Clustering of Words using Dictionary-Learnt Word Representations

Remya R.K.Menon
Department of Computer Science and Applications,
Amrita School of Engineering
Amrita Vishwa Vidyapeetham,
Amrita University, India
ramya@am.amrita.edu,

Gargi S
Department of Computer Science and Applications,
Amrita School of Engineering
Amrita Vishwa Vidyapeetham,
Amrita University, India
gargisudha.ashok@gmail.com

Samili S
Department of Computer Science and Applications,
Amrita School of Engineering
Amrita Vishwa Vidyapeetham,
Amrita University, India
Samili15.vs@gmail.com

Abstract — **Language is the way of communication through words. This will help to get a better insight of the world. Natural Language Processing (NLP) mainly concentrate on expanding systems that allow computers to communicate with people using everyday language. One of the challenges inherent in NLP is teaching computers to recognize the way humans learn and use language. Word representations give rise to capture syntactic and semantic properties of words. So the main purpose of this work is to find out the set of words which have similar semantics by matching the context in which the words occur. In this work we explore a new method for learning word representations using sparse coding, a technique usually done on signals and images. We present an efficient sparse coding algorithm, Orthogonal Matching Pursuit to generate the sparse code. Based on the input given, sparse codes are generated for the input. The input term vectors are classified based on the sparse code by grouping the terms which have same sparse code into one class. K-Means algorithm is also used to classify the input terms vectors which have semantic similarities. Finally, this paper makes a comparison that gives the best representation from the sparse code and K-Means. The result shows an improved set of similar words using sparse code when compared to K-Means. This is because SVD is used as a part of dictionary learning which captures the latent relationship that exists between the words.**

Keywords—NLP, Sparse Coding, Dictionary Learning, K-Means, OMP, K-SVD, Terms, PMI.

I. INTRODUCTION

Representation of words (terms) means, representing a word in some language with a high dimensional vector. There are diverse techniques to influence substantial measure of unlabeled data. One particular technique is depicting individual words of a language as vectors that catch semantic properties of the words in a multidimensional space. These depictions can serve as a noteworthy building unit to various Natural Language Processing (NLP) applications. When we create word vectors from a considerable amount of information, the outcome vectors can be utilized to answer the semantic connections between words, for example, a city and the nation it belongs to. Word vectors with such semantic relationships could be used to improve many existing NLP applications used for translating text from one language to another, to retrieve particular information from an information resource and in question answering systems. So the main purpose of this work is learning word semantics based on a set of pre-defined context words (like an n-word window over the text) so as to capture the semantic relationships that exist among the words in a collection of documents. In this work, we explore new methods for learning word semantics. We propose a new method for learning word semantics using sparse coding. Here we capture the association between terms and its contexts using sparse coding. The main idea behind sparse coding is to find a set of basis vectors and input vector is represented as a linear combination of these basis vectors. We show an algorithm for sparse coding. This algorithm enables application of sparse coding to learn word semantics from a corpus of words.

The training data contains set of sentences collected from different areas such as medicine, agriculture, human-computer interaction. The sentences in this dataset are tokenized. Tokenization is the procedure of breaking up the sentences into terms, phrases, symbols or other meaningful elements called 'tokens'. Carry out stop word removal to these tokens. Stop words are the most commonly used words in a language which has less importance. All the stop words will be discarded from the tokens. After tokenization and stop word removal we will get a new list of terms. So we use this list of terms for further processing.

Term-Term matrix is a mathematical matrix which expresses the association between terms in a training dataset. In Term-Term matrix, rows correspond to context and columns correspond to terms. Terms that come in similar context have similar meanings. Row vector in a term-term matrix indicates similar word semantics.

Pointwise Mutual Information measures how much one word tells us about the other. The Pointwise Mutual Information of a pair of terms taken from the training dataset quantifies the association between the terms. It takes the count of occurrences of each term and count of co-occurrences between the terms for calculating the association.

Clustering is the task of assembling a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). There are many types of cluster models and each of these models use different algorithms to represent the clusters. Here

978-1-5090-2029-4/16/$31.00 @2016 IEEE

we use two different types of clustering techniques. One of them is clustering words by K-Means and another one is clustering words using sparse codes.

Here in this work we use the centroid based model for clustering the terms and the algorithm we used is K-Means algorithm. It represents each cluster by a single mean vector. *K-Means* clustering is a method of vector quantization. It aims to partition *n* observations (vectors in the PMI matrix) into *k* clusters in which each observation belongs to the cluster with the nearest mean. The k-means algorithm changes the centroid until adequate advancement cannot be made, i.e. the conformity in distortion since the last emphasis is short of what some edge. This yields a term book, mapping centroid to terms and the other way around. Distortion is characterized as the whole of the squared contrast between the perceptions and the comparing centroid. The input parameters to the K-Means algorithm are n-dimensional array in which each column of the array is an observation vector, the quantity of centroid to produce. The output will be the generated centroid and the terms are represented using the centroid index in which it comes under. There are different ways to calculate the 'K' value in the K-Means algorithm. It can either calculated manually from a cost function or from Eigen values

Sparse coding is a class of unsupervised techniques for learning sets of over complete bases to speak to information productively It is the process of computing the representation co-efficient based on the given input and the learnt dictionary. The sparse codes are generated for the given input using a learnt dictionary, and it is created by using dictionary learning[7]. A dictionary contains collection of word vector; input vector is represented as combination of these word vectors with the use of sparse coding technique. There exist various algorithms to perform dictionary learning. Here we adopt the K-SVD algorithm for creating a learnt discriminative dictionary. Here we use the Singular Value Decomposition method. The K-SVD algorithm is a generalization of K-Means algorithm.

In section II, we tend to check with related work done in the sector of learning and representing words, sparse representation and different algorithms used for this work. Section III represents our proposed methods to word representation and classification. Section IV represents the experimental result and its analysis and Section V shows the conclusion of the work done.

II. Related Work

This section of the paper explains the techniques and researches that are already done for the representation of words. The aim of these studies is to make better understanding about different types of methods and algorithms that are used.

[1]present a strategy for learning word representations utilizing progressive regularization as a part of sparse coding propelled by the phonetic investigation of word meanings. They show an efficient learning algorithm for hierarchical sparse coding that is suited for problems where the input matrix is large. The regularizer promotes hierarchical organization of the latent dimensions of vector-space word embeddings. Based on the standard evaluation task such as word similarity ranking, analogies, sentence completion and sentiment analysis they showed that their method is competitive with the best published representations.

[1] Mainly focus on algorithms that give method for learning and representing words. These algorithms are utilized as a part of the situation where the input matrix is expansive. Regardless, [2] presents two new model outlines for learning representations of words that attempt to minimize computational complication. The initial design is Continuous Bag-of-Words. In this model, the request of words does not hold the projection. By building commitment with a log-direct classifier they have gotten the best execution that have four future and four history words , where the planning rule is to arrange the present (center) word precisely. Assume that the current word in a sentence is w(t). The contribution to the model could be w(t−2), w(t−1),w(t+1),w(t+2), the previous and taking after expressions of the present word we are at. The outcome of the neural system will be w(t).

Continuous Skip-gram Model is the second design, rather than anticipating the present word considering the setting, it tries to help social event of a word considering another word in the same sentence. Input to the mode is w(t) and w(t−1),w(t−2),w(t+1),w(t+2) could be the output . More effectively, every present word is used as the commitment to a log-straight classifier with persevering projection layer and suspect words inside a particular degree already, then afterward the present word. Growing the degree oversees nature of the consequent word vectors, it fabricates the computational complexity. Since the more far away words are ordinarily less identified with the present word than those nearby it, they give less weight to the far off words by looking at less from those words in our course of action cases.

From [1] and [2] it is clear that paper[1] propose an alternative approach based on decomposition of a multi-dimensional matrix catching surface insights of relationship between a word and its "contexts" with sparse coding. The key novelty in their method is to govern the relationships among dimensions of the learnt word vectors, presenting various levelled association forced through an organized punishment known as the group lasso. Also the algorithm enables application of hierarchical sparse coding to learn word representations from a corpus of word tokens. [2] also take inputs from large data sets and compute the vector representation of words. Based on different types of neural networks the quality of these representations is measured in a word similarity task. This concentrated in accuracy at lower computational cos

[3] Display a model called continuous Skip-gram model which is a gainful system for adjusting amazing disseminated vector representations that catch endless semantic and syntactic word affiliations. This paper demonstrate a couple changes that make the Skip-gram model more expressive and empower it to

learn higher quality vectors more quickly. By sub sampling regular words they get immense speedup, moreover learn higher quality representations as measured by their tasks. They likewise exhibit Negative Sampling, an enhanced variety of Noise Contrastive Estimation (NCE) that adjusts more correct vectors for reliable words stood out from the various leveled softmax. Regular requirement of word representations is their impassion to word demand and their inability to speak to colloquial expressions. For instance, the implications of India" and "Air" can't be viably joined to get "Air India". Inspired by this outline, they display an essential and powerful methodology for finding expressions, and show that the vector representations can be absolutely learned by the Skip-gram model.

Like [1] and [2], [3] additionally tells about an efficient strategy for learning high quality vector representations of words from a lot of unstructured content information. Unlike most of the previously used neural network architectures for learning word vectors, preparing of the Skip-gram model does not include dense matrix multiplications. This makes the training extremely efficient.

The proposed paper uses algorithms for learning word representations using SVD based factorization. This paper aims to approximate the sparse coding problem through techniques like clustering and finds a way to get a representation for a group of words and also compares the result by clustering terms using the well known K-Means clustering algorithm.

III. METHODOLOGY

We propose a new method for learning word semantics using sparse coding, a technique usually done on signals and images. Here we capture the association between terms and its contexts using sparse coding. The implementation of this work starts with the extraction of terms from a collection of documents. From a set of documents, extract terms by tokenizing and removing stop words. Now we find out the semantic similarity between these terms. So in this work we adopted a method for clustering words using learnt word semantics

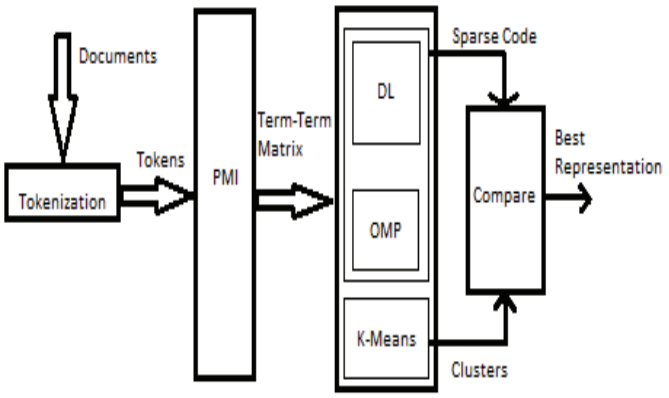

Figure 1: System Architecture

A. Pointwise Mutual Information(PMI)

In order to identify the most valid terms in a particular context, the list of terms obtained from the training dataset is considered. With these terms PMI calculation is performed. It measures how much one word tells us about the other. It is calculated using:

$$PMI(x, y) = \log (p(x, y)/p(x)\, p(y)) \qquad (1)$$

Consider two terms x and y. calculate the probability of each terms x and y, then find out the count of co- occurrence of terms x and y within the training dataset. Then calculate the PMI using these results. The output will be a Term-Term matrix. It is a mathematical matrix, which contains the vectors representing the context in which each words to be clustered or mapped are used.

Table1: PMI based Term-Term matrix

	Term1	Term2	Term3	Term4	Term5
Term1	0.0	4.8159	5.4008	2.2716	4.4008
Term2	4.8159	0.0	4.8159	4.8159	4.0789
Term3	5.4008	4.8159	0.0	5.4008	0.0
Term4	4.8159	4.8159	4.0789	0.0	5.4008
Term5	4.4008	0.0	0.0	5.4008	0.0

B. K-Means Clustering

In this work we adopt the centroid based model for clustering the terms, which is extracted from the training dataset. The algorithm we utilized is K-Means algorithm. The algorithm takes as info the quantity of clusters to produce 'k', and an arrangement of perception vectors to group. Here we take an n by n term-term matrix as the set of observation vectors to K-Means. Each column in the n by n matrix is an observation. The row is the context in which each observation occurs. ‘k’ value is calculated using the Eigen value decomposition method. Count of Eigen value is taken as the value of the ‘k’. ‘k’ is the number of clusters to be generated. After performing K-Means it gives back an arrangement of centroid, one for each of the k clusters. Each of the perception vectors from the Term-Term Matrix is ordered with the centroid list of the centroid nearest to it.

C. Sparse Coding

The aim of sparse coding is to locate an arrangement of premise vectors such that we can speak to an information vector as a direct blend of these premise vectors.

$$x = D\alpha \qquad (2)$$

Consider an input vector ‘x’. ‘x’ is a term vector which is used to represent, in which all context the term x is used with. We need to find out a sparse representation of this input vector with the help of a dictionary D, which contains n terms vectors. An input vector can be represented as a linear combination of some of the term vectors in the dictionary. The vector α contains the representation co-efficient of the term x. In order to perform sparse coding here we use the Orthogonal Matching Pursuit algorithm. OMP algorithm aims to

approximate the sparse coding problem by minimizing the total representation error, E:

$$\left\| x - D\alpha \right\|_2^2 \quad (3)$$

Here x is the term vector which we have to find the best approximation, $D \in R^{n \times v}$ is a fixed dictionary that contains v term vectors for columns $\{d_j\}_{j=1}^{v}$, the term x can be represented as a sparse linear combinations of these term vectors. The vector α contains the representation co-efficient of the term x. In approximation, l_2 norm is used to find the representation error. At each sparse coding step it decreases the total representational error, E.

1: Input: Dictionary D, term x, target sparsity K or error E

2: Output: Sparse representation, α, such that $x \approx D\alpha$

3: Init: Set I :=(), r := x, α:= 0

4: **while** (stopping criterion not met) do

5: k:= Argmax $|d_k^T r|$

6: I:= (I, k)

7: $\alpha_I := (D_I) + x$

8: r:= x - $D_I \alpha_I$

9: end while

Algorithm 1: OMP algorithm

D. *Dictionary Learning*

In this section a dictionary is learned from a given set of word vectors. Here we use dictionary learning for the sparse representation of input term vectors. The K-SVD [4] algorithm is used to train a dictionary from the given set of word vectors.

In this stage the normalized Term-Term matrix is used as the input to the OMP algorithm, which is used to generate the sparse coefficients for the given input. After performing OMP we get the sparse co-efficient and sparse index for the given input. By taking the actual Term-Term matrix, sparse co-efficient and sparse index as the input to the K-SVD, the K-SVD algorithm generate the actual learned discriminative dictionary, which is then used to find out the sparse representation for the new user input.

In this work we proposed the K-SVD algorithm for creating the learnt dictionary so as to represent the terms sparsely. Given a set of training signals, $\{x_i\}_{i=1}^{N}$, we seek the dictionary, D that leads to the best possible representations for each member in this set with strict sparsity constraints. We introduce the K-SVD algorithm that addresses the above task, generalizing the K-Means algorithm. The K-SVD is an iterative method that alternates between sparse coding dictionary and an update process for the dictionary terms so as to better fit the data. The following algorithm explains the Dictionary Learning Process.

1: Input: Normalized dictionary, D, generated coefficient matrix, α from the sparse coding phase and the weight matrix, x.

2: Output: Updated dictionary, D and the updated coefficient matrix, α.

3: For each column v = 1,2, ...,v in the Dictionary, update it by the following steps

4: Find out the group of terms which use that particular dictionary column i.e., find w_v:

$$w_v = \left\{ i : 1 \le i \le N, \vec{\alpha}_v^T(i) \ne 0 \right\}$$

5: Compute the overall representation error matrix, E_v by

$$E_v = x - \sum_{i \ne v} \vec{d}_i \, \vec{a}_j^T$$

6: From the error matrix, choose only the columns corresponding to the indices in w_v and obtain the matrix, E_R^v.

7: Perform SVD on E_R^v to get USV^T. Then the dictionary column is updated with the first column of U matrix and the corresponding coefficient vector is updated by multiplying the first row of V^T with the spectral norm of S matrix.

Algorithm2: Dictionary Learning using SVD

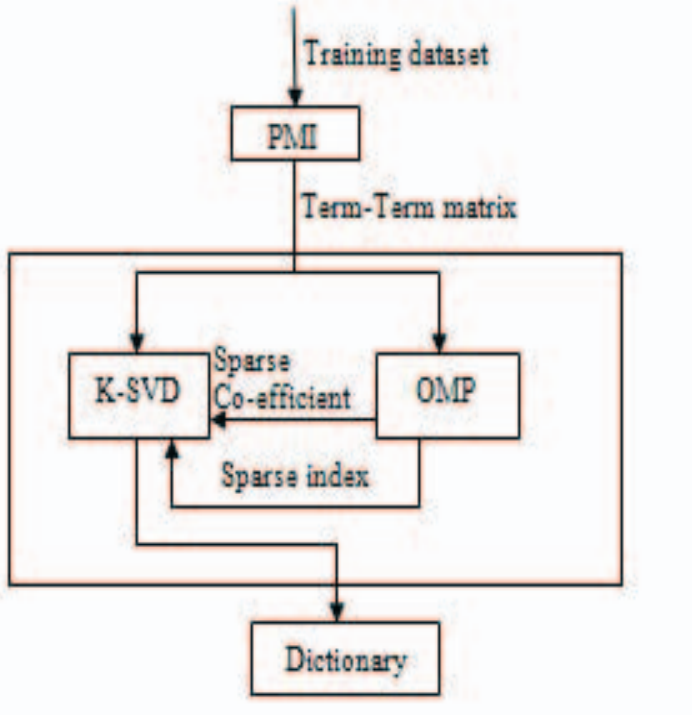

Figure2: Dictionary Learning

IV. Results and Analysis

The training dataset contains set of sentences collected from different areas such as medicine, agriculture, human-computer interaction. The following figure represents the percentage of data from each field in the training dataset.

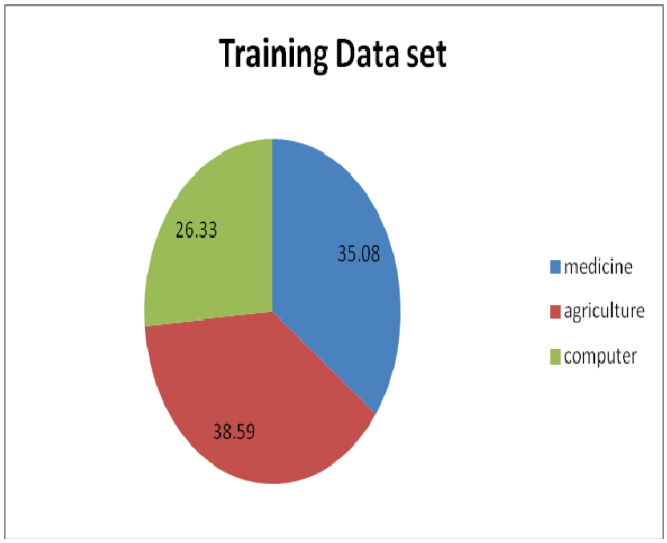

Figure 3: Training Dataset

Valid terms are extracted from the dataset. And it is represented as a 2D matrix in a python numpy array. This is a Term by Term matrix. Using this matrix we perform K-Means clustering and sparse coding technique. The table below shows the calculation of PMI for creating Term-Term matrix:

Table2: Pointwise Mutual Information

x	y	P(x)	P(y)	P(x,y)	PMI(x,y)
industrial	production	0.0118	0.0117	1	5.4008
computer	interface	0.0118	0.0118	1	5.4008
images	illusion	0.0118	0.0118	1	6.4008
health	care	0.0355	0.0650	2	2.3564
treatment	prevention	0.0177	0.0118	1	4.8159
plants	products	0.0178	0.0178	2	5.2309
global	warming	0.0177	0.0119	2	5.8159

The technique preferred in this paper for calculating 'k' value is Eigen value decomposition. Using Eigen value decomposition we calculated the Eigen values. We choose the Elbow method [5], to choose the value of 'k', the number of clusters to perform K-Means. We take the 'k' to be the number of Eigen values above the elbow curve. The following figure shows the elbow curve obtained by plotting Eigen value to the k value.

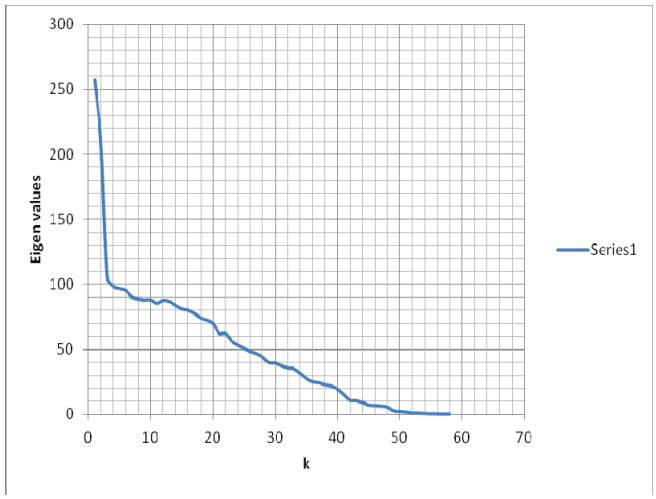

Figure 4: K value plotting

By using the 'k' value obtained from Eigen value decomposition, K-Means is performed. After performing K-Means the terms are clustered into their corresponding clusters. The resulting clusters are shown below in the figure.

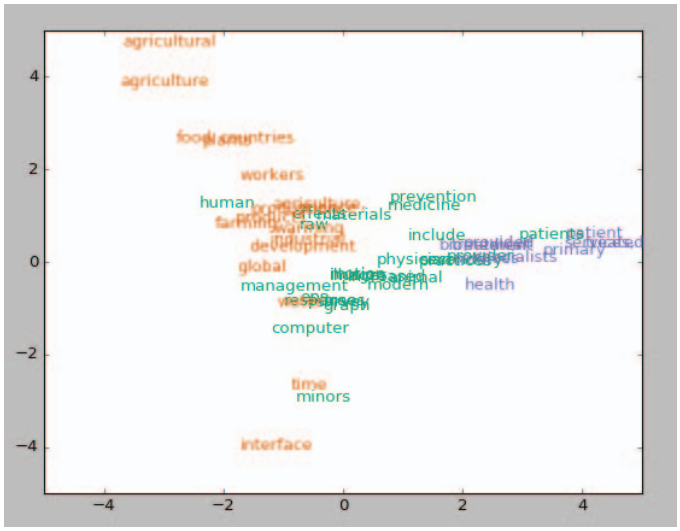

Figure 5: K-Means clustering

The method proposed in this paper is K-SVD based on which a learned discriminative dictionary is generated. Evaluation is performed on the normalized Term-Term matrix which has word vectors. Using this learnt dictionary OMP is performed on a new set of input which contains term vectors. These term vectors are randomly taken from the training dataset. For the given input sparse index and sparse coefficient is generated. Based on the sparse index generated we identify which all words have similarity. Sparse index is the index of each term in the Term-Term matrix. Grouping is done by matching the sparse index occurred for each term vector in the input. The term vectors which come under same sparse index can be grouped into one cluster. For sparse coding sparse count is also set. Sparse count indicates

the number of terms we have to choose from the dictionary to represent the input term vector. Choose *n* term vectors from the dictionary to represent the input term vector, and then check whether the input vector shows any similarity with the terms which come under the generated sparse index, and based on that, cluster the term vectors. Similarity can be checked using Cosine Similarity between the input term vector, *x* and dictionary term vectors, d_i. Cosine similarity is a measure of similarity between two vectors. The cosine of two vectors can be derived by using the Euclidean dot product formula.

$$similarity = \frac{x.d_i}{\|x\|\|d_i\|} \quad (4)$$

The table given below shows the similarity between input term vector and dictionary vectors. d_i indicates the terms in the dictionary and x_i indicates input terms

Table3: cosine similarity between Dictionary terms and input terms vectors

x_i / d_i	x_1	x_2
d_1	0.581	0.794
d_2	0.844	0.370
d_3	0.207	0.000
d_4	0.000	0.533

Input term vectors can be represented using terms in the dictionary which shows highest cosine similarity with the input vector.

We applied both K-Means and sparse coding methods on the new test set for several times and also we increased the dimension of the test set. After performing OMP sparse codes are generated for the new dataset. In order to describe the performance of these two methods, we use 'Confusion Matrix'. For each test set we found TP, TN, FP and FN values, and added these values to the corresponding cells in the confusion matrix, and also added the row and column totals. Accuracy measures how often the classifying method is correct. It is calculated from the confusion matrix using the equation given below.

$$Accuracy = \frac{TP+TN}{Total} \quad (5)$$

The figure below shows the accuracy and F-measure obtained by performing K-Means and sparse coding over changing the size of the test set. From this we found that sparse coding is better in terms of Accuracy and F-Measure for term clustering.

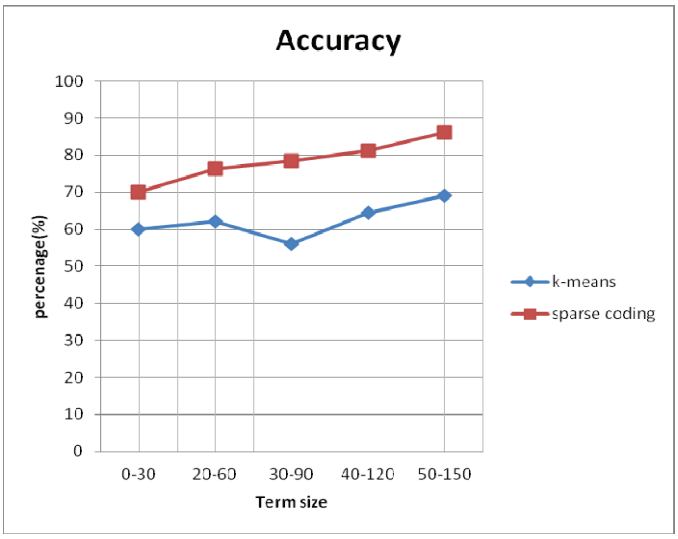

Figure6: Accuracy variation over increased test set

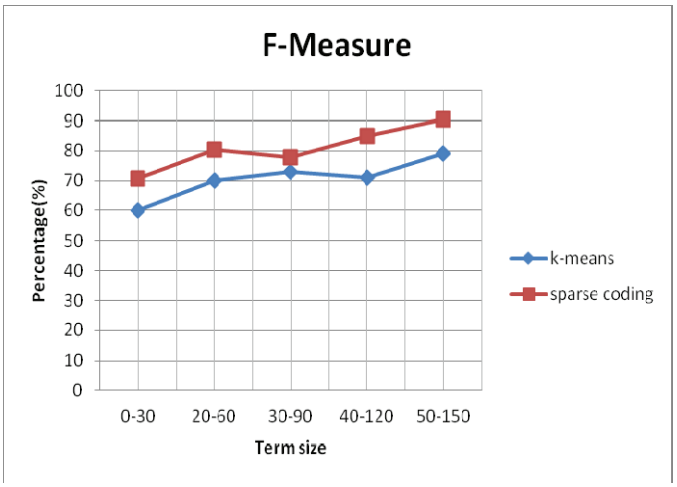

Figure 7: F-measure variation over Increased test set

V. Conclusion

The proposed work helps us to identify the relationship between the words that exist in a document taken from a specific domain. Clustering is used to group words which have similar semantics. Here we use two methods K-Means and Sparse Coding to illustrate this. We generate sparse codes by using a learnt dictionary for the given input, in order to identify, which all words in the dictionary express similar semantics with the given input. Also in K-Means, the words are clustered of by giving number of cluster to be generated. After performing these two clustering methods, we analyze the performance of these two methods using F-measure and Accuracy, and then we conclude that sparse coding is better than K-Means for clustering words having similar semantics when the term size is large.

ACKNOWLEGMENT

We would like to express our sincere gratitude to the faculty of Department of Computer Science and Applications of Amrita Vishwa Vidyapeetham, Amritapuri for providing help and guidance. Our sincere thanks to Dr. M. R. Kaimal, Chairman, Computer Science Department, Amrita Vishwa Vidyapeetham, Amritapuri for his prompt support.

REFERENCES

[1] Dani Yogatama, Manaal Faruqui, Chris Dyer, Noah A. Smith."Learning Word Representations with Hierarchical Sparse Coding". Language Technologies Institute Carnegie Mellon University Pittsburgh, PA 15213, USA {dyogatama, mfaruqui, cdyer, nasmith}@cs.cmu.edu

[2] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space.

[3] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

[4] Remya R.K.Menon, Shruthi S. Nair, K. Srindhya, and M. R. Kaimal. "Sparsity-based representation for categorical data." In *Intelligent Computational Systems (RAICS), 2013 IEEE Recent Advances in*, pp. 74-79. IEEE, 2013.

[5] Hardy, André. "An examination of procedures for determining the number of clusters in a data set." *New approaches in classification and data analysis*. Springer Berlin Heidelberg, 1994. 178-185.

[6] Lee, Honglak, et al. "Efficient sparse coding algorithms." *Advances in neural information processing systems*. 2006.

[7] Sprechmann, Pablo, and Guillermo Sapiro. "Dictionary learning and sparse coding for unsupervised clustering." *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010.

[8] Luong, Thang, Richard Socher, and Christopher D. Manning. "Better Word Representations with Recursive Neural Networks for Morphology." *CoNLL*. 2013.