Machine Learning Approach for Correcting Preposition Errors using SVD Features

Anuja Aravind and Anand Kumar M
Centre for Excellence in Computational Engineering and Networking
Amrita Vishwa Vidyapeetham
Coimbatore, India
anuja4mails@gmail.com, m_anandkumar@cb.amrita.edu

***Abstract*—Non-native English writers often make preposition errors in English language. The most commonly occurring preposition errors are preposition replacement, preposition missing and unwanted preposition. So, in this method, a system is developed for finding and handling the English preposition errors in preposition replacement case. The proposed method applies 2-Singular Value Decomposition (SVD^2) concept for data decomposition resulting in fast calculation and these features are given for classification using Support Vector Machines (SVM) classifier which obtains an overall accuracy above 90%. Features are retrieved using novel SVD^2 based method applied on trigrams which is having a preposition in the middle of the context. A matrix with the left and right vectors of each word in the trigram is computed for applying SVD^2 concept and these features are used for supervised classification. Preliminary results show that this novel feature extraction and dimensionality reduction method is the appropriate method for handling preposition errors.**

Keywords—Preposition error correction; Singular Value Decomposition; Support Vector Machines.

I. INTRODUCTION

Natural Language Processing (NLP) is a field of computer science, artificial intelligence, and linguistics which is concerned with the communications between machines and human languages [1]. There are many challenges in NLP. Among those challenges of NLP, this developed system tries to solve the English preposition errors mostly faced by non-native English writers. Machine learning based approach is developed for the correction of preposition errors. For feature extraction SVD^2 concept is applied and the obtained features are used in the SVM classification.

II. RELATED WORKS

Preposition disambiguation or preposition error correction/selection is a distinct computational task making it one of the problems in computational linguistics. Methods like Statistical and machine learning have been successfully applied to the preposition selection problem. Nowadays, methods that train on manually sense-tagged corpora have become the mainstream approach to preposition disambiguation, having the best results in task of the *Senseval* competitions [2]. The HOO 2012 shared task [3] for preposition errors works on error detection and correction in the use of prepositions and determiners. The system solved three error types of preposition which are replacement preposition, missing preposition and unwanted preposition. *SemEval* task to disambiguate prepositions investigates the extent to which an important closed class of words could be disambiguated [4]. The result had two baselines: the *FirstSense* baseline selects the first sense of each preposition as the answer and the second one the *FreqSense* baseline selects the most frequent sense from the training set. The greater number of senses leads to the performance degradation.

Stephen Tratz and Dirk Hovy [5] built a supervised classification system for disambiguation of preposition senses. Linguistically motivated features were derived from both sides of the preposition. For testing the developed system five different classifiers were used in that research. The classifiers which were used are Multinomial Naive Bayes, SVMs, kNN, and decision trees using the WEKA toolkit and Maximum entropy which was chosen as the primary classifier.

III. PREPOSITION ERROR CORRECTION METHODOLOGY

Prepositions act as an important vehicle for indicating semantic roles. They are often discarded in processing text as their meanings are difficult to analyze [4]. Prepositions are a closed class, which means that the number of prepositions remains relatively constant and their meanings are relatively stable. Despite this, their treatment in computational linguistics has been limited. Prepositions are viewed as function words that occur with high frequency words and therefore carry little meaning [1]. Sometimes preposition selection for a particular context may become a dilemma as to put which preposition after certain words. The selection of appropriate preposition in a particular context is a matter of idiom or convention rather than being governed by some set of rules. Prepositions pose such a challenge to learners because they appear to have no easily definable pattern which can be of any use in making choices in novel contexts [6]. Thus leads to preposition errors in the context. Here, this system is built to find the misclassifications and selecting apt prepositions in the given n-gram context.

Choosing the right parameters for preposition disambiguation task is critical to the success of the experiments. A substantial amount of work has been done in disambiguating prepositional attachment, words, and names. Prepositions are ambiguous with most of the other word types [7]. This project explores the idea for prepositions, an often overlooked word class. Disambiguating prepositions correctly helps in improving the translation quality. This preposition error correction methodology applies SVD^1 and SVD^2 features for data decomposition resulting in fast calculation and feature dimension is reduced. These features are further given to the classification algorithm.

978-1-4799-3080-7/14/$31.00

IV. SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition of a matrix say A is the factorization of A into the product of three matrices i.e., $A = USV^T$ with U and V as orthonormal matrices, where U are the left singular vectors, S is a diagonal matrix with singular values and V^T has rows that are the right singular vectors [8]. The SVD represents an expansion of the original data in a coordinate system where the covariance matrix is diagonal. Calculating the SVD consists of finding the Eigen values and eigenvectors of AA^T and A^TA. The eigenvectors of A^TA make up the columns of V and the eigenvectors of AA^T make up the columns of U. Also, the singular values in S are the square roots of Eigen values from AA^T or A^TA. The singular values are the diagonal entries of the S matrix and are arranged in descending order. The singular values are always real numbers.

In the special case in which A is just an $m \times m$ square matrix with positive determinant whose entries are plain real numbers, then U, V^T, and S are $m \times m$ matrices of real numbers as well, S can be regarded as a scaling matrix, and also U and V^T can be viewed as rotation matrices [9]. If the above mentioned conditions are met, then the expression USV^T can be intuitively interpreted as a composition of three geometrical transformations: a rotation, a scaling, and another rotation. Since U and V^T are unitary, the columns of each of them form a set of orthonormal vectors, which can be regarded as basis vectors. By the definition of a unitary matrix, the same is true for their conjugate transposes U^T and V. In short, the columns of U, U^T, V, and V^T are orthonormal bases.

A. SVD^1 and SVD^2

Given a $m \times n$ matrix A, we compute the first k dimensions of an SVD on the matrix A. $A_k = USV^T$. Then we use $A^1 = U*S$ to represent each object as a k-dimensional vector [10]. Before applying SVD, the matrix is normalized. This results in the SVD^1 representation which leads to a new matrix of reduced dimensionality and compactness without any loss of information. After applying SVD^1 we got A^1, then second SVD is preformed on the resulting A^1 matrix of dimensionality $n \times k$, (i.e., $A^1 = U^1 S^1 V^{1T}$). Then we again use $A^2 = U^1 * S^1$ to represent each object as k-dimensional vector. This is the representation of SVD^2.

In this system, the left and right context vectors are extracted for each word of the trigram of a corpus. Then SVD is used to reduce the feature vector space dimensionality from higher dimension to low dimension space. Each vector is normalized to unit length as our representations are words count where the absolute magnitude of a count contains little useful information. Then SVD^2 is preformed which rotates the representation to increase the focality [10]. SVD^2 is used to find the representation that reveals the underlying structure of the dataset. After pre-processing, a machine learning algorithm is used to learn how to classify the prepositions, i.e. creating a model for input-output mappings. Based on the training model the corresponding preposition labels are obtained. For classification purpose Support Vector Machine is selected as a flexible and well-established classifier.

V. FEATURE EXTRACTION USING SVD^2

In this method, SVD^2 output is used as features. For that first step is finding the left vectors and right vectors of each word in the preposition trigram context with respect to the vocabulary words. The prepositions considered are *at, for, on* and *with.*

A. Feature Extraction-Left and Right Context Words

In this system, the focus was put on feature reduction methods and comparing them. First, the features are extracted by extracting the left and right context vectors for each preposition in the middle of the context. The left and right context vectors are extracted by going through a corpus and counting the number of times a word occurs one word to the left (or the right, respectively) of the target preposition (i.e., the preposition for which you want to construct a vector).

So each word is represented as a vector. Each word contains left and right vector. The target word is the preposition and the number of times the preceding word of the target word and the target word occurred together gives the left context vector. Similarly we found the right context word by finding the number of occurrences of the target word and the subsequent word of the target word. Similarly, the left and right vectors of each word in the trigram context with respect to the vocabulary words are found out. Each trigram is represented as the concatenation of six vectors, the left and right context vectors for the three words.

B. Layer1 Representation

We are applying SVD to trigrams with preposition in the middle of the context. The prepositions considered are at, for, on and with. Then we get the left vector and right vector for each word in the trigrams with the vocabulary words and form a matrix. That matrix is log normalized then we find SVD of that log normalized matrix obtained.

With SVD we reduce the dimensionality of the matrix. As explained in Fig.1 we have found SVD^1 for 100, 200, 300, 400 and 500. Then again SVD^2 is applied to output of SVD^1 and label those features manually into corresponding 4 classes. These labeled features are then given for SVM classification.

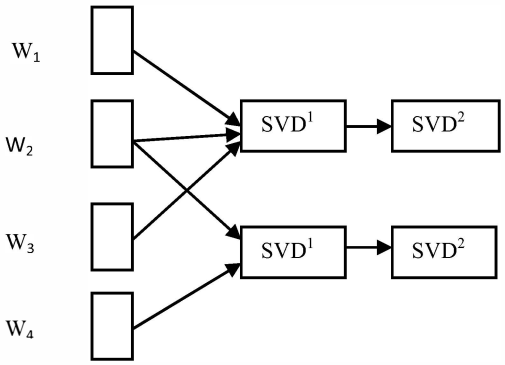

Fig. 1. Layer-1 Representation

C. Layer-2 Representation

In layer 2 the left vector and right vectors of all the vocabulary words is calculated and formed a matrix. After log normalizing the matrix SVD is applied. To reduce the dimensionality we apply SVD for different *k* values: 100, 200, 300, 400 and 500. As in layer 1 we have 4 preposition trigrams text files. From those trigrams take each word and match with the normalized matrix having left and right vectors of each vocabulary word, for those words corresponding vectors are extracted and form another matrix. Thus the features obtained will be $m \times 3k$. So again we take SVD for the obtained feature matrix with previous *k* values used initially in the first SVD. This obtained matrix will be the output of SVD^1. Then again SVD^2 of the output of SVD^1 is done which shown in Fig. 3.3. Manually label the features obtained from SVD^1 and SVD^2. And these manually labeled features are given for SVM classification.

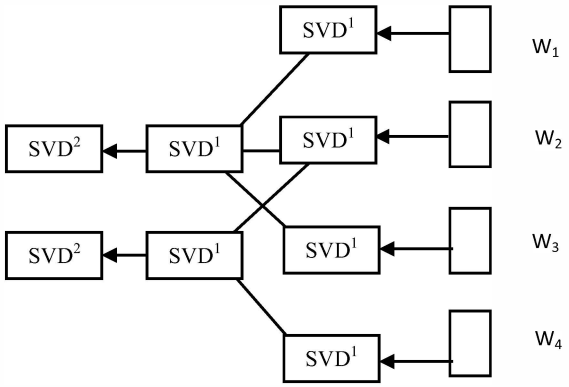

Fig. 2. Layer-2 Representation

VI. Classification using Support Vector Machine

In machine algorithms, Support Vector Machines are supervised learning models which are associated with learning algorithms that analyze data and recognize patterns, which are used for classification and regression analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can also efficiently perform a non-linear classification using kernel trick i.e. implicitly mapping their inputs into high-dimensional feature spaces. Support Vector Machines are successfully applied to Natural Language Processing problems like Part Of Speech Tagging [11], Chunking [12], Transliteration [13] and Morphological Analysis [14] [15] [16] [17].

Classifying data is the most common task in machine learning. Suppose some data points are given each belonging to one of two classes, and the goal is to decide which class a new data point will belong to. There are many hyper planes that might classify the given data. The best hyper plane is the one that represents the largest separation, or margin, between the two classes. So a hyper plane is chosen such that the distance from it to the nearest data point on each side is maximized. If this kind of hyper plane exists, it is known as the maximum-margin hyper plane and the linear classifier defined by it is known as maximum margin classifier [18].

SVM searches for a hyper plane, which separates positive and negative examples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal. Fig.3 illustrates the Maximum-margin hyper plane and margins for an SVM trained with samples.

The equation of a hyper plane is:

$$w^T x + b = 0 \qquad (1)$$

The classification of an unseen test data say **x** is based on the sign of $w^T x + b = 0$.
The separator property can be formalized as:

$$w^T x_i + b \geq 1 \text{ if } y_i = +1 \qquad (2)$$

$$w^T x_i + b \leq 1 \text{ if } y_i = -1 \qquad (3)$$

Where, $\mathbf{w}^T\mathbf{x}$ is defined as the inner product between the weight vector **w** and the input data vector **x**. SVM is used as a linear classifier by setting the class to 1 if $f(\mathbf{x}) > 0$ and setting to 0 otherwise. The main motive of SVM is to select a hyper plane that can separate the positive and negative examples while also maximizing the smallest margin. Let a set of training examples be $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is an input data vector and y_i is its corresponding class label, $y_i \in \{1, -1\}$. The problem of finding the hyper plane can be stated as the following optimization problem [19]:

$$\text{Minimize: } \tfrac{1}{2} w^T w \qquad (4)$$

$$\text{Subject to: } y_t(w^T x_i + b) \geq 1,\ i=1,2..n \qquad (5)$$

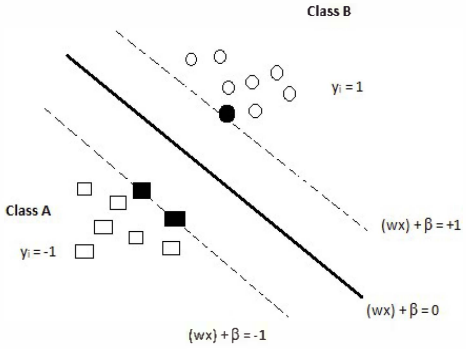

Fig. 3 Maximum-margin hyper plane and Margins[19]

The soft margin SVM is proposed, to deal with the cases where there may be no separating hyperplane due to noisy labels of both positive and negative training examples, which is formulated as:

$$\text{Minimize: } \tfrac{1}{2} w^T w + C \sum_{i=1}^{n} \varepsilon_t \qquad (6)$$

$$\text{Subject to: } y_t(w^T x_i + b) > 1 - \varepsilon_t \quad i=1,2..n \qquad (7)$$

Where, $C \geq 0$ is a parameter that controls the amount of training errors allowed. Machine learning algorithms tend to over learn when the dimensionality is high. SVM avoids this, because it does not combine features, but it linearly combines a function of the examples.

This paper is mainly focusing on 4 classes of prepositions which are *at, for, on* and *with.* Each training point belongs to one of these four different classes. The goal is to construct a model which, given a new data point, will correctly predict the class to which the new point belongs. The most appropriate approach for doing so is by reducing the single multiclass problem into multiple binary classification problems.

VII. Results and Discussions

In this method, a small hand-encoded corpus is used for experiments, which is having a vocabulary of 4491 unique words. From that corpus, trigrams are extracted with prepositions like *at, for, on* and *with* in the middle of the context. Thus 811 trigrams have been extracted. Out of 811 trigrams, 611 has been given for training and remaining 200 is been used for testing. In that trigrams, 134 is trigrams with preposition *at*, 304 with preposition *for*, 137 with *on* and 236 trigrams having preposition with.

Detailed experiments are done with various dimensions of SVD. Table.I and II gives the recognition rate for different reduced-rank dimensions which are 100, 200, 300, 400 and 500 of Layer1 SVD^1 and SVD^2 and for different kernels. The different kernels used here are Linear, Gaussian RBF, Homogeneous polynomial and Non-homogeneous polynomial kernels which are represented as K1, K2, K3 and K4 respectively in the tables given below. Similarly Table III and IV gives the recognition rate of Layer2 SVD^1 and SVD^2 for different kernels and for the different reduced-rank dimensions same as that used for Layer1. In MSVM tool the recognition rate is calculated using the formula:

$$R = \frac{value\ of\ dual\ objetive\ function}{upper\ bound\ on\ the\ optimum}$$

TABLE I. RECOGNITION RATE FOR LAYER1- SVD^1 FOR DIFFRENT DIMENSIONS AND KERNELS

SVD Dimensons	K1	K2	K3	K4
100	100%	99%	77%	86.50%
200	99%	98.50%	66.50%	73%
300	98.50%	94%	61.50	66%
400	78.50%	78.50%	55%	58%
500	74%	69.50%	52.50%	56%

TABLE II . RECOGNITION RATE FOR LAYER1-SVD^2 FOR DIFFERENT DIMENSIONS AND KERNELS

SVD Dimensons	K1	K2	K3	K4
100	100%	100%	90.50%	99%
200	99%	99%	84%	77.50%
300	99%	96.50%	79%	72%
400	95.50%	88%	60.50%	66%
500	93.50%	87%	68%	63.50%

TABLE III RECOGNITION RATE FOR LAYER2- SVD^1 FOR DIFFRENT DIMENSIONS AND KERNELS

SVD Dimensons	K1	K2	K3	K4
100	100%	99%	76.50%	88%
200	99%	97.50%	92%	89%
300	96.50%	91.50%	83%	69%
400	78.50%	79%	55.50%	59.50%
500	84.50%	81%	74.50%	77%

TABLE IV. RECOGNITION RATE FOR LAYER2- SVD^2 FOR DIFFRENT DIMENSIONS AND KERNELS

SVD Dimensons	K1	K2	K3	K4
100	100%	100%	91.50%	99.50%
200	99%	99%	94%	79.50%
300	98.50%	96%	87.50%	74.50%
400	89%	88%	57.50%	67%
500	87.50%	82.50%	76%	80%

The following tables (Table.V-XIV) show the confusion matrices for Layer1 and Layer2 both the SVDs which are got on running Kernel 2 for all the five dimensions. There are four preposition classes. Thus the confusion matrix shows how many got correclty classified to its own class and how many got missclassified to remaining classes. Out of that 200 trigram features, 50 trigram features belongs to each of the four classes making it total of 200 used for testing.

TABLE V. CONFUSION MATRIX FOR LAYER1 DIMENSION 100

	SVD^1					SVD^2			
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	50	0	0	0	**at**	50	0	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	2	48	0	**on**	0	0	50	0
with	0	0	0	50	**with**	0	0	0	50

TABLE VI. CONFUSION MATRIX FOR LAYER1 DIMENSION 200

	SVD^1					SVD^2			
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	49	1	0	0	**at**	50	0	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	2	48	0	**on**	0	2	48	0
with	0	0	0	50	**with**	0	0	0	50

TABLE VII. CONFUSION MATRIX FOR LAYER1 DIMENSION 300

	SVD^1					SVD^2			
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	44	6	0	0	**at**	48	2	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	4	46	0	**on**	0	4	46	0
with	0	2	0	48	**with**	0	1	0	49

TABLE VIII. CONFUSION MATRIX FOR LAYER1 DIMENSION 400

	SVD^1					SVD^2			
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	30	20	0	0	**at**	39	11	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	11	39	0	**on**	0	6	44	0
with	0	12	0	38	**with**	0	7	0	43

TABLE IX. CONFUSION MATRIX FOR LAYER1 DIMENSION 500

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	30	17	2	1	**at**	35	14	0	1
for	0	50	0	0	**for**	0	49	0	1
on	6	4	40	0	**on**	0	8	41	1
with	0	6	2	42	**with**	0	1	0	49

TABLE X. CONFUSION MATRIX FOR LAYER2 DIMENSION 100

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	50	0	0	0	**at**	50	0	0	0
for	0	2	48	0	**for**	0	50	0	0
on	0	0	50	0	**on**	0	0	50	0
with	0	0	0	50	**with**	0	0	0	50

TABLE XI. CONFUSION MATRIX FOR LAYER2 DIMENSION 200

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**At**	**for**	**on**	**with**
at	48	2	0	0	**at**	48	2	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	2	48	0	**on**	0	2	48	0
with	0	1	0	49	**with**	0	1	0	49

TABLE XII. CONFUSION MATRIX FOR LAYER2 DIMENSION 300

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	42	8	0	0	**at**	47	3	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	6	44	0	**on**	0	4	46	0
with	0	3	0	47	**with**	0	1	0	49

TABLE XIII. CONFUSION MATRIX FOR LAYER2 DIMENSION 400

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	30	20	0	0	**at**	38	12	0	0
for	0	50	0	0	**for**	0	50	0	0
on	0	10	40	0	**on**	0	7	43	0
with	0	12	0	38	**with**	0	5	0	45

TABLE XIV. CONFUSION MATRIX FOR LAYER1 DIMENSION 500

SVD[1]					SVD[2]				
Prepos-itions	**at**	**for**	**on**	**with**	**Prepos-itions**	**at**	**for**	**on**	**with**
at	29	19	0	0	**at**	31	18	0	1
for	0	50	0	0	**for**	0	50	0	0
on	0	10	40	0	**on**	0	8	41	1
with	0	7	0	43	**with**	0	7	0	43

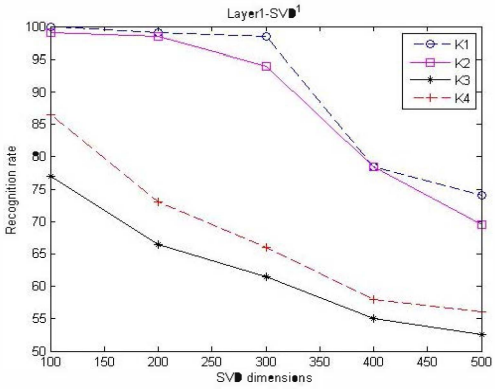

Fig. 4. Layer1-SVD[1] for different dimensions and kernels

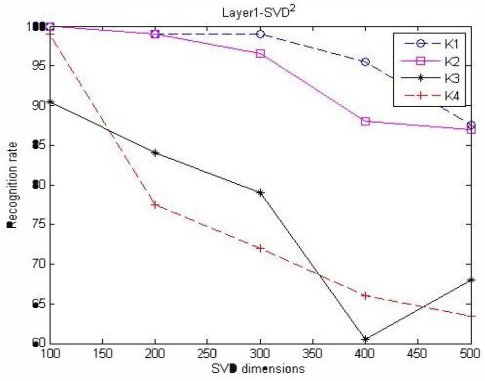

Fig. 5. Layer1-SVD[2] for different dimensions and kernels

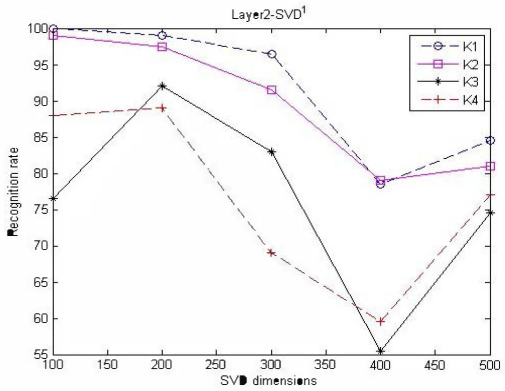

Fig. 6. Layer2-SVD[1] for different dimensions and kernels

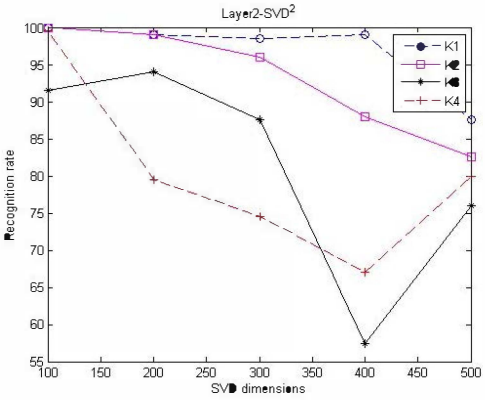

Fig. 7. Layer2-SVD^2 for different dimensions and kernels

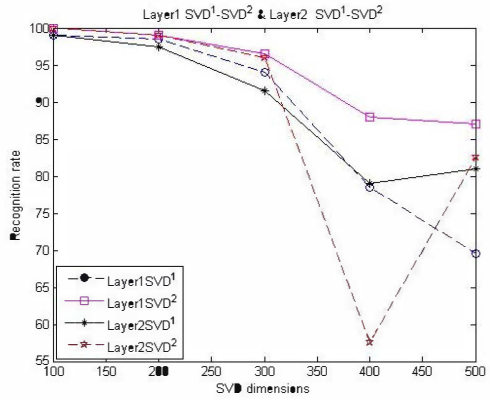

Fig. 8. Layer1 SVD^1-SVD^2 and Layer2 SVD^1-SVD^2 for kernel 2

From Fig.4 to Fig.7, are the graphs plotted with the values from Layer1 and Layer2 tabulation results drawn above. These graphs show the decreasing values for different kernels with the increasing SVD dimension feature values. Fig.8 is the graph which shows the comparision of values for different layers and different SVDs with all the five dimensions on using Kernel 2.

VIII. Conclusions

This paper presents a novel approach for correcting preposition errors using Two-SVD and machine learning classification. Context based feature descriptors are retrieved from the neighboring context and the dimensions are reduced using Singular Value Decomposition and then these descriptors used as features in SVM classifier. To our knowledge this present approach is the initial attempt in the preposition error correction problem. Here we applied the SVD^1 and SVD^2 for reducing the context distributional features. Detailed experiments are conducted with various k reduced-rank values of SVD and different Support Vector Machine kernels. Preliminary results show that this novel feature extraction and dimensionality reduction method is the appropriate method for handling preposition errors. One of the major findings in this research is SVD^1 is sufficient for small dataset compare with SVD^2. SVD dimension also plays a major role in finding the accuracy of the system. If the reduced-rank dimension k increases then the performance degrades gradually. Future possible work is to build the system with huge dataset and formulate it to handle more prepositions.

Other directions of this research are to increase the data size and n-gram window size and removal of stop words in context distributions.

References

[1] Natural Language Processing "http:/en.wikipedia.org/wiki/ Natural_language_processing".

[2] Sense Eval, http://www.senseval.org/.

[3] Robert Dale, Ilya Anisimoff and George Narroway, "HOO 2012:A Report on the Preposition and Determiner Error Correction Shared Task", Proceedings of the seventh workshop on building educational applications using nlp, 2012, pp 54-62.

[4] Ken Litkowski and Orin Hargraves, "SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions" , Proceeding SemEval '07 Proceedings of the 4th International Workshop on Semantic Evaluations, 2007, pp 24-29.

[5] Stephen Tratz and Dirk Hovy, "Disambiguation of Preposition Sense Using Linguistically Motivated Features", SRWS '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium, 2009, pp 96-100.

[6] De Felice, Rachele, and Stephen Pulman. "Automatic detection of preposition errors in learner writing." *Calico Journal* 26, no. 3, 2009, pp 512-528.

[7] Dirk Hovy, Stephen Tratz, and Eduard Hovy, "What's in a Preposition? Dimensions of Sense Disambiguation for an InterestingWord Class", COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Posters, 2010, pp 454–462

[8] Singular Value Decomposition chapter, www.cs.princeton.edu/ courses/archive/spring12/ cos598C/ svdchapter.pdf.

[9] Singular Value Decomposition, http://en.wikipedia.org/wiki/ Support_vector_machine.

[10] Hinrich Schuetze and Christian Scheible. "Two SVDs produce more focal deep learning representations" Proceedings of the 1st International Conference on Learning Representations (ICLR), 2013.

[11] Dhanalakshmi, V., Anandkumar, M., Vijaya, M. S., Loganathan, R., Soman, K. P., & Rajendran, S. Tamil Part-of-Speech tagger based on SVMTool. In Proceedings of the COLIPS International Conference on natural language processing (IALP), Chiang Mai, Thailand, 2008.

[12] Dhanalakshmi, V.; Padmavathy, P.; Anand, K.M.; Soman, K.P.; Rajendran, S., "Chunker for Tamil," Advances in Recent Technologies in Communication and Computing, ARTCom, 2009, pp.436,438.

[13] Keerthana, S., V. Dhanalakshmi, M. Anand Kumar, V. P. Ajith, and K. P. Soman. "Tamil to Hindi Machine Transliteration Using Support Vector Machines." In *Signal Processing and Information Technology*, pp. 262-264. Springer Berlin Heidelberg, 2012.

[14] Dhanalakshmi, V, Anand Kumar, M.; Rekha, R.U.; Arun Kumar, C.; Soman, K.P. and Rajendran, S., "Morphological Analyzer for Agglutinative Languages Using Machine Learning Approaches," Advances in Recent Technologies in Communication and Computing, 2009., pp.433-435.

[15] Sai Kiranmai, G., K. Mallika, M. Anand Kumar, V. Dhanalakshmi, and K. P. Soman. "Morphological Analyzer for Telugu Using Support Vector Machine." In Information and Communication Technologies, Springer Berlin Heidelberg, 2010, pp. 430-433.

[16] Abeera, V. P., S. Aparna, R. U. Rekha, M. Anand Kumar, V. Dhanalakshmi, K. P. Soman, and S. Rajendran. "Morphological analyzer for malayalam using machine learning." In Data Engineering and Management, Springer Berlin Heidelberg, 2012, pp. 252-254.

[17] Anand Kumar, M, V. Dhanalakshmi, K. P. Soman, and S. Rajendran. "A Sequence Labeling Approach to Morphological Analyzer for Tamil Language." International Journal on Computer Science & Engineering, 2010.

[18] K. P. Soman, Ajav. V, Loganathan R, "Machine Learning with SVM and other Kernel Methods", PHI Learning Pvt. Ltd, 2009.

[19] K.P. Soman, Shyam Diwakar and V. Ajay," Insight into Data Mining Theory and Practice", Prentice-Hall of India Pvt. Ltd, January 2006.