up to one in order to nullify the effect of extreme values.

$$n(t_i, d_j) = \frac{n(t_i, d_j)}{\sum_{i^1=1}^{m} n(t_{i^1}, d_j)} \quad (5)$$

where $n(t_i, d_j)$ → Frequency of term 'i' in document 'j'.

B. Normalized Document Vector Weighting

This LWF normalizes the document vector based on its magnitude. It renders the column vector of unit length.

$$n(t_i, d_j) = \frac{n(t_i, d_j)}{\sqrt{\sum_{i^1=1}^{m} (n(t_{i^1}, d_j))^2}} \quad (6)$$

C. Inverse Document Frequency

This GWF is based on the number of documents in which, a particular word appears [5], [6]. It is probably the most popular weighting function used in Information Retrieval. IDF suppress the importance of words by allocating less weight to words that often occur in the corpus and by boosting the weight of words that seldom occur.

$$n(t_i, d_j) = n(t_i, d_j) * \log \frac{N}{NF(t_i)} \quad (7)$$

where N → Total number of documents in the corpus

$NF(t_i)$ → No. of docs in which term 'i' appears.

D. Inverse Word Frequency Weighting

IWF [10] is similar to IDF but it calculates weight based on frequency of occurrence of a particular word across the documents.

$$n(t_i, d_j) = n(t_i, d_j) * \log\left[\frac{\sum_{i=1}^{m}\sum_{j^1=1}^{n} n(t_i, d_{j^1})}{\sum_{j^1=1}^{n} n(t_i, d_{j^1})}\right] \quad (8)$$

E. Composite Weighting Function

By Composite Weighting Function we refer to the application of both LWF and GWF to the TDM. This was done in order both to allocate weight to terms based on their occurrence throughout the entire corpus and also to normalize the column vector. For this experiment, we tried two combination of this weighting function, a) IWF + NDV b) IDF + NDV, (applied IWF or IDF first on the TDM and then NDV). In our experiments, we tried Normal, NDV, IDF, IWF, IDF+NDV and IWF+NDV as pre-weighting schemes followed by IDF and IWF as Post-weighting schemes.

V. EFFECT OF WEIGHTING FUNCTIONS

We conducted a variety of experiments in order to analyze the effect of weighting functions against various attributes. The inference of the experiments is presented below. We have compared the effect of weighting on precision across both varying corpus sizes and varying external document domains. We have compared the choice of best rank for different weighting against the TDM matrix density and corpus size. We have also studied the effect of singular value scaling on precision across varying weighting functions and varying corpus sizes. Finally we have presented experiments on percentile based document filtering and their effect on precision for selected weighting functions.

A. Performance of Weighting based on Corpus size

To evaluate various weighting methods, we applied the aforementioned weighting algorithms on four different corpus sizes: approximately 10,000, 40,000, 70,000 and 100,000 documents. This experiment enabled us to study the effect of weighting functions when corpus size increases.

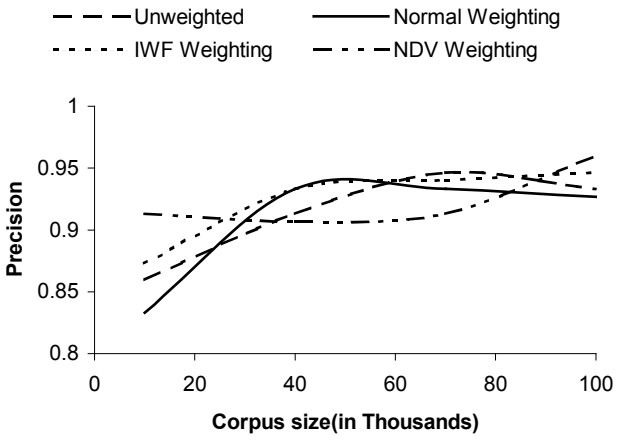


Fig. 1. Performance of Normal, NDV, IWF and Unweighted schemes based on Corpus Size.

On comparing the performances of all weighting algorithms from the graphs it is clear that the IDF+NDV weighting function performs better than other weighting functions across different corpus sizes. Precision of IDF+NDV and IDF weight functions increase if the corpus size increases. The IDF+NDV weighting scheme has produced the highest precision, 0.9933 for corpii of 100,000 documents. IDF technique can also produce good results comparable with those yielded by IDF+NDV, achieving a 98% precision rate for the 100,000 document corpus. From this experiment we show that, for a given corpus size, retrieval precision depends on the choice of weighting algorithm.

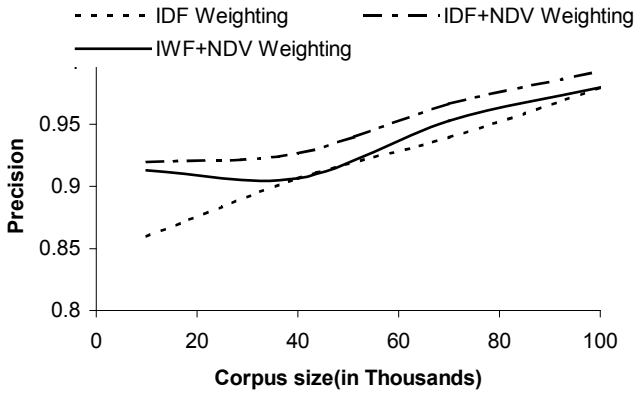


Fig. 2. Performance of IDF, IDF+NDV and IWF+NDV Weighting schemes based on Corpus Size.

B. Effect of Weighting against Number of Domains

In another experiment, we analyzed weighting functions against the number of domains. We collected documents from different domain categories and analyzed how weighting functions vary with respect to change in number of domains for same corpus size. For this experiment we considered a corpus size of 10,000 and carried out three experiments considering 5, 10 and 20 equally populated domains.

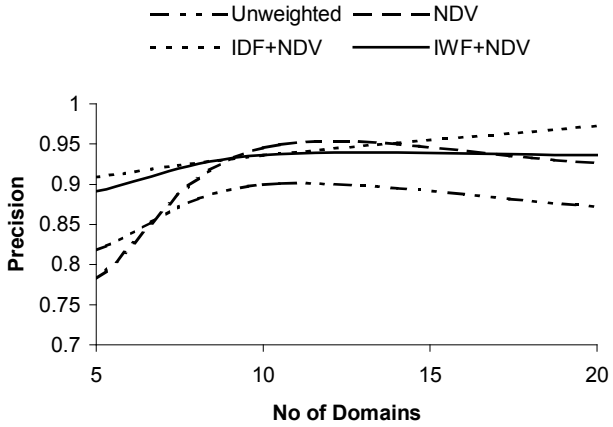


Fig. 3. Effect of IDF+NDV, IWF+NDV, NDV and Unweighted functions against the number of domains.

It can be noted from the graphs that, the performance of all the weighting algorithms steadily increase when the number of domains increases except for the Normal weighting. We see a steady increase in precision for IDF+NDV function. The

performance of IWF+NDV remains almost flat across variations in domain count. Precision for normal weighting initially increases as the number of domains increases from 5 to 10 but the performance falls when the number of domains is increased further. The overall performance of IDF+NDV weighting algorithm was found to be superior when compared to other weighting functions.

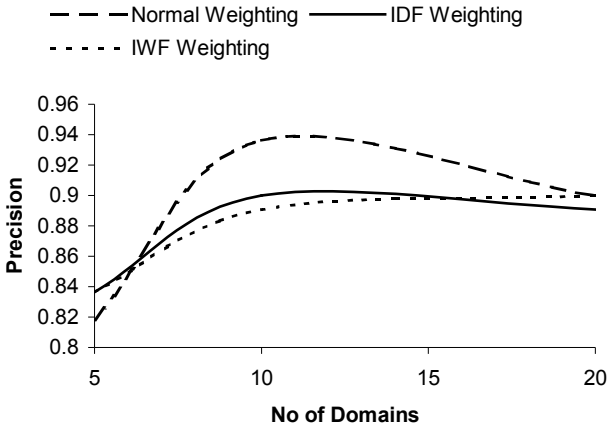


Fig. 4. Effect of IDF, IWF and Normal weighting schemes against the number of domains.

C. Impact of Weighting with respect to Matrix Density

We have performed some experiments to compare the choice of best rank for different matrix densities of TDM. In general TDM is very sparse (< 2%). This study is important in the context of computer programs. That is, the bigger the rank, the greater the amount of memory needed, the greater the time taken to complete computation and the bigger the disk footprint.

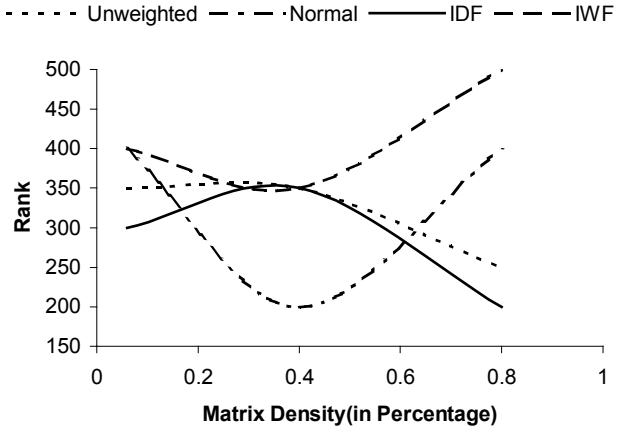


Fig. 5. Impact of IDF, IWF, Normal and Unweighted functions against Matrix Density.

"Fig. 5", "Fig. 6" shows how rank varies with respect to matrix density for each weighting functions. For IDF and unweighted schemes, the best rank lies between 200 and 350. The best rank increases when matrix density increases from 0.4% and then decreases as density increases. But the choice of rank for Normal weighting function falls down from 400 to 200 when matrix density is 0.4% and from there, rank increases steadily as matrix density increases. For the IWF function, the best rank decreases from 400 to 350 when matrix density is 0.4% and from there it grows higher when the matrix densities increase. For the IDF+NDV weighting scheme, the best rank drops from 350 to 250 when matrix density is 0.4% and from there it increases slightly as matrix density increases.

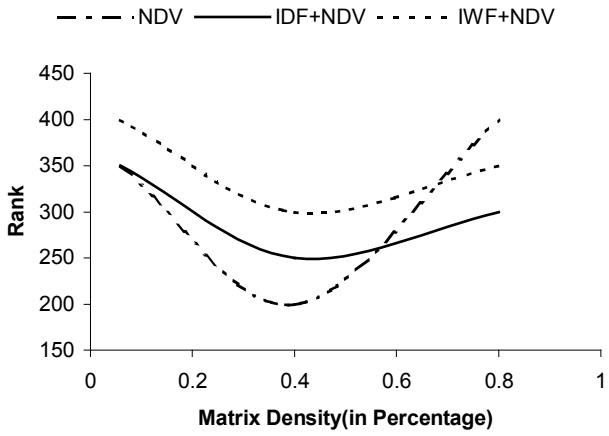


Fig. 6. Impact of IDF+NDV, IWF+NDV, NDV functions against Matrix Density.

For IWF+NDV, there is a decrease to 200 when the matrix density is approximately 0.4%. Thereafter the best rank increases, as does the matrix density. For the NDV, the choice of rank decreases when matrix density increases towards 0.4%; thereafter rank increases steadily as matrix density increases. We find that beyond a 0.4% density, rank can be either in increasing or decreasing order but in either case, the trend is consist. That is after 0.4%; it becomes apparent whether we would end up with a bigger or smaller best rank.

D. Choice of rank with respect to Weighting

In another experiment, we compared the choice of best rank against the choice of weighting functions. As LSA uses SVD, a general form of factor analysis, to condense a very large matrix of word-by-context data into a much smaller, but still large-typically 100-500 dimensional representations, the right number of dimensions appears to be crucial.

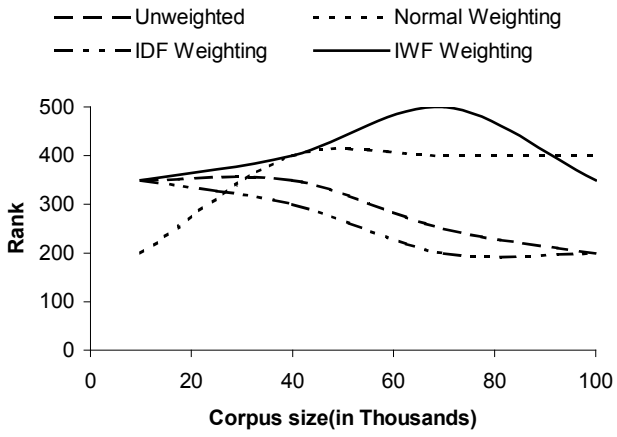


Fig. 7. Choice of Rank with respect to Normal, IDF, IWF and Unweighted schemes.

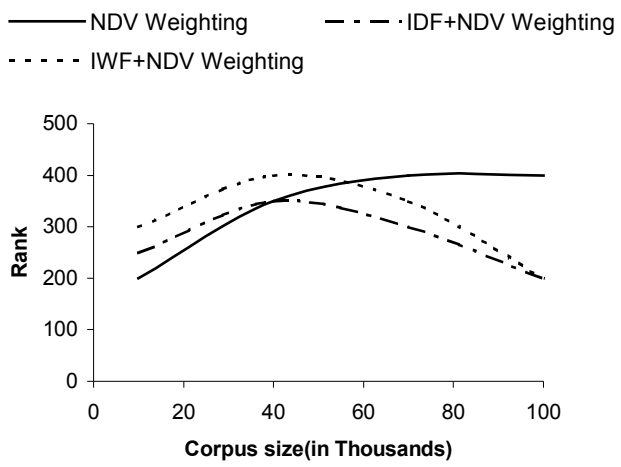


Fig. 8. Choice of Rank against IDF+NDV, IWF+NDV and NDV Weighting

From the graphs, we can infer the following: 1) When NDV is applied along with IWF and IDF, best rank was found to be between 200 and 400; 2) Choice of rank decreases as the corpus size increase for Unweighted and IDF functions; 3) For NDV and Normal weighting function, choice of rank increases as the does the corpus size; 4) IWF weighting function has the tendency to perform well when the rank is above 350.

E. Effect of Weighting With Singular Value Scaling

The S–matrix in LSA is an array of singular values, which can be understood as a scaling factor. It is not mandatory that one should use S value scaling while computing word-word correlation. But the use of S-value scaling boosts and diminishes the scores considerably. In this experiment, we studied performance by varying weighting functions along with S-value scaling. We have redone the experiments as that of "Fig. 1", "Fig. 2" with scaling to plot the following graphs.

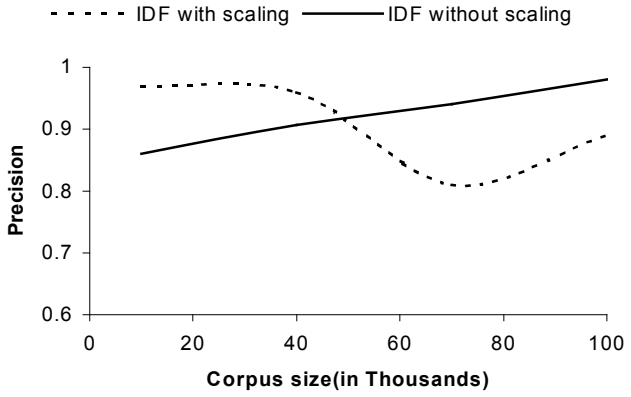


Fig. 9. Effect of scaling using S-Matrix with respect to IDF Weighting.

We find that the effect of S-value scaling is significant when we hold the rank constant as that of the rank chosen for experiments without scaling. Although there is degradation in the performance when the corpus size grows, it is apparent that we should nonetheless choose a different best rank. We have to redo precision-recall calculations from scratch to evaluate performance with scaling as the choice of rank changes when scaling is applied.

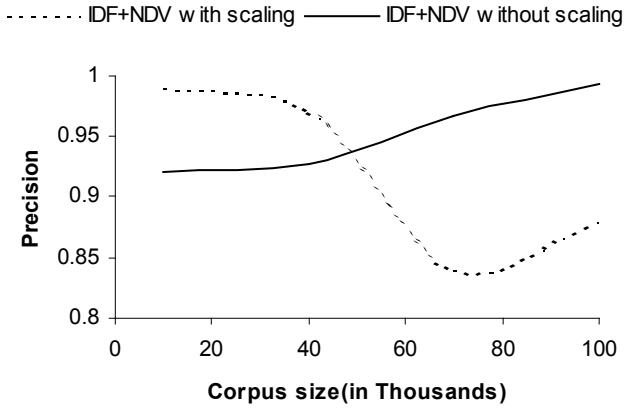


Fig. 10. Effect of scaling using S-Matrix with IDF+NDV weighting

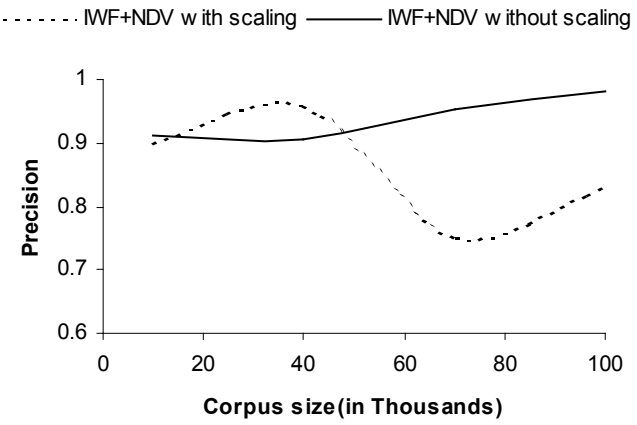


Fig. 11. Effect of S-Matrix scaling for IWF+NDV Weighting

F. Weighting Functions and Percentile based Noise Filtering

After generating the TDM, we performed percentile-based noise filtering on TDM. This was done in order to remove documents that are populated with fewer keywords so that we can ensure the richness of the training set. We used percentile value as one of the tuning parameter in the learning process of the system. We used the normalized keywords count in each document as an input to the percentile filter. Documents that have a count less than n^{th} percentile were removed from TDM. For our experiment, we considered corpus size of 100,000 and shrunk the TDM at the 10^{th}, 25^{th} and 50^{th} percentiles. We compared performance with that exhibited in unfiltered TDM. The graphs below indicate the performance of weighting algorithms with respect to Percentile based noise filtering. It can be seen that precision of IDF and IDF+NDV increases at the 10^{th} percentile when compared to that of an unfiltered model. After the 10^{th} percentile, precision falls as the percentile increases. There is an increase in precision for IWF and IWF+NDV when the percentile is increased. Performance of IWF and IWF+NDV increases when the corpus is rich enough. Overall the performance of all the weighting algorithms increases when we do filtering based on percentiles. But the right choice percentile value proves crucial.

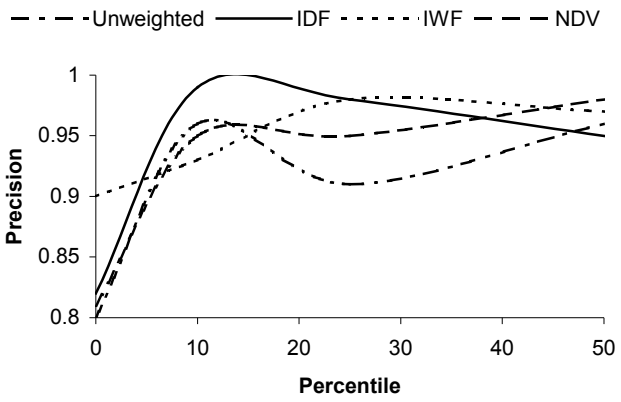


Fig. 12. Performance of IDF, IWF, NDV and Unweighted functions with respect to Percentile based noise filtering.

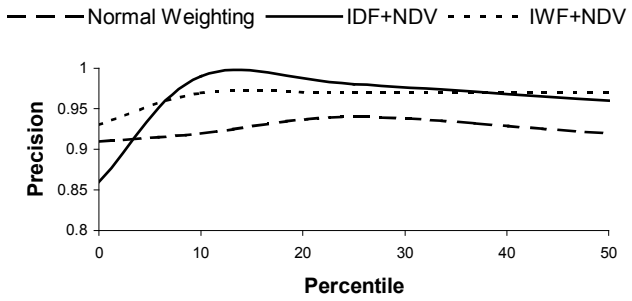


Fig. 13. Performance of IDF+NDV, IWF+NDV and Normal Weighting with respect to Percentile based noise filtering.

VI. Conclusion

In this work we described the effects of weighting on various factors as demonstrated through the results of six core experiments. The first experiment analyzed how performance of the weighting algorithms varies with training corpus size. In this experiment the IDF+NDV weighting function performed better across all corpus sizes. In the second experiment, we compared the performance of the LSA model as we applied various domain-specific weightings derived from observations from the training set. In this experiment, we found that performance of all the weighting schemes increase when number of domains increase except that Normal Weighting schemes and IDF+NDV delivered relatively superior performance. In our third experiment, we tried to relate best rank of SVD for different weighting-function to Matrix Density. The fourth experiment measured the relationship between the choice of best rank and corpus size for different weighting functions. The fifth experiment compared the results with and without S-Matrix scaling applied on the correlation estimates. In the last experiment, we studied the effect of applying percentile based noise filtering method to the TDM before applying weights. We found that performance of the model is better when we filtered TDM at 10 percentile.

Based on the above experiments, we conclude that the combination of IDF and NDV functions deliver superior performance relative to that of other weighting algorithms. IDF+NDV produced 99% accuracy for 100K corpus whereas IDF produced 98%.

VII. References

[1] C.J van Rijsbergen "*Information Retrieval*", (2nd Edition). Butterworths, London 1979

[2] Landauer, T. K., Foltz, P. W., & Laham, D. "Introduction to Latent Semantic Analysis*", Discourse Processes*, 25, 259-284. (1998).

[3] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. "Indexing By Latent Semantic Analysis", *Journal of the American Society For Information Science*, 41, 391-407. (1990)

[4] G. Furnas, S. Deerwester, S. Dumais, T. Landauer, R. Harshman, L. Streeter and K. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," *in The 11th International Conference on Research and Development in Information Retrieval, Grenoble, France: ACM Press*, pp. 465--480. (1988)

[5] K.W. Church and W.A. Gale. "*Inverse document frequency: a measure of deviations from Poisson*". In Armstrong et al. (eds.), NLP using very large corpora, Kluwer Academic Publishers, 1999**.**

[6] Papineni, "Why Inverse Document Frequency?", North American Chapter Of The Association For Computational Linguistics. 2001

[7] Todd A. Letsche and Michael W. Berry. "Large-scale information retrieval with latent semantic indexing." Information Sciences, 1997.

[8] P. Nakov, A. Popova, P. Mateev. "*Weight functions impact on LSA performance*". Proc. RANLP'2001, pp. 187-193, 2001

[9] Kalman "A Singularly Valuable Decomposition: The SVD of a Matrix", *The College Mathematics Journal*, Vol. 27, NO. 1, January 1996

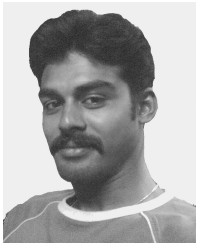
[10] Keli Chen and Chengqing Zong, "A New Weighting Algorithm for Linear Classifier"

[11] Dumais S. "Improving the retrieval of information from external sources". Behavior Research Methods, Instruments, & Computers, 23(2):229-236.1991.

[12] Jones, S. K. "A statistical interpretation of term specificity and its application in retrieval". *Journal of Documentation*, 28:11—21. (1972)

VIII. Biographies

Sudarsun S (M' 2002) is the Director – R & D at Checktronix India Pvt Ltd, Chennai. He holds an M.Tech Computer Science from IIT Madras. He received a BE degree in Electronics and Instrumentation Engineering from Madras University with a Gold Medal. His research experience includes Statistical Natural Language Processing, Machine Learning and Distributed Computing.



Venkatesh Prabhu G is a Research Associate at Checktronix India Pvt Ltd, Chennai. He holds an ME in Computer Science from Anna University, Chennai. He completed his BE in Computer Science at MKU, Madurai. His research interests include AI, Data Mining, Pattern Classification, Knowledge Based Neural Networks, Machine Learning, Committee Machines and Hybrid Techniques in Soft Computing.

Sathish Kumar V is a Research Associate at Checktronix India Pvt Ltd, Chennai. He completed his M.Sc and B.Sc. in Statistics from Loyola College, Chennai. His research interests include LSA, Data Mining and Probabilistic Machine Learning Models.