# Natural Language Processing Project

Mooizz cs17b034, Subhash cs17b005

Indian Institute of Technology Madras, Tamil Nadu, India.

**Abstract.** In this project, We build a Information Retrieval System and evaluate it's performance using Cranfield Dataset, which contains 1400 documents and 225 queries. We implement and compare the performance of different Vector Space based models.

**Keywords:** search engine · vector space · latent semantic indexing · explicit semantic analysis · relevance feedback.
**Github link to the project:** here

## 1 Introduction

Information retrieval is the most widely used application of Natural Language processing. Vector Space Models are conventionally used to build the application. In this model we represent the queries and documents in an multidimensional space and perform retrieval based on these representations.

The basic form of this model that we implemented in the course assignments represents the documents and queries in term space. These representations are formed by performing indexing on terms occurred and forming a vector of weights associated with each term. These weights are formed using the frequency of terms and other measures. Finally we use similarity measures between document and query representations to perform retrieval.

In this project we try to improve upon the Basic vector space retrieval Model by incorporating semantic content and relevance feedback.

## 2 Motivation

This section talks about the limitations of the Basic vector space model and the methods we selected to tackle these limitations.

The Basic vector space Model is a bag of words model and it only considers the individual terms. The fundamental problem here is that users want to retrieve on the basis of conceptual content, and individual words provide unreliable evidence about the conceptual topic or meaning of a document. The need of including the "semantic content" can be explained by two issues

1. **Synonymy** , There may be different terms to describe the same object. Users using different context or having different needs, knowledge and different linguistic habits will describe the same information using different terms.

2. **Polysemy** Many terms have more than one distinct meaning. Users may use the same terms in different context. So when a user mentions a term it may or may not refer to the the document containing that term.

Synonymy effects the Recall of the Retrieval System and Polysemy effects the precision of the system.

Other limitations of this model stem from the nature of this model, Order of terms is not considered in this model since its a bag of words model and the context of the terms is not modelled here because of the unigram nature of the model. The term document matrix is high dimensional and Sparse making it suspectible to Noise.

We hope to reduce some of these limitations and improve the performance of the retrieval system by using ESA and Latent Semantic Indexing.

Another aspect of retrieval we wanted to address is the a way to improve the Model by using the feedback information from user, known as **relevance feedback** technique.

Relevance feedback identifies and utilizes, among documents from a retrieved set, those that are most relevant to the original query to improve the performance of next query by adjusting the model.

## 3 Problem Statement

As mentioned above the main goal of our project is to improve the basic vector space model. We intend to do so by considering semantic content and relevance feedback. We selected the below methods to improve the Basic Model.

1. **Latent Semantic Indexing**
2. Method which includes **relevance feedback** based on Supervised Learning
3. **Use Explicit Semantic Analysis** to better answer queries

From experimentation we intend to prove the below hypotheses

1. System based on Latent Semantic Indexing is better than the system with Basic Vector Space Model at Information Retrieval.
2. System which includes relevance feedback based on Supervised Learning is better than the system based on Latent Semantic Indexing at Information Retrieval
3. System which incorporates ESA is better system than the system based on naive tf-idf table

## 4 Background and related work

Latent Semantic Indexing we implemented is based on the paper here[1].In this paper the authors introduce the method which takes advantage of higher order associations between documents and queries to construct a semantic space. The documents and terms can be represented in this semantic space and similarities

can be measured. So we represent the queries in this space and use the similarities with documents to perform retrieval. In this paper the experiments were done on CISI and Medline Dataset.

In previous methods to include relevance feedback in retrieval system, the retrieval queries were adjusted, information retrieved in a relevance feedback process can hardly be preserved for a long term in the system and be used later. A technique of using the user's feedback information, User Lens[[2]] is a method where we re-weight the vectors which represent documents and/or queries in information retrieval systems, so that the vectors get automatically adjusted according to the user's relevance feedback. As a result, relevance feedback information from the user can be preserved for a long term in the system.

A supervised Learning model we implement is the model proposed in the paper here [3]. This is a method that utilizes information of documents that are relevant to the query. In addition to the user's relevance feedback of queries, information such as inter-document similarity values are also incorporated into the retrieval model built by using a sequence of linear transformations. Similar to the technique User Lens, the proposed method preserves feedback information for a long term in the system.

But it differs from the User Lens in the way it adjusts the model by using a sequence of linear transformations. In this paper the experimentation is done on Medline and Cranfield dataset.The queries here are divided into training and test queries. Training queries are used to build the relevancy retrieval model and test queries are used to evaluate it. The percentage of improvement in average precision is 28% on train queries and 4% on test queries in Cranfield dataset.

In ESA, we use a pre-trained machine learning model to map the terms/ documents of corpus to a weighted sequence of Wikipedia concepts of d-dimension (parameter) ordered by their relevance to the input. Using these Wikipedia synset vectors and a suitable similarity measure (cosine metric, movers distance) we find relatedness among queries and documents. For spelling correction in preprocessing, we intend to use dictionary models that takes context into account.

## 5 Proposed methodology

### 5.1 Basic Vector Space Model

**Formulation** Consider a document $d_j$ $(1 \leq j \leq n)$where n is the number of documents, it is represented as vector of weights in the term space

$$d_j = (w_{1j}, ....w_{tj})^T$$

where $w_kj$ is the weight (or importance) of term k in the document $d_j$, and t is the size of the indexing term set. A collection of n documents is then represented

by a t × n term-document matrix D:

$$D = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{t1} & w_{t2} & \dots & w_{tn} \end{bmatrix}$$

and when we represent the queries in the same space we get a similar matrix Q

$$Q = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1l} \\ q_{21} & q_{22} & \dots & q_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ q_{t1} & q_{t2} & \dots & q_{tl} \end{bmatrix}$$

where l is the size of the query vector set. When we are given a query $q_i$:

$$q_i = (q_{1i}, , ..., q_{ti})^T (1 \leq i \leq l)$$

retrieval is achieved by measuring the cosine similarity between a document and a query in the underlying vector space:

$$sim(d_j, q_i) = \sum_{k=1}^{t} (w_{kj} q_{ki})$$

Assuming that the D and Q matrices are processed,ie(columns divided by their norms) such that the above summation gives the cosine similarity. In addition, when we are given a collection of documents D, and queries Q, the resulting similarity matrix is given by:

$$S = D^T Q$$

We use this similarity matrix to rank the documents retrieved for each query.

### 5.2 Latent Semantic Indexing

**Introduction** Latent Semantic Indexing is variant of Vector Space Model aimed at addressing the problem of Synonymy and polysemy.

In this method, we assume there is some underlying latent semantic structure in the data that is not completely relied on the word choice of the document, query. We use statistical technique called **Singular value Decomposition(SVD)** to estimate this latent structure.

We take the term-document matrix **D**(mentioned in the Basic Vector Space model) and construct a "semantic" space wherein terms and documents that are closely associated are placed near one another.

Singular-value decomposition is used to construct this space by considering major associative patterns in the data and ignoring smaller associations. An implicit higher-order (or latent) structure in the association of terms and documents is considered in constructing this space. As a result, terms that did not actually appear in a document may still be closely associated to the document in the semantic space. Retrieval for a particular query is then done by representing this query in the semantic space along side documents.

**SVD and Formulation** SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors, each term and document is represented by its vector of factor values. SVD helps in replacing the representations using individual terms by representations using these orthogonal factors.

The factors can be thought as artificial concepts which represent the extracted meaning components of the words and documents. The factor values define the strength of association between these meanings and the words or documents. So when we consider k factors, the words and documents can be represented as vectors of k factor values in this dimension reduced semantic space.

The t × n term-document matrix D:

$$D = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{t1} & w_{t2} & \dots & w_{tn} \end{bmatrix}$$

is analysed by the singular value decomposition(SVD) to derive the latent semantic structure.

In SVD, a rectangular matrix is decomposed into three matrices of very special forms. These matrices contain "singular values" and "singular vectors". In this case

$$D = USV^T$$

U and V are the matrices of left and right singular vectors S is the diagonal matrix of singular values and in decreasing order of their magnitude.

The first k values are kept and others and others are set to zero and singular vectors corresponding to zero singular values are ignored. When we reconstruct the matrix with the modifications it is as below.

$$D_k = U_k S_k V_k$$

where $U_k, S_k, V_k$ are the matrices formed after modifications.
$U_k(t,k), S_k(k,k)$ and $(V_k)(k,n)$ dimensions and observe that $V_k = V^T[:\ k,:]$, $U_k = U[:,:k]$.

This resulted matrix is the rank-k model with best possible least squares fit to D. The value of k plays a important role in the retrieval performance. we want a value of k that is large enough to fit all the structure in the data, but small enough so that we do not also fit the unreliable features which do not participate in the semantic structure.

The $U_k$ and $V_k$ contain the representations of terms and documents respectively in k-dimensional semantic space. $V_k$ has the document representations, ie vectors of k-values, these can be used in computing similarity from a query

representation in the same space.

$$V_k = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1} & w_{k2} & \dots & w_{kn} \end{bmatrix}$$

To represent a query in this space we take t x l matrix Q:

$$Q = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1l} \\ q_{21} & q_{22} & \dots & q_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ q_{t1} & q_{t2} & \dots & q_{tl} \end{bmatrix}$$

and apply the transformation

$$Q_k = (Q^T U_k S_k)^T$$

where $Q_k$:

$$Q_k = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1l} \\ q_{21} & q_{22} & \dots & q_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ q_{k1} & q_{k2} & \dots & q_{kl} \end{bmatrix}$$

the columns are the query document factor vectors in the semantic or concept space. Now we compute the similarity matrix as Follows

$$S_{lsi} = (V_k)^T Q_k = (D_k)^T Q$$

We use this similarity matrix to rank the documents retrieved for each query.

### 5.3 Supervised Learning Model which includes relevance feedback

**Introduction** In this method we try to improve the retrieval performance by utilizing information of documents that are relevant to the query. In addition to this relevancy information we also use inter-document similarity values are also incorporated in the model.

This method uses a sequence of linear transformations to adjust the model so that the incorporated relevancy information can be preserved in the system for a long term. This a supervised approach, consider the t x l Query Matrix Q and t x n document matrix D. From the relevancy information we construct a matrix A. $A = a_{ij}$ where $a_{ij}$ is 1 if document i is relevant to query j or else 0. We assume this matrix A, can be obtained by the following linear transformations involving a matrix X.

$$M = g(Q) = XQ, A = f(M) = D^T M$$

from above transformations $A = L(Q) = f \circ g(Q) = D^T XQ$. The matrix X acts as a way to incorporate relevance feedback into the system. Note the Following matrix sizes

$$A(n,l) = D^T(n,t)X(t,t)Q(t,l)$$

In the course of the linear transformation L the query vector is transformed from the term space to the relevance documents vector space. After constructing A, we find X satisfying the above transformation.

**Finding X and Inter Document Similarity** As you can see the above method is based on finding the matrix X by solving two linear transformations f and g. First we solve the transformation $A = D^T M$ and then $M = XQ$.

Note that these Linear equations don't have a straightforward solution every time, i.e. using inverse. So we have to tolerate some error in order to arrive at a satisfying model.

This problem is famously known as least squares problem

**Problem:**

When AX = B, we need to find a solution X* such that the frobenius norm of the error matrix is reduced ie, minimize $(||(B - AX*)||_F)^2$

This problem can be solved using QR decomposition or SVD decomposition of A.Here we used QR decomposition to solve this problem.
minimize $(||(A - D^T M^*)||_F)^2$ , $A = D^T M$, solution M*

1. QR decomposition, $D^T = qr$
2. $y = q^T A$
3. $M^* = (r^-1)y$

Use this M* to solve the following problem. $M = XQ \sim M^T = Q^T X^T$ .
minimize $(||(M^T - Q^T X^T)||_F)^2$ , $M^T = Q^T X^T$, solution M*

1. QR decomposition, $Q^T = qr$
2. $y = q^T M^T$
3. $(X^*)^T = (r^-1)y$

We take a training set of Queries, form A and find X and if we try to calculate the Similarity matrix as follows.

$$S_{sup} = D^T XQ$$

The training data gives 100% precision but the test data performance is abysmal. Consider the Precision Recall curves of training and test data in 1. To deal with this case of overfitting we use Inter document similarity Matrix,ie $D^T D$ in the model. Consider the similarity matrix of basic vector space model,

$$S \circ D^T D = (D^T Q) \circ (D^T D) = D^T (Q \circ D)$$

We extend this to the above model, While Training

$$A \circ D^T D = D^T X(Q \circ D)$$

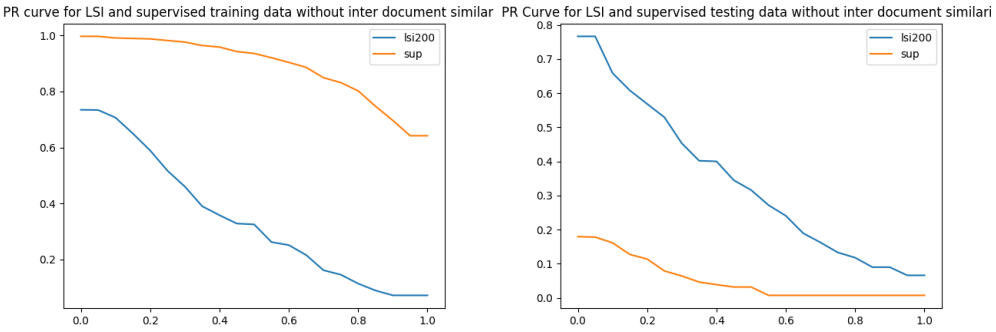


**Fig. 1.** This figure shows the Precision recall curves for training and testing data when inter document similarity is not considered in the model.

So our model similarity matrix is

$$S_{sup} = D^T X(Q \circ D)$$

The dimension of this matrix is (n, l+n) so we only consider first l (number of queries) coloumns as the relevancy scores of documents for each query for retrieval.

**SVD and Formulation** We addressed the problem of inability of capturing underlying semantic structure of Term Document Matrix D in LSI. So we use SVD to take advantage of the implicit higher-order structures in term document association to further improve this model. Since we are essentially building upon the LSI, it becomes a perfect baseline for this model. The SVD decomposition

$$D = USV^T$$

Now we consider the largest k singular values.

$$D_k = U_k S_k (V_k)^T$$

k is the value of number of dimensions in the semantic space. $V_k$ is the representations of Documents in this space and Q is the vector representation of Queries. When the dimensional space in which we represent is changed the model can be changed appropriately.

$$A \circ (V_k)^T (V_k) = (V_k)^T X(Q_k \circ V_k)$$

The model similarity matrix is as follows

$$S_{sup} = (V_k)^T X(Q_k \circ V_k)$$

,

### 5.4 Incorporating ESA into our model

**Introduction** In this method we try to incorporate external knowledge into our model to better understand synonymy rather than just simple properties like different words belonging to same document are similar.

**Word2Vec Model** Word2Vec models are used to construct vector embedding to words from corpora using neural networks. These embedding capture contextual closeness properly.
There are two different methods to do so

**Common Bag of Words(CBOW) Model** This model is simpler compared to other, it uses single/multiple input word(s) to get the context which is same length vector of original. The neural-network has single hidden layer and no activation function. The non-linearity in the context is captured by softmax in the output layer.

**Skip-Gram Model** This model has similar network but it takes only one input word and produces a 'n' probability distributions of same size as the input. This network has more parameters and takes longer to train but works better if amount of data available is less.

**GloVe Model** In contrast to Word2Vec this algorithm computes whole term co-occurrence of the corpus. Word vectors are obtained by factorization of the matrix.
We will be using pretrained Word2Vec(Skip-Gram Model) and GloVe models as they are known to work better for IR. Cranfield has 1400 odd aerospace related documents. It would have been better if there is a wiki-dump of aerospace related articles readily available, so we can train our models on it. Nevertheless there are veriety of models triained on Wikipedia, webpages and news articles like
1. en-crawl(Word2Vec) - Trained on 2017 Wikipedia, web crawls and news
2. glove-wiki(GloVe) - Trained on 2014 Wikipedia and Gigaword 5
Each doc has different number of words hence we get 2d vectors with different first dimension. Finding relatedness using this representation is not possible so we use **word centroid similarity (WCS)** approach. In this approach each document is represented as a centroid of all it's words in the same dimensional space. Queries are represented similarly and relatedness can be found intuitively by cosine similarity.
While using context knowledge from external sources we get much more information than mere direction which is used in cosine similarity. For example in most of representation $man - woman \simeq king - queen$. But we can't use euclidean distance which has its own drawbacks. To address this problem we use -

**Word Mover's Distance(WMD)** The idea comes from Earth Mover's Distance (EMD) which solves the problem - m trucks and n warehouse, we need to minimize the shipping of all goods. A similar approach to word relatedness will

overcome synonym problem. WMD finds the distance between two documents A and B is calculated by the minimum cumulative distance that words from the document A needs to travel to match exactly the points of document B. If we look at Fig.2. we can see that external knowledge puts similar words like 'obama' and 'president' near and mover's distance finds the cost to map it to a nearest word from other document. This method with knowledge encoded in the word2vec / Glove space seen to achieve high retrieval accuracy.

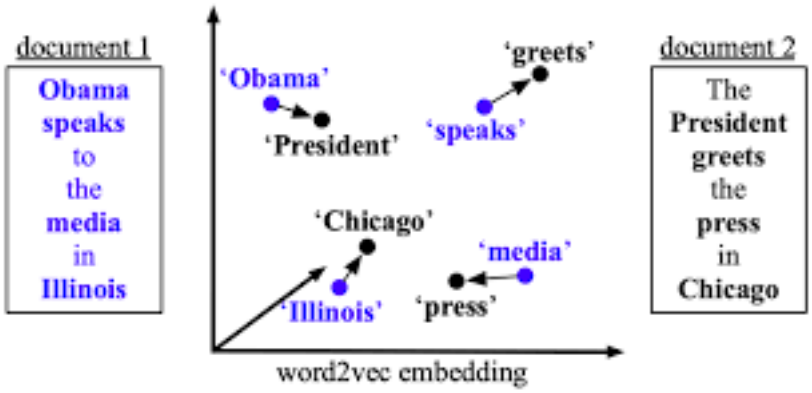


**Fig. 2.** Google Images

## 6 Experiments

In all the experiments we tested the retrieval systems using Cranfield dataset.

### 6.1 Latent Semantic Analysis

**Latent Semantic Indexing** The Basic Vector Space Model is the baseline for testing LSI on cranfield dataset.

**Preprocessing of Document and Queries**

We perform the same preprocessing for both systems. Preprocessing includes sentence segmentation, tokenization, inflection Reduction and stopword removal. For tokenization we used Penn-Tree Bank tokenizer from nltk library, Stemming is used as a way to reduce the inflection among tokens. Nltk list of english stopwords are used to do stopword removal.

Cranfield Dataset contains 1400 number of Documents and 225 queries. After performing stemming using Porter Algorithm we are left with 7029 terms. We remained cautious about NULL queries and NULL documents while evaluating the model.

**Term Weighting**

In Basic vector Space model we build the index using these terms and using this index we extract term space representations of documents and Queries by performing some term weighting use a weighting method.

Term weighting method consists of Two measures one local and other global. Local measure used in this case is term frequency(tf) in the document and global measure used is inverse document frequency(idf)

**Term Weight** $w = tf * idf = tf * log_{10}^{n/n_t}$

Here n is the number of documents and $n_t$ is the number of documents in which term t occurs.

D and Q with coloumns as vector representations of documents and queries respectively. To compute similarity of basic vector space model, we divide each coloumn of D and Q with coloumn norm to deal with different lengths of document and query. In LSI method uses the same above mentioned term weighting and recompute D and Q.

SVD is used to get the representations of Documents and Queries in semantic or concept space. As mentioned in the formulation of LSI, we use these representations to compute similarity matrix, the same operation dividing each coloumn by its norm to deal with the different document lengths.

We consider the first 20 among the ranked retrieved documents to calculate precisions, recalls, MAP, F score and nDCG. We then use these results to perform interpolation and plot precision Recall curve.

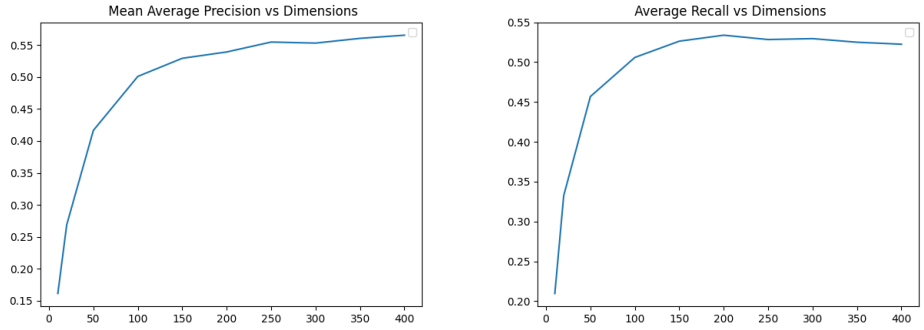


**Fig. 3.** The plot on the left side, shows the variation of Average Precision with respect to number of the factors and the plot on the right side, shows the variation of Recall with respect to number of the factors.

The SVD expects a parameter the number of dimensions in the constructed semantic space or number of factors in the representations. This parameter has substantial effect on the system performance, so we experimented on number of factors ranging from 10 to 400. LSI with more than 400 number of factors is not economical, even though the average precision of the system increases as number

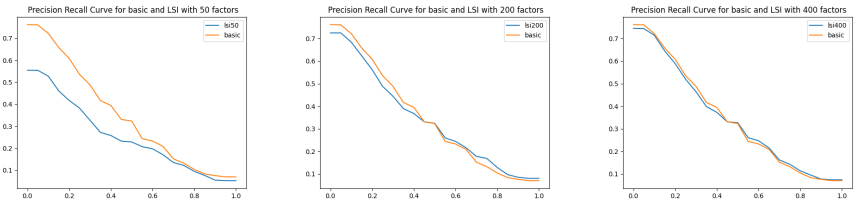


**Fig. 4.** The Precision Recall Curve Comparisons of Basic Vector Space model and LSI model with 50, 200, 400 number of factors from left to right.

of factors imcrease. 200 factors gave a better recall [3] and better Precision Recall curve when compared to others[4].

**Supervised Learning Model** This Model can be viewed as a extension of Latent Semantic Indexing, making it the perfect baseline to compare the performance of the system with.

Even in this experiment, the same Preprocessing and Term weighting mentioned in LSI experiment is used. The cranfield dataset of 1400 documents and 225 queries is used. Since this is a supervised learning model based on queries, we need to divide the queries into two disjoint parts. Training data consisting of 180 queries and Testing data consisting of 45 queries.

The relevancy information of cranfield dataset is used to construct A. We perform SVD with 200 factors and obtain semantic space representations of Documents required for both the baseline LSI and proposed model. Training and Testing Query semantic space representations can also be obtained respectively by the method mentioned in formulation.

In the Formulation it is mentioned that the elements in matrix are 0 or 1. We introduce a parameter w for further experimentation of the model.

$$A = wA$$

Training is done with the updated relevance matrix. We need to find a w value with which the model performs better. As w increases, the training data retrieval average precision increases because the relevancy increases[5]. But with strange or test data it will not be the case. Since we don't want to experiment on test data, conventionally a development or holdout data containing 18 queries among the training data is used to experiment on this parameter.

The best w which maximizes average precision of the holdout data is intended to perform better in the test data. When we experimented on w from 0.1 to 10. $w = 2.0$ is found to be performing better in holdout data. See [5]

Note that even in this experiment to compare the two systems we consider first 20 of the ranked documents. We calculate precision, recalls, MAP, F score and nDCG. Precision Recall curve is then constructed followed by these calculations. Mainly Precision Recall curve is used to perform comparisons.

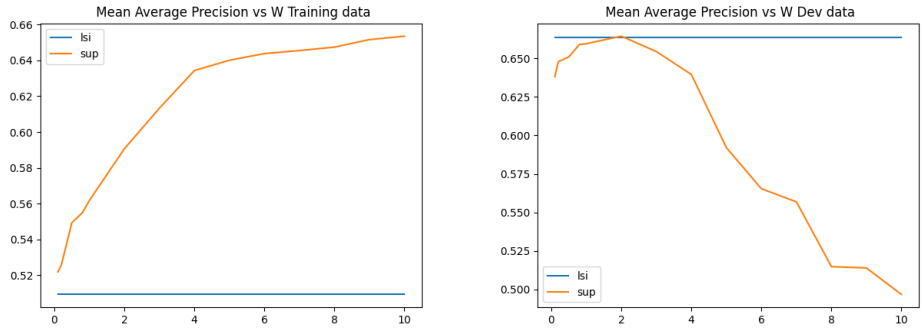


**Fig. 5.** The left side plot is the Variation of Average Precision with w for training data and right side plot shows the Variation of Average Precision with w for Development data

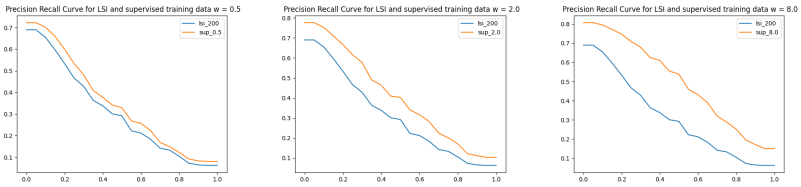


**Fig. 6.** These plots show Supervised Model with w values [0.5,2.0,8.0](left to right) system performance comparison using precision recall curves for training data


**Fig. 7.** These plots show Supervised Model with w values [0.5,2.0,8.0](left to right) system performance comparison using precision recall curves for development data

### 6.2 Explicit Semantic Analysis

We intend to retrieve embeddings of words from pretrained models described in section 5.3 and use word centroid to represent documents. These vectors normalized and combined with standard tf-idf model known to give good retrieval accuracy. These models only work if the input word has no spelling errors. To make this possible first we need to use lemmetizer instead of stemmer in inflection reduction process and also use a spelling corrector at the end of pre-processing pipeline.
After a lot of experimentation we came up with 5 different versions of ranking pre-processed documents.

**v2** In this approach preform cosine similarity between centroid of words embedddings produced from pretrained en-crawl word2vec model joined with tf-idf representation.

**v3** This is similar to above but we use wiki-glove data-set

**v4** Same as above but here we calculate individual cosine similarities of tf-idf, en-crawl, wiki-glove and summation of cosine similarities of these values

**v5** In this version we calculate heuristic value which is summation of word movers' distance and $\alpha$(parameter) the cosine similarity of tf-idf. Upon experimentation better results are obtained when $\alpha = 2, 1.5$. The best methods to perform wmd have the complexity of order $O(n^3 \log(n))$ where is $n$ is length of longest of all documents. This method is computationally expensive so experimentation is done for some-what lesser number of queries. The embeddings for calculating wmd is taken from glove-wiki model.

**v6** Similar to v5 but embeddings are taken from pretrained fasttext model.

| (k=10) Measure Version | Description | Precision | Recall | F-Score | MAP | nDCG |
|---|---|---|---|---|---|---|
| tf-idf | Simple tf-idf model | 0.2978 | 0.4246 | 0.324 | 0.6722 | 0.74 |
| v2 | tf-idf + en-crawl | 0.2991 | 0.4278 | 0.3255 | 0.6871 | 0.7533 |
| v3 | tf-idf + GloVe | 0.2942 | 0.4253 | 0.3223 | 0.6775 | 0.7476 |
| v4 | tf-idf + en-crawl + GloVe | 0.2929 | 0.4228 | 0.3206 | 0.6796 | 0.7518 |
| v5 | tf-idf + wmd(GloVe) | 0.305 | 0.4702 | 0.3346 | 0.7065 | 0.777 |
| v6 | tf-idf + wmd(fasttext) | 0.305 | 0.4694 | 0.3341 | 0.6959 | 0.772 |

**Fig. 8.** Small subset of results depicting improvement in different measures

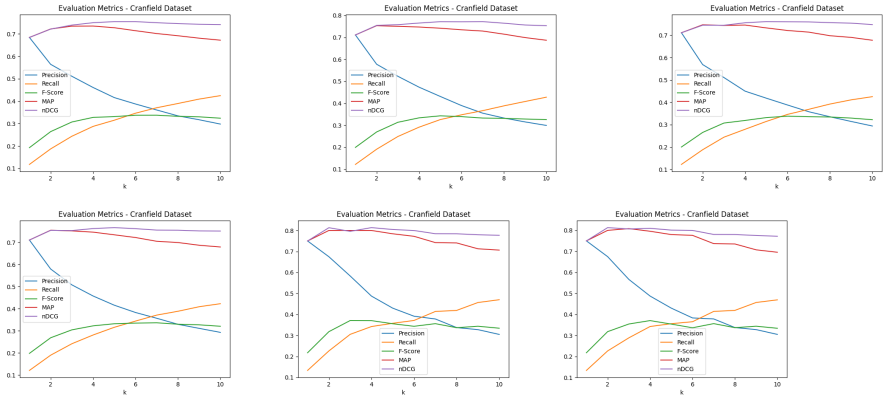


**Fig. 9.** Evaluation Plots of different versions in the order mentioned in the text

# 7 Results

## 7.1 Latent Semantic Indexing

The number of factors plays an important role in the LSI retrieval performance. If the number of factors is very low , the semantic structure any not be captured well enough. If the number of factors is very high, the factors the model may start capturing noise instead of the associations of term documents we hope to capture. One of the features of LSI is that it is computationally economical, so 400 is used as the maximum for number of factors.

From the [3], we can observe that the average precision of the system increases as number of factors increase. System has 0.25 average precision with 10 factors and around 0.52 with 200 factors, it almost doubled.

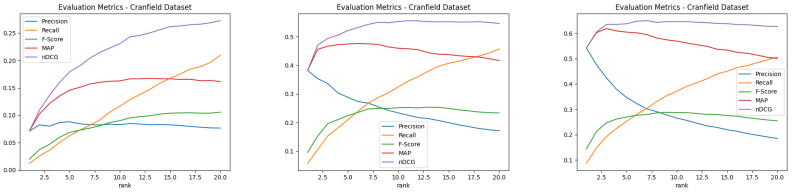


**Fig. 10.** These are the evaluation plots of different evaluation metrics for LSI system with 10,50,100 number of factors (left to right).

Now observe the eval plots in 10, as the number of factors are increased you can clearly see the increase in system's performance . All evaluation metrics indicate that the System with higher number of factors in this range is better. When eval plots in 11 are observed there is no enough difference in the evaluation

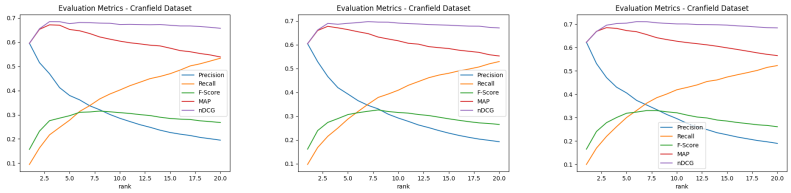


**Fig. 11.** These are the evaluation plots of different evaluation metrics for LSI system with 200,300,400 number of factors (left to right).

metrics which indicates that one system is better than other. Recall over the range of number of factors is captured in [3], This indicates system with 200 factors has better recall.

From the [4], The LSI system with 200 factors seems to have a better Precision Recall curve when compared to 50 and 400 factor systems at high recall. Consider the Precision Recall Curve Comparison between Basic Vector Space

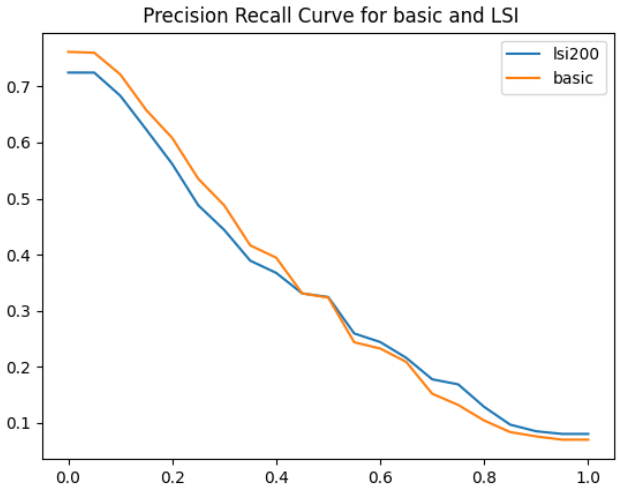


**Fig. 12.** This shows the precision recall curve comparison for Basic VSM and LSI with 200 number of factors.

Model and Latent Semantic Indexing System with 200 factors [12]. LSI system lies above the basic VSM system for recall greater than 0.5.

At lower recalls , precision is higher among all systems. Synonymy is dealt better compared to Polysemy in LSI systems, this improves Recall. At low recalls, ie at the highest ranked documents there may be relevancy with the words which

exhibit synonymy. So the effect of LSI dealing with Synonymy is really seen at higher recalls. We can state that LSI system perform better at higher recalls when compared to basic VSM because it deals with Synonymy problem.

When the number of factors is increased, the precision of system increase which may result in improvement of performance at low recall but the performance at high recall is reduced[4].

### 7.2 Supervised Learning Model

We introduced a parameter w in the model to experiment on the system's performance. For the training data, the system performance increases as w increases[6] to get a value which performs best when the system is tested with strange query, we use holdout data.

From [5], the holdout data indicate that $w = 2.0$ is the optimal value to get a better system performance. This can also be observed from the PR curves in [9], where $w = 2.0$ system has better performance. The eval plots [13][14],indicate the same. The average precision of the system is poor with the development data when we increase w further than 2.

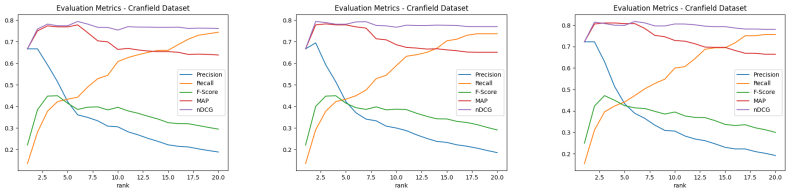


**Fig. 13.** These are the Development data evaluation plots of different evaluation metrics for Supervised Model with w values 0.1, 0.5, 2.0 (left to right)

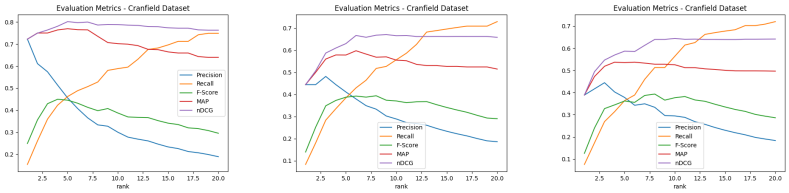


**Fig. 14.** These are the Development data evaluation plots of different evaluation metrics for Supervised Model with w values 4.0, 8.0, 10.0(left to right)

In case of training data, The Average precision for the LSI is 0.509 and the average precision of Supervised Learning Model is 0.59, around 20 percent

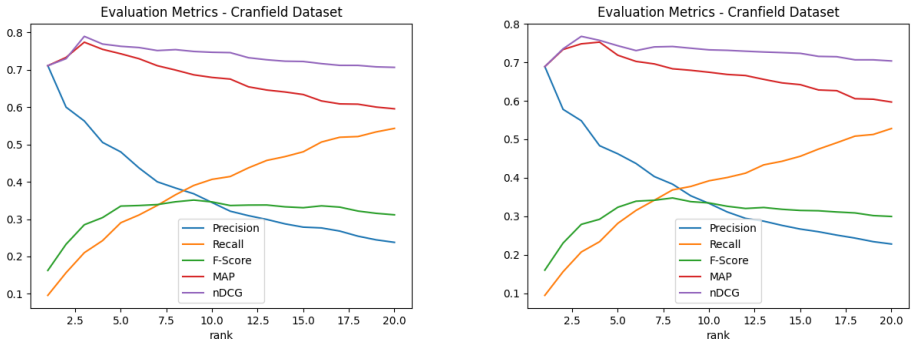


**Fig. 15.** These are testing data evaluation plots for Supervised model with w = 2.0 (left) and LSI with 200 number of factors.(right)

improvement in performance. For the testing data, the average precision at rank 1 for LSI and Supervised model are 0.688 and 0.711 respectively and at rank 10 the average precision are 0.674 and 0.679.

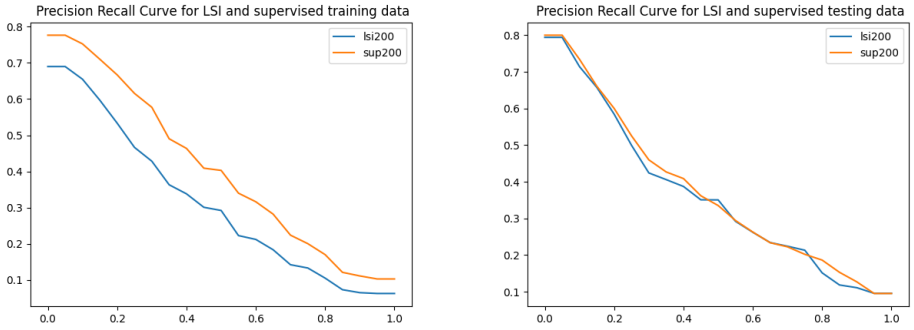


**Fig. 16.** The PR curve for performance comparison with Supervised Model and LSI Model for training data on the left and for testing data on the right.

The improvement on training data is more compared to the testing data. This improvement can be seen the precision recall curves for train queries and test queries in [16] .

### 7.3 Explicit Semantic Analysis

As stated in the hypothesis we can see different versions of ESA preforming better in different measures. The improvement is minute as most of these models are

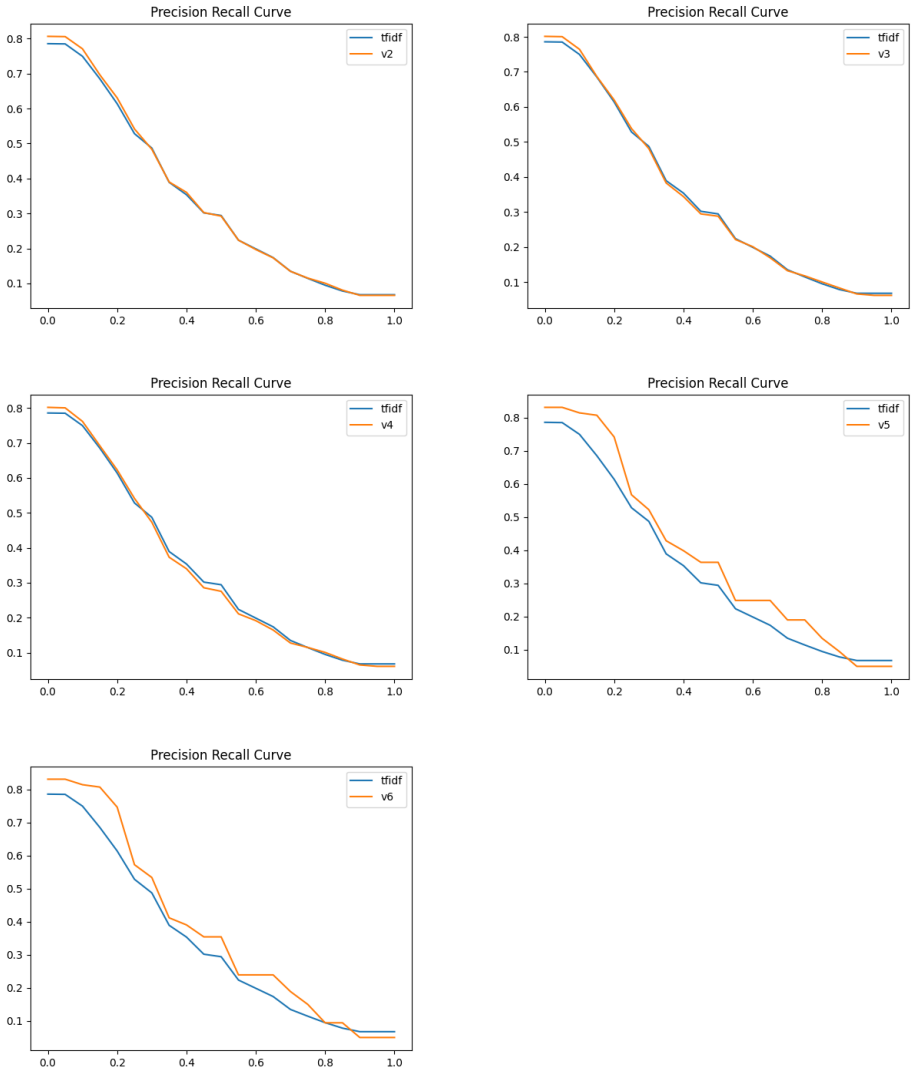


**Fig. 17.** Precision vs Recall comparison plot for different version w.r.t tf-idf model

designed to represent whole English vocabulary but the differentiating words in our documents are mostly aerospace related. Nevertheless we can improvement in retrieving the best docs (i.e. k = 1, 2) while using first 3 versions and an overall better retrieval using wmd in later versions. We can see the same in [17]

## 8 Conclusion

In the case of ESA, We can clearly see that there is a substantial improvement in retrieval when movers' distance is used instead of cosine similarity but it is computationally expensive. To cater this need there are different versions of wmd, recently, like relaxed-wmd etc some of them with complexity close to $O(n^2)$. The idea capturing relatedness using distance between nearest words in context space i.e. the idea of WMD is major take-away.

Introducing semantic content by Latent Semantic Indexing technique in Basic vector space model helped in the performance of model and in addressing the basic model's limitations. While the LSI method deals nicely with the synonymy problem, it offers only a partial solution to the polysemy problem. It helps with multiple meanings(polysemy) because the meaning of a word can be defined by the context words and some other associations and LSI can model that. The failure comes in the fact that every term has only one representation, so a term with more than one entirely different meaning is represented as a weighted average of these meanings. If the meaning used by the term in a query is not close to average meaning it may lead to imprecise results.

Latent Semantic Indexing is computationally economical, the representations used are only of length 200, whereas in basic vector space model the representations are of length 7000.

The implemented model to incorporate relevancy information(ie, user-supplied information of those documents that are relevance documents to the query in question) into the Latent Semantic Indexing model improved the performance of the retrieval system. The goal to preserve the feedback information from the user in system for a long term to adjust retrieval model has been successful.

## References

1. Deerwester, S., Dnmals, S., Furnas, G. W., Landauer, T. K. and Harshman, R. **Indexing by latent semantic analysis**.Journal of the American Society for Information Science.
2. Vogt, C. C., Cottrell, G. W., Belew, R. K. and Bartell, B. T..**User lenses-achieving 100% precision on frequently asked questions. Proceedings of User Modeling**.
3. Xiaoying Tai, Minoru Sasaki, Yasuhito Tanaka, Kenji Kita **Improvement of Vector Space Information Retrieval Model based on Supervised Learning**
4. Evgeniy Gabrilovich and Shaul Markovitch **Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis**
5. Matt J., Yu Sun, Nicholas I., Kilian Q. **From Word Embeddings To Document Distances**