Word Sense Induction and Disambiguation Using Principal Component Analysis and Latent Semantic Indexing

Vibhor Gaur, Dr. Satbir Jain

Dept. of Computer Engineering , NSIT, New Delhi

***Abstract*-** In this paper we present a statistical method using principal component analysis and latent semantic indexing to solve the problem of word sense induction and use the generated sense inventory to perform word sense disambiguation. We use standard co-occurrence graph algorithms and word dependency matrices (context words, dependency relations) and map them to a matrix. Then, we apply non-negative matrix factorization and principal component analysis on dimensions (latent factors) obtained from the training set. The intuition behind this is to merge dimensions that overlap in the semantic space (like computers and processors) and hence reinforcing their effect mutually to improve the disambiguation step. We work on the idea that each sense obtained in the induction process corresponds to a topical dimension. We extend this idea for each word to obtain a word's most dominant sense. The framework is tested on the standard SEM-EVAL 2010 for WSI/WSD on which it produces state of the art results.

***Index Terms*-** word sense induction, word sense disambiguation, principal component analysis, latent semantic indexing, non negative matrix factorization

I. INTRODUCTION

Word sense induction is the task of automatically identifying senses or meanings for words in the test corpus. Constructing a sense inventory manually is a time-consuming job, and the results are not objective as they are highly dependent on the annotators perception of the domain. By applying an automatic procedure, we are able to extract the senses objectively and intrinsically present in a particular corpus. The intrinsic factor covers any new domain not defined in any of the reference senses to be easily embeded in the sense inventory. (For eg. A new car named apple is launched in the market)

Word disambiguation on the other side is a process of assigning a meaning to the word in context of it's occurrence in the corpus from a sense inventory. Majority of the WSD algorithms use a supervised approach by using pre-defined sense dictionaries such as WordNet. The problems of domain and dynamic introduction of new senses become serious considerations while applying them in computational semantics. Also training these models by manual annotation requires a considerable effort.

The model presented in this paper takes an unsupervised approach to the problem by automatically inducing senses for words and using this sense inventory for unsupervised disambiguation of various occurrences of words (the same word) in different contexts. The induction step uses the standard co-occurrence graphs, word dependency relationships and principal component analysis along with non-negative matrix factorization. The induced senses are then mapped onto a semantic space with topical dimensions. In the disambiguation step we map the context in which the word(target word) occurs to the semantic space. The idea is that we combine the local approach of co-occurrence graphs along with the global approach of finding semantics in reference to the whole corpus which uses words, bag of words (context window) and dependency relations like synonymy for clustering senses. The intuition behind this is somewhat similar to topic modeling where the creation of each word is attributed to a topic. In a similar analogy, we consider a dominant dimension responsible for the creation of a word and similarly a distinct dimension inducing each of it's senses.

The layout of the paper is: Section 2 provides an overview of some previous work on word sense induction where we highlight on some of the key techniques that provide a framework for understanding .Section 3 presents the elaborate method of our model in a stage wise process and a working example to explain it's execution. Section 4 provides a comprehensive analysis of quantitative evaluation and comparison of our model with other algorithms in the framework of SEMEVAL-2010 WSI/WSD task. In, the last section we draw conclusions based on our results and throw some light on future work in this direction.

II. PREVIOUS WORK

A. Distributional Semantics

The main concept behind distributional similarity is that words that occur in similar contexts tend to be semantically more similar. As a result many algorithms have been proposed that utilize that this property by mapping documents and words to a semantic space through a vector model and compare their distribution.

A pioneering model in this respect is latent semantic analysis — LSA (Landauer and Dumais,1997; Landauer et al., 1998). In LSA, a term document matrix is created, which contains the frequency of each word in a particular document. This matrix is then factorised into three other matrices with a mathematical factorization technique singular value decomposition (SVD). The most important dimensions that come out of the SVD represent latent semantic dimensions, according to which nouns and documents can be represented more

efficiently. Another work in this direction is by Van de Cruys & Marianna Apidianaki (2011) which uses non-negative matrix factorization(NMF) instead of SVD. Our model also applies the same factorization technique but using it with Principal component analysis(PCA) and local co-occurence graphs which improves upon the dimensional considerations in his work by introducing a matrix based on graphical parameters and reducing the dimensionality(PCA) by merging senses that point nearly in the same direction in the semantic space.

Context is important in deciding the nature of semantic similarity. A broad context window yields broad topical similarity whereas a small window can help in obtaining tight synonym like similarity. As a result many researchers use dependency relations as the contextual features a particular word takes. An important approach in this regard is Lin (1998).

B. Word Sense Induction

The algorithms for word sense induction can be divided into two major categories local and global. Local WSI algorithms extract various senses of a word on a per-word basis, i.e. the different senses for each word are determined separately. They can be further subdivided into context-clustering algorithms and graph-based algorithms. In the context-clustering approach, context vectors are created for the different instances of a particular word, and those contexts are clustered, each representing a distinct sense of the word. The context vectors may be represented as first or second order co-occurrences (i.e. the contexts of the target word are similar if the words they in turn co-occur with are similar). First work done in this direction was by Sch¨utze (1998), and many researchers (Purandare and Pedersen, 2004) followed.

In the graph-based approach a co-occurrence graph is created, in which nodes are represent words, and edges connect words that appear in the same context (dependency relation or context window) representing co-occurences. The senses of a word are then discovered by using graph clustering techniques (Widdows and Dorow, 2002), or algorithms such as HyperLex (V´eronis, 2004) or Pagerank (Agirre et al., 2006) that find hubs in a graph. Bordag (2006) proposed an approach that uses word triplets to perform word sense induction.

Global algorithms,on the other hand try to find the different senses of a word by comparing them with the different senses of other words in a semantic word space model. The best known global approach is given by Pantel and Lin (2002). They present a global clustering algorithm – coined clustering by committee (CBC) – that automatically discovers word senses from text. The key idea is to initially find a set of unambiguous clusters, to which possibly ambiguous words can be assigned.After assigning a word to a cluster the features assosciated with that cluster are removed from the word vector,the remaining vector represents the other less known senses of the word and the process can continue iteratively.

Van de cruys & Marianna Apidianaki (2011) proposes a method for WSI based on latent semantic indexing using non negative matrix factorization , and later on extends it to do word sense disambiguation as well. The model proposed below incorporates the dependency features and broad contextual features with principal component analysis with local occurrence graphs (by mapping them to a matrix) and including them in the process of non-negative matrix factorization. We also use a different mapping model to map to semantic space, by using both local and global factors in our factorization to reduce effects of co-occurences that arise as a result of usage rather than any semantic relationships (like idioms).

III. METHODOLOGY

A. Local co-occurrence graphs

Initially we use various standard co-occurrence graphs ,to produce a matrix D of words cross classified by parameters that approximate a words' chances of being part of a component(sense) in the graph. We construct the graph for each word and augment the entry D_{ij} in the matrix by calculating parameter j for word i, this way we provide increased value to a word that occurs in similar context multiple times. Following are the steps as given by Navigli (1998)

1.For each word in the text corpus create a graph as follows

a.The nodes of the graph are all words obtained from context windows for each occurrence of word in the text corpus.

b.for each pair of words w and w_ in the graph calculate the dice coefficient Dice(w,w_)

$$\text{Dice}(w,w\_) = \frac{2c(w,w\_)}{c(w)+c(w\_)}$$

where c(w),c(w_) are occurrences of words w and w_ and c(w, w_) represents co-occurences.

c.First order co-occurences

In the above graph,we retain only those nodes that satisfy

$$\frac{c(q,w)}{c(q)} \geq \delta$$

$Dice(q,w) \geq \delta'$ where δ and δ' are experimentally tuned thresholds

d.Second order co-occurences

We augment the nodes obtained in step d) with words that co-occur with words in step c. that is satisfy both equations of step c.

d. Creating the co-occurrence graph: For each pair of words (w,w_) ∈ V × V, we add the corresponding edge {w,w_} to *E* with weight Dice(w,w_) if the following condition is satisfied:

Dice(w,w_) ≥ θ(where θ is a confidence threshold for the co-occurrence relation. Note that we use a threshold δ _ to select which vertices to add to the graph and we use a potentially different threshold θ for the selection of which edges to add to the graph. Finally, we remove from *V* all the disconnected vertices (i.e., those with degree 0).

e. Now our graph is ready for calculating the parameters for each word w in the graph. The parameters are:

1.curv(w)=#triangles w participates in/#triangles w could participate in
Triangles : cycles of length 3

2.sqr(w)=#squares w participates in/#squares w could participate in
Squares: cycles of length 4

3. dia(w) =#diamonds w participates in/#diamonds w could participate in
Diamonds: squares with a diagonal edge

4.connect(w)=#number of edges incident on w/# total number of nodes in graph

These parameters represent the tendency of a word to fall into a sense for the word w' (whose graph is being constructed).So, a less curvature indicates that the word is a bridge between two senses and a higher value indicates that it conveys a sense for the word.

f. Augment entry D_{ij} for each node(word) i and each parameter j of matrix D with above values ,i.e
$D_{i1}=D_{i1}$+curv(i) and so on for i=2,3 and 4

2.Go to step 1.

B. Non-negative matrix factorization

Non-negative matrix factorization (Lee and Sung,2000) factorises a matrice A into two matrices W and H such that

$A_{ij} \approx W_{ik} \times H_{kj}$

Matrices W and H are randomly initialized, and the rules in 2 and 3 are iteratively applied
– alternating between them. In each iteration, each vector is adequately normalized, so that all dimension values sum to 1.

$$H_{a\mu} \leftarrow H_{a\mu} * (\sum_I W_{ia} * A_{i\mu}/(WH)_{i\mu})/\sum_k W_{ka} \quad ..(2)$$

$$W_{ia} \leftarrow W_{ia} * (\sum_\mu H_{a\mu} * A_{i\mu}/(WH)_{i\mu})/\sum_v H_{av} \quad ..(3)$$

There are many advantages of using this in our context as we can represent the k latent dimensions (senses) in the semantic space with the help of W and H. It is better in comparison to the standard known techniques of SVD .NMF minimizes the Kulback-Liebler divergence which is again more suitable for language phenomena than the Euclidean distance that serves as a minimizing criterion for SVD. The non-negative property of NMF allows us to interpret the result probabilistically. But the problem of using this over SVD is that the dimensions obtained are not orthogonal and hence semantics of same dimensions (like apples and fruit) are mapped differently. We use principal component analysis to combat this. Elaborate analysis will be done only after the NMF step.

C. Latent Semantic Indexing (using NMF)

We use an extension of non-negative matrix factorization to induce latent factors for matrices capturing semantics in the document. We use 4 matrices instead of 3 as used by Van de Cruy, 2008. The first 3 matrices remain the same. The first matrix contains co-occurrence frequencies cross-classified by their (appearing in context window of the word).The fourth matrix consists of words cross classified by their parameters obtained from the local co-occurrence graphs earlier. This fourth matrix helps us to incorporate the effects of various local senses that are better captured through graphs. Next we apply NMF on the 4 matrices and interleave the factorizations (initializing the matrix in the next factorization, which is same as that updated before with the results of the former factorization).This has been illustratred in the table below:

Abbreviations used:

Decomposed Matrix1/Decomposed Matrix 2:DCMP1/DCMP2

Matrix generated from previous step/Matrix generating this matrix: MGPS/MGTS

Matrix updated in this step/Equation Used: MUTS/EU

Words/Dependency Relations/Dimensions/Context words/parameters: W/DR/D/CW/P

Matrix	DCMP1	DCMP 2	MGP S/MG TS	MUTS /EU
A (W × DR)	L (W × D)	M (D × DR)	M/Q	L/3
B (W× CW)	N(W × D)	O (D × CW)	N/L	O/2
C (CW× DR)	P (CW × D)	Q (D × DR)	P/O	Q/2
D (W×P)	R (W × D)	S (D × P)	R/L	S/2

Formally the process can be described as: we consider the 4 matrices to be A, B,C and D and their factorizations to be L,M; N,O; P,Q; R,S. Initially matrices L, M,O, Q are initialized randomly. Now, we carry out the first iteration in which we compute the update of L using M(equation 3).This matrix is copied into matrices N and R. Next we compute the update of O using N (equation 2) and the update of matrix S using R(equation 2). Next we copy the matrix O into matrix P. Finally, we compute the update of matrix Q using matrix P (equation 2).The iteration is then repeated and we continue to iterate until a specific number of iterations have been performed.

When the factorization is complete the 4 different modes(words, context words, dependency relations, parameters) are all represented by latent factors. But the process is incomplete in the sense that these latent factors are correlated which does not provide us with discrete senses but senses that overlap in the semantic space. As a result, the semantic space is still diffused and cannot be used for mapping words and senses to the semantic space. To address this concern we use principal component analysis of the latent factors as described in the next sub-section.

D. Principal Component Analysis

Principal component analysis (PCA) is a mathematical procedure that uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components.

In our case the correlated variables are the latent dimensions induced in the NMF step. We apply the PCA process to each of the 8 matrices obtained in the NMF factorization step, the method however remains the same. In this way we reduce the dimensionality of each matrice from k to k' where k' are the reduced number of linearly uncorrelated variables(latent dimensions) obtained after the application of PCA.

Let us consider one of the matrices and demonstrate application of PCA. Let the matrix be matrix L which is words cross classified by latent dimensions and is W × K (W:number of words, k:number of dimensions).Our aim is to convert this to a matrix L' of dimensionality W × K' where k' will be chosen as guided by the algorithm. Following is the standard covariance method to compute the PCA (WIKIPEDIA).

Calculate the empirical mean

- Find the empirical mean along each dimension j = 1, ..., k.
- Place the calculated mean values into an empirical mean vector **u** of dimensions k × 1.

$$u[j] = \frac{1}{W}\sum_{i=1}^{W} L[i,j]$$

Calculate the deviations from the mean

Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data.Hence we proceed by centering the data as follows:

- Subtract the empirical mean vector u from each row of the data matrix X.
- Store mean-subtracted data in the $w \times k$ matrix B.

$B=X-h\mu^T$

where h is an $w \times 1$ column vector of all 1s:

h[i]=1 for i=1….w

Find the covariance matrix

Find the $k \times k$ empirical covariance matrix C from the outer product of matrix B with itself:

$$C = \frac{B^* \cdot B}{w-1}$$

where

*is the conjugate transpose operator. Note that if B consists entirely of real numbers, which is the case in many applications, the "conjugate transpose" is the same as the regular transpose

- Please note that outer products apply to vectors. For tensor cases we should apply tensor products, but the covariance matrix in PCA is a sum of outer products between its sample vectors; indeed, it could be represented as $B^*.B$. See the covariance matrix sections on the discussion page for more information.
- The reasoning behind using W-1 instead of W to calculate the covariance is Bessels' Correction

Find the eigenvectors and eigenvalues of the covariance matrix

- Compute the matrix **V** of eigenvectors which diagonalizes the covariance matrix **C**:

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$$

where **D** is the orthogonal matrix of eigenvalues of **C**. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues . These algorithms are readily available as sub-components of most matrix algebra systems, such as R, MATLAB,Mathematica,Scipy ,IDL (Interactive Data Language), or, GNU Octave as well as OpenCV.

- Matrix D will take the form of an $M \times M$ diagonal matrix, where

$$D[k,l] = \lambda_k \qquad \text{for } k = l = j$$

is the jth eigenvalue of the covariance matrix C, and

$$D[k,l] = 0 \qquad \text{for } k \neq l.$$

- Matrix **V**, also of dimension $p \times p$, contains p column vectors, each of length p, which represent the p eigenvectors of the covariance matrix C.
- The eigenvalues and eigenvectors are ordered and paired. The jth eigenvalue corresponds to the jth eigenvector.

Rearrange the eigenvectors and eigenvalues

- Sort the columns of the eigenvector matrix **V** and eigenvalue matrix **D** in order of *decreasing* eigenvalue.
- Make sure to maintain the correct pairings between the columns in each matrix.

Compute the cumulative energy content for each eigenvector

- The eigenvalues represent the distribution of the source data's energy among each of the eigenvectors, where the eigenvectors form a basis for the data. The cumulative energy content g for the jth eigenvector is the sum of the energy content across all of the eigenvalues from 1 through j:

$g[m] = \sum_{k=1}^{j} D[k,k]$ for j=1….k

Select a subset of the eigenvectors as basis vectors

- Save the first k' columns of V as the w × k' matrix L':
- L'[r,l] = V[r,l] for r=1…..w l=1…..k'

Where

1≤L≤k

- Use the vector g as a guide in choosing an appropriate value for L. The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis. For example, you may want to choose L so that the cumulative energy g is above a certain threshold, like 90 percent. In this case, choose the smallest value of L such that

$$\frac{g[k']}{g[k]} \geq 0.9$$

E. Word Sense Induction-Generating the sense inventory

Now that we have induces latent factors over our matrices (representing semantic relationships) and reduced their dimensionality using PCA to obtain broader senses to which words can be assigned, our task is to create a sense inventory by clustering words using their vectors in this reduces dimensional space. The word vectors are obtained from rows of matrice L' (dimensionally reduced form of L words*dimension).

We use k-means clustering as PCA automatically projects to the subspace where the global solution of K-means clustering lies, and thus facilitates K-means clustering to find near-optimal solutions. K-means yields a hard clustering in which every noun will be assigned to one dominant cluster. In order to assign this noun to different clusters we subtract the salient dimensions(vector) of the centroid of the cluster(which is presently being considered in an iterative loop) from the word vector of this noun and check if it can be assigned to a the next cluster. The salient dimensions of the centroid are computed by averaging word vectors of all the elements in the cluster except the element which we want to reassign.This word vector is then fed again to the clustering algorithm. If no reassignment takes place it means all senses for the word have been discovered and it has been assigned to all clusters that convey one of it's senses. By subtracting the centroid from the word vector we strip of it's dominant senses to discover it's other less discovered senses.

We use the liberal approach of assigning the word vector to a new cluster. In this approach, the next best cluster(candidate sense) is selected for the target noun untill a certain threshold similarity is reached. We use the cosine similarity as a criterion for measuring the similarity. After performing these steps we have obtained a sense inventory of candidate senses, each represented by a cluster and characterized by the centroid of the cluster. This sense inventory will facilitate our process of word sense disambiguation in which we assign a sense to a word depending on it's context.In the next section we carry out the process of WSD and illustrate it with an example.

F. Word Sense Disambiguation

In order to find the sense of a given word in context of it's occurrence in the document we need to create the context vector v for this word (using it's context window). We then need to map the senses(clusters) obtained in the previous step to the semantic space. We then follow the same procedure and map the context vector v of the target noun to the same semantic space and obtain v'. Finally, we compare the mapping of v with the mapping of each cluster obtained earlier using kulback leibler divergence. The target word is assigned to the cluster giving minimum divergence. Now, to assign the final sense, we find the most dominant dimension in this cluster and assign it to the word.

We consider two separate mappings, viz; global and local. For mapping to the global semantic space we use matrix M (dimensions*dependency relations) and for mapping to the local semantic space we use the matrix S(dimensions*parameters).Let us first consider mapping for each of the clusters C_i. In order to map the cluster to the semantic space we use it's vectors C_{i1} and C_{i2}.C_{i1} is a row vector and is equal to the average of all the dependency relation vectors (rows of matrix A, words*dependency relations) of all the words in the cluster.C_{i2} is also a row vector and is equal to average of all the parameter vectors (rows of matrix D, words*parameters).The two mappings for the cluster are represented as M_1C_i and M_2C_i

$$M_1C_i=C_{i1}*M^T$$
$$M_2C_i=C_{i2}*S^T$$

Now we need to map the target noun (whose sense is to be found).For global mapping of the context we use the matrix O(dimensions*context words) and for local mapping we use the same matrix S as used for clusters. Again now we need to construct two matrices W_1 and W_2.W_1 is a row vector representing the context of the word. It is calculated by counting the number of occurrences of each context word(same as in matrix O) in the context window of the target word.W_2 is also a row vector representing the parameters of the target word. We will need to calculate it by applying the graph algorithm as given in 3.1.The reason that we need to recalculate it even when we can find it directly from the matrix D is that matrix D will have the parameter vector for this word that had been calculated and augmented after each iteration(of all words) whereas we are considering target word in it's own context to determine which sense it conveys in that context-window. The two mappings are represented as M_1W and M_2W

$$M_1W=W_1*O^T$$
$$M_2W=W_2*S^T$$

Now in order to compare semantics of the target word with that of a cluster C_i we calculate Kulback-liebler divergence between M_1C_i and M_1W and between M_2C_i and M_2W.Let us say the first is D_1 and D_2.The net divergence for cluster C_i is D_i

$D_i=a.D_1+b.D_2$

Where a and b are normalizing constants calculated experimentally and a+b=1;

Finally we assign cluster C_i to the target word W such that D_i is minimum.

Now to extract the correct dimension from this cluster we follow a two step procedure. First we consider the largest dimension of the vector C_{i1}.Let this dimension be X. Now this dimension will not represent a single sense but a mixture of senses(as we reduced dimensionality of original set and hence combining some senses in the original training set into a single dimension). So,in the second step consider the row vector in the matrix M'(reduced dimensions(k')*dependency relations) with row labeled as X. Now take dot product of this row vector with each row vector of the matrix M(original dimensions(k)*dependency relations).Since each row of M represents a sense from the original training set, the row giving the highest value of dot product is assigned as the sense to the target word.

G. WORKING EXAMPLE

Let us illustrate the process with a working example for the noun apple. The sense induction algorithm finds the following candidate senses:

1. tree, doctor, sweet, leaves, health,cancer,sweet

2. company, officials, mobile, iphone, portfolio,apps,games,macintosh,sales,profits,manufacture

3. iphone, galaxy, apps, internet, ios, mobile, games, processor, sales ,manufacturer

Each of these senses is assosciated with a centroid (average frequency vector of the cluster members) that is mapped to the semantic dimensions, i.e. yields a probability distribution over the semantic dimensions. If we consider each of three senses we can see that the 'fruit' dimension is the most dominant in the first sense. In the second sense the 'corporate' sense is most dominant. Similarly in the third sense the dimension phone is dominant.

Let us now see, a particular instance of the noun apple

Many doctors have come up with results that show excessive use of mobiles is not good for health. They say that although people playing games , using apps and internet on these phones find sweet pleasure in doing these activities they are at high risk of getting cancer. This has led to a slight decline in the sales of iphone coming as a surprise for **apple** and has further triggered a delay in the declaration of their portfolio.

Now, a context vector is created for the noun apple and is folded into the semantic space as we had done for the senses. By selecting the lowest weighted kulback leibler divergence the algorithm is able to assign the sense corporate to apple. If we take a closer look the sense fruit also could have been assigned with almost equal probability. What prevented this from happening is our dimensionality reduction due to which the dimension corporate was merged with a significant weight from the dimension mobile and acquired many dependency relations from it(which is intuitive as mobiles are produced by corporate), this helped the algorithm give more weight to the corporate sense and recognize it correctly. Hence, this form of transitive relationship between senses is introduced by using principal component analysis and hence improves results.

IV. Evaluation

A. Dataset

Our word sense induction and disambiguation model is trained and tested on the dataset of the SEMEVAL-2010 WSI/WSD task (Manandhar et al.,2010).We do this to maintain consistency of dataset as used by Van de Cruy(2011) and show improved results by introducing PCA and local co-occurrence graphs in the NMF, latent semantic indexing model given by him. The SEMEVAL-2010 WSI/WSD task is based on a dataset of 100 target words, 50 nouns and 50 verbs. For each target word, a training set is provided from which the senses of the word have to be induced without using any other resources. The training set for a target word consists of a set of target word instances in context (sentences or paragraphs).The complete training set contains 879,807 instances, viz. 716,945 noun and 162,862 verb instances.

The senses obtained during the training phase are used for disambiguation in the testing phase. In which the system is provided with a test set that consists of unseen instances of the target words. The test set contains a total of 8,915 instances , of which 5,285 nouns and 3,630 verbs. The instances in the test set are tagged with OntoNotes senses (Hovy et al.,2006). The system needs to disambiguate these instances using the senses acquired during training.

B. Implementation details

The SEMEVAL training set has been part of speech tagged and lemmatized with the Stanford Part-Of-Speech Tagger (Toutanova and Manning, 2000;Toutanova et al., 2003) and parsed with Malt-Parser (Nivre et al., 2006), trained on sections 2-21 of the Wall Street Journal section of the Penn Treebank extended with about 4000 questions from the QuestionBank6 in order to extract dependency triples. The SEMEVAL test set has only been tagged and lemmatized, as our disambiguation model does not use dependency triples as features (contrary to the induction model). We constructed different models one each for nouns and verbs. For each model, the matrices needed for our interleaved NMF factorization are extracted from the corpus. The noun model was built using 5K nouns, 80K dependency relations, and 2K context words (excluding stop words) with highest frequency in the training set, which yields matrices of 5K nouns × 80K dependency relations, 5K nouns × 2K context words, and 80K dependency relations × 2K context words. The parameter matrix is constructed by building local co-occurrence graphs and obtaining the parameter matrix 5k nouns*4 parameters. The model for verbs was constructed in a similar vein, using 3K verbs, and the same number of dependency relations and context words. For the initial k-means clustering, we set k = 500 for nouns, and k = 350 for verbs. To use the NMF model, we used 50 iterations, and factored the model to 50 dimensions. We then used the standard software mathematica to perform principal component analysis of the factored matrices and reduced dimensionality to 35.

C. Evaluation measures

The results of the systems participating in the SEMEVAL-2010 WSI/WSD task are evaluated using supervised and unsupervised techniques. The supervised evaluation in the SEMEVAL-2010 WSI/WSD task follows the sane scheme as SEMEVAL-2007 WSI task (Agirre and Soroa, 2007), with slight modifications. First step of the test set is used as a mapping corpus, to map the automatically induced clusters to gold standard senses; the second step acts as an evaluation corpus. The mapping between clusters and gold standard senses is used to tag the evaluation corpus with gold standard tags. The systems are then evaluated as in a standard WSD task, using recall. In the unsupervised evaluation, the induced senses are evaluated as clusters of instances that are compared to the sets of instances tagged with the gold standard senses (corresponding to classes).Hence, two groupings are created over the test set for a target word: a set of automatically generated clusters and a set of gold standard classes. A number of these instances will be members of both one gold standard class and one cluster. Therefore, the quality of the proposed clustering solution is evaluated by comparing the two groupings and measuring their similarity. There are two evaluation metrics in the unsupervised evaluation to estimate the quality of the clustering solutions, the V-Measure (Rosenberg and Hirschberg, 2007) and the paired F-1481 Score (Artiles et al., 2009). V-Measure measures the quality of a clustering by calculating its homogeneity (h) and its

completeness (c). Homogeneity is defined as the degree that each cluster consists of data points primarily belonging to a single gold standard class, while completeness refers to the degree that each gold standard class consists of data points assigned to a single cluster. V-Measure is the harmonic mean of h and c.

$$VM = \frac{2.h.c}{h+c} \text{..(7)}$$

In the paired F-Score (Artiles et al., 2009) evaluation, the clustering problem is treated as a classification problem (Manandhar et al., 2010). A set of instance pairs is generated from the automatically induced clusters, which consists of pairs of the instances found in each cluster. Similarly, a set of instance pairs is created from the gold standard classes, containing pairs of the instances found in each class. Precision is then defined as the number of common instance pairs between the two sets to the total number of pairs in the clustering solution (cf. formula 8).
Recall is defined as the number of common instance pairs between the two sets to the total number of pairs in the gold standard (cf. formula 9). Precision and recall combined together produce the harmonic mean (cf. formula 10)

$$P = \frac{|F(K) \cap F(S)|}{|F(K)|} \text{..(8)}$$

$$R = \frac{|F(K) \cap F(S)|}{|F(S)|} \text{..(9)}$$

$$FS = \frac{2.P.R}{P+R} \text{..(10)}$$

The obtained results are also compared to two baselines. The most frequent sense (MFS) baseline groups all testing instances of a target word into one cluster. The Random baseline randomly assigns an instance to one of the clusters.7 This baseline is executed five times and the results are averaged.
7The number of clusters in Random was chosen to be
roughly equal to the average number of senses in the gold standard

D. Results
a. Unsupervised Evaluation

In the table shown below the v-measures of some of the top ranked systems in the semeval 2010 WSI/WSD task, for nouns ,verbs as well as the complete data set separately.The fourth column shows the average number of clusters induced in the test set by the respective algorithm.V-measure of MFS baseline is 0 as by definition it's completeness is 1 and it's homogeneity is 0.

Our algorithm labeled PCNMA with overall score of 11.3 does better than the NMF_{con} and just slightly less than the NMF_{lib} ,overall it performs fairly well inducing 4.5 clusters nearly the same as NMF_{lib} and others like Duluth-WSI.

VM(%)	all	noun	verb	#cl
Hermit	16.2	16.7	15.6	10.78
UoY	15.7	20.6	8.5	11.54
KSU KDD	15.7	18.0	12.4	17.50
NMF_{lib}	11.8	13.5	9.4	4.80
PCNMA	11.34	12.4	9.5	4.5
Duluth-WSI	9.0	11.4	5.7	4.15
Random	4.4	4.2	4.6	4.00
NMF_{con}	3.9	3.9	3.9	1.58
MFS	0.0	0.0	0.0	1.00

Next we evaluate our system on the more reliable measure F-scores which penalizes systems when they produce a higher number of clusters (low recall) or a lower number of clusters (low precision) than the gold standard number of senses. We again compare our results with the scores of the best ranked systems in the SEMEVAL-2010 WSI/WSD.

FS (%)	all	noun	verb	cl
MFS	63.5	57.0	72.7	1.00
Duluth-WSI-SVD-Gap	63.3	57.0	72.4	1.02
NMF_{con}	60.2	54.6	68.4	1.58
PCNMA	55.8	50.9	61.8	4.13
NMF_{lib}	45.3	42.2	49.8	5.42
Duluth-WSI	41.1	37.1	46.7	4.15
Random	31.9	30.4	34.1	4.00

PCNMA achieves a score of 55.8 and induces a similar number of clusters as produced by other algorithms achieving similar results.This time our algorithm does better than NMF_{lib} while still inducing a significant numer of clusters which is not the case with NMF_{con} that does slightly better than our algorithm.

b. Supervised Evaluation

In the supervised evaluation the dataset is partitioned into two sets, the mapping set and the evaluation set. The automatically induced clusters are mapped to the gold standard senses
using the mapping corpus. The obtained mappings are then used to tag the evaluation set with these gold standard tags.

Table 3 shows the recall of our algorithm in the supervised evaluation in comparison with the other best performing systems in the SEMEVAL- 2010 WSI/WSD task.

Table 3: Supervised recall for SEMEVAL testset, 80% mapping, 20% evaluation

SR(%)	all	noun	verb	#S
PCNMA	63.3	59.4	71.7	1.93
NMF_{lib}	62.6	57.3	70.2	1.82
UoY	62.4	59.4	66.8	1.51
Duluth-WSI	60.5	54.7	68.9	1.66
NMF_{con}	60.3	54.5	68.8	1.21
MFS	58.7	53.2	66.6	1.00
Random	57.3	51.5	65.7	1.53

PCNMA outperforms both the NMF techniques and overall achieves the highest result in the supervised evaluation. The overall accuracy obtained is 63.3 and again the number of induced clusters come out to be nearly same as other algorithms performing similarly.

V. Conclusions and Future work

In this paper we present a model based on latent semantics and principal component analysis that is able to perform word sense induction as well word sense disambiguation. We combine the local and global approaches of word sense induction by introducing a new parameter matrix in the latent semantic induction step where we use it in non-negative matrix factorization (Van De Cruy,2008).The central focus of the paper is though the step of principal component analysis where we reduce dimensionality by merging similar senses together. This helps in combining the effect of transitve senses and providing better performance in the induction and disambiguation step. This is evident from the results where our model reaches state of the art performance as compared to other systems in the SEMEVAL word sense induction and disambiguation task .The evaluation set consists of many contexts for only a small amount of target words. In this regard, the global aspect of our approach performs well to extract the various senses scattered in the corpus and the local aspect strengthens the disambiguation step where the local context is more important. Hence, we think that the model presented in this paper provides a strong and holistic solution to the problem at hand.

Before we conclude we would like to throw some light on issues for future work.Although we use a new method for mapping to semantic space we would like to further train the model (in an unsupervised manner) to recognize wild senses that occur as a result of mere usage rather than semantics(like idioms).This can help improve accuracy in the unsupervised evaluation stage.Second, we would like to perform disambiguation in a hierarchical process where a noun is assigned first to it's broadest sense and then further senses down the hierarchy. Like a chip can be first assigned to a broad sense computer and then to the finer sense processor. It would require some modification in the PCA process and some vector techniques to recognize weather the dimensions are proper subsets of each other. We would also like to include grammatical dependencies in our disambiguation step to enrich our set of dependency relations.

References

[1] [1] Roberto Navigli. 2009. Word Sense Disambiguation: a Survey. ACM Computing Surveys, 41(2):1–69.

[2] Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach,and Sameer S. Pradhan. 2010. SemEval-2010 Task 14: Word Sense Induction & Disambiguation. In Proceedings of the fifth International Workshop on Semantic Evaluation (SemEval), ACL-10, pages 63–68, Uppsala, Sweden.

[3] Marianna Apidianaki and Tim Van de Cruys. 2011. A Quantitative Evaluation of Global Word Sense Induction.In Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), published in Springer Lecture Notes in Computer Science (LNCS), volume 6608, pages 253–264, Tokyo, Japan.

[4] WIDDOWS, D. AND DOROW, B. 2002. A graph model for unsupervised lexical acquisition.In Proceedings of the19th International Conference on Computational Linguistics (COLING, Taipei, Taiwan). 1–7.

[5] Biemann, 2006 Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems R. Navigli, G. Crisafulli. Inducing Word Senses to Improve Web Search Result Clustering.

[6] Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010), MIT Stata Center, Massachusetts, USA, pp. 116–126.

[7] (Van de Cruys, 2010)Mining for Meaning. The Extraction of Lexico-Semantic Knowledge from Text

[8] (Schütze, 1998) Dimensions of meaning. In Proc. of the 1992 ACM/IEEE Conference on Supercomputing. IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 787–796

[9] Thomas Landauer, Peter Foltz, and Darrell Laham. 1998 An Introduction to Latent Semantic Analysis. Discourse Processes, 25:295–284.

[10] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing Systems, volume 13, pages 556–562.

[11] D. Lin and P. Pantel. Discovering word senses from text. In Proc. of the 8th International Conference on Knowledge Discovery and Data Mining (KDD), Edmonton, Canada, 2002, pp. 613–619.

[12] J. Véronis. Hyperlex: Lexical cartography for information retrieval. Computer Speech and Language, 18(3), 2004, pp. 223–252

[13] Ted Pedersen. 2010. Duluth-WSI: SenseClusters Applied to the Sense Induction Task of SemEval-2. In Proceedings of the fifth International Workshop on Semantic Evaluations (SemEval-2010), pages 363–366,Uppsala, Sweden

[14] Andrew Rosenberg and Julia Hirschberg. 2007. Vmeasure:A conditional entropy-based external cluster evaluation measure. In Proceedings of the Joint 2007 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 410–420,Prague, Czech Republic.

[15] Sebastian Pad´o and Mirella Lapata. 2007. Dependencybased construction of semantic space models. Computational Linguistics, 33(2):161–199.

[16] Amruta Purandare and Ted Pedersen. 2004. Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces. In Proceedings of the Conference on Computational Natural Language Learning (CoNLL), pages 41–48, Boston, MA.

[17] Kristina Toutanova and Christopher D. Manning. 2000.Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pages 63–70.

[18] Jolliffe, I. T. (1986). Principal Component Analysis. Springer-Verlag. p. 487. doi:10.1007/b98835. ISBN 978-0-387-95442-4.

[19] Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. Psychology Review, 104:211–240.

[20] E. Agirre, D. Martinez, O. Lopez De Lacalle, A. Soroa. Two graph-based algorithms for state-of-the-art WSD. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP), Sydney, Australia, pp. 585–593

[21] J. Véronis. Hyperlex: Lexical cartography for information retrieval. Computer Speech and Language, 18(3), 2004, pp. 223–252

Authors

First Author – Vibhor Gaur , NSIT, New Delhi, vibhor1510@gmail.com
Second Author – Dr.Satbir Jain ,Astt. Professor ,NSIT sjain.satbir@gmail.com