CONTEXT-AWARE CLUSTERING USING GLOVE AND K-MEANS

Pulkit Juneja, Hemant Jain, Tanay Deshmukh, Siddhant Somani, B.K. Tripathy, Senior Member, IEEE

School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India

ABSTRACT

In this paper we propose a novel method to cluster categorical data while retaining their context. Typically, clustering is performed on numerical data. However it is often useful to cluster categorical data as well, especially when dealing with data in real-world contexts. Several methods exist which can cluster categorical data, but our approach is unique in that we use recent text-processing and machine learning advancements like GloVe and t- SNE to develop a a context-aware clustering approach (using pre-trained word embeddings). We encode words or categorical data into numerical, context-aware, vectors that we use to cluster the data points using common clustering algorithms like K-means.

KEYWORDS

Natural language processing, context-aware clustering, k-means, word embeddings, GloVe, t-SNE

1. INTRODUCTION

In recent years, due to the exponential growth of data and rising demand for data analytics, cluster analysis has emerged as an important field of study. It is often useful to categorize unclassified data, with applications in areas ranging from Biology and Medicine to Computer Science. Several techniques exist which can cluster data points based on their attributes [1]. One of the most popular clustering algorithms is the k-means algorithm [2] which partitions n observations into k $\leq$ n clusters by minimizing the cost function C which is defined as the sum of the intra-cluster distance of each point to its cluster center.

Despite the utility and efficiency of the k -means algorithm, it's limitation of working with only numerical data has been a barrier to its application on real-world datasets, where categorical data is more frequently encountered. Several solutions to this have been proposed. Huang [3] proposed two methods 1) the use of a simple matching dissimilarity measure such as Hamming distance and 2) The use of the most frequently occurring values of a feature i.e. the mode to computer the cluster centers. This extension to the k-means algorithm is called the k-modes algorithm. Guha et. al. [4] presented a concept of hierarchical clustering.

In contrast to the above approaches, we opted to use geometrical encoding of words derived from global contextual data. These embeddings of words do not have local context and are instead obtained from a model trained on a global corpus. We believe that this will impart a sense of generality to the clustering and higher quality results can be achieved as a result.

2. RELATED WORK

Most of the data available to the masses is unsupervised data with no predefined classes and labels, as a result Unsupervised data is much easier to obtain than supervised data but unfortunately unsupervised clustering algorithms are much less accurate and require more iterations/epochs and data to converge to an accuracy commensurate to their super- vised counterparts. Yuan *et al.* were the first one to propose the concept of context aware clustering in their 2008 paper [5]. They proposed the idea of obtaining supervision rules from unsupervised data by using the contextual dependencies of the data and using those supervision rules to improve the clustering performance. In the paper [5] they try to substantiate the concept using 2 case studies. Firstly they demonstrated the concept by clustering primitive visual features for finding local spatial patterns in images and (2) by clustering 09 handwritten digits with help of contextual patterns among different types of features. They then used this contextual information as supervision to characterize the co-occurrence statistics which were then used to resolve ambiguous data samples.

In another paper [6] A Mary Posonoa and V.L. Jyothi proposed an approach for con- textually clustering XML documents based on the similarity of the documents without any knowledge of their structural representation. Their approach used feature extraction to ex- tract prevalent features from the documents (e.g. the number of times a term appears in a document) and clustering the documents based on these features. One interesting aspect of the proposed approach is that the class labels and the context of clustering is automatically chosen based on the most dominant pattern in the documents. For example a given set of data may be clustered into market segments if that is the most prevalent pattern found by the algorithm.

The decision of the similarity measure is paramount in a clustering task. The decision greatly affects the quality of the clustering and chosen based on the application at hand. For numerical data the choice is trivial and simple distance metrics such as the Euclidean distance work really well. For categorical data, however, such distance based metrics such as Euclidean distance or Jaccard distance don't work very well because they only measure the equality between pair of values. This is a strong limitation for a clustering algorithm, since it prevents to capture contextual similarities and intricate patterns that might be hidden in the data. Dino Ienco et al. acknowledge and try to address this in their paper [7] and introduce the DILAC algorithm which uses the correlation between the categorical attributes to compute distance between the values of an attribute. The algorithm proceeds by first finding the context for a target attribute which is the set of attributes other than the target attribute that correlate most with it. Then the distance between two values of the target attribute is then calculated by applying Euclidean distance or other similar distance metric to the conditional probabilities of those values given particular values of the chosen context attributes.

3. LITERATURE REVIEW

3.1. COMPARISON OF DIFFERENT WORD EMBEDDING MODELS

The term Word Embeddings was coined By Bengio et al. (2003) and has since catapulted to become one of the most widely used language models in natural language processing and distributional semantics.

Word embeddings are method of capturing semantic analogies between the words of a corpus, which are impossible to discriminate just based on a simple number, by training a model to represent the words as a dense vector of real numbers with a lower dimension compared to the size of the vocabulary of the corpus.

Two of the most popular and efficient models that we will elaborate upon and compare in this section are word2vec [8] and GloVe [9] (which we would be using in our application).

3.1.1. Word2Vec

One of the most popular word embedding generation models that brought word embed- ding to the masses can be attributed to Mikolov et. al. (2013).

In the original paper published by Mikolov et. al. [8], they proposed two architectural models for learning word embeddings, both of these models were computationally less ex- pensive and offered two main benefits over previously proposed models:

- No expensive hidden layer
- Both of these models allow taking additional context into account which allows for better learning

At the heart of Word2Vec is a 3 layer shallow neural network that learns the distributed vectors representations for the words of the input corpus as the weight vectors of the network
Both the input and output layers have the same number of nodes as the size of the vocabulary(V), the number of nodes in the hidden layer(N) are less than V and is also the dimension of the word vectors learned.

Two vector representations are learned for each word in the corpus. One which comes from the rows of the input-hidden layer matrix known as the input vector and other one that comes from the columns of the hidden-output layer matrix known as the output vector.

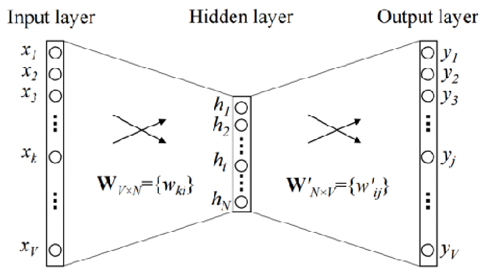


Figure 1. Basic Architecture of the word2vec Neural Model [10]

Continuous bag of words: - The continuous bag of words learning model is an unsupervised learning model that works with the objective of calculating a multinomial probability distribution predicting which word would best fit given a context. The context is represented by a collection of $\boldsymbol{N}$ words both before and after the target word.

Since there are multiple input words instead of just one, modifications to the neural network are made. The input connections to the hidden layer are duplicated $\boldsymbol{C}$ times, where $\boldsymbol{C}$ is the chosen size of the context.

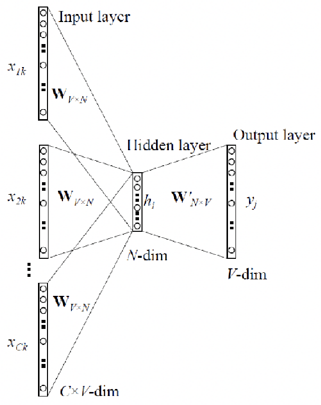


Figure 2: Continuous bag of words Model [10]

The output of the hidden layer is also modified. Instead of directly passing the input to the output layer it adds the inputs from all the connections, multiplies it with the input– >hidden layer weight matrix and averages the sum over ***C***. This vector is the final output of the hidden layer.

$$h = \frac{1}{C} W^T (x_1 + x_2 + \cdots + x_c) \quad (1)$$

Where x_i is the input vector for word i.

The training of the net (weight updation) is done by using back propagation to maximize the conditional probability of the actual output word W_o, represented by $p(W_o|W_1 \dots W_c)$.

To accomplish this a loss function is calculated which is defined as the logarithm of this Conditional probability.

$$E = -\log P(W_o\,|W_1 \dots W_c) \quad (2)$$

So the main aim of the learning algorithm is to minimize this loss function.

Skip Gram model: - The Skip Gram learning model's objective function is the exact opposite of the CBOW model. The model learns by predicting the context words for a given target word. Hence the target word is fed at the input layer and the output layer is replicated ***C*** times.

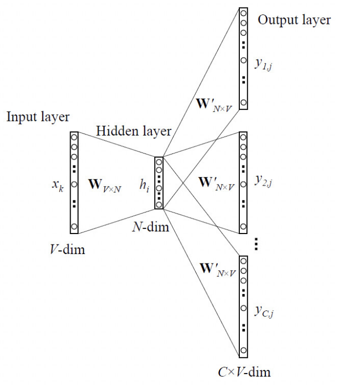


Figure 3. Skip gram Model [10]

Since there is only a single input layer in case of skip gram the output of the hidden layer remains the same that is it has a linear activation function and just copies its input as the output.

The output layer is more complicated in this model. Instead of outputting a single multinomial probability distribution, the output layer now outputs ***C*** multinomial probability distributions. The output is computed as follows

$$y_{c,j} = P(w_{c,j} = w_{o,c}|w_i) \tag{3}$$

Where $W_{c,j}$ is the j^{th} word on the c^{th} panel of the output layer, $w_{o,c}$ is the actual output word of the c^{th} panel, $y_{c,j}$ is the output of the j^{th} unit on the c^{th} panel of the output layer [10], the loss function is calculated similarly as

$$E = -\log P\left(W_{o,1}, W_{0,2}, \dots, W_{o,c}|W_I\right) \tag{4}$$

Where $P\left(W_{o,1}, W_{o,2}, \dots, W_{o,c}|W_I\right)$ represents the conditional probability of getting the actual output context given the only target word W_I. The training objective of the algorithm is to minimize this loss function.

3.1.2. GloVe

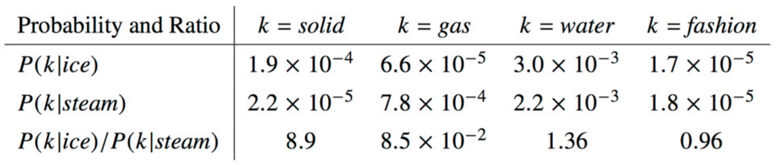
The Global Vectors for Word Representation (GloVe) model [9] is a word-vector representation model that utilizes global statistical information regarding word co-occurrences to obtain geometrical encoding of words. Pennington et al. argue that shallow window-based methods such as Word2Vec do not fully exploit the co-occurrence information present in the corpus. They claim that their method “produces a vector space with meaningful sub- structure, as evidenced by its performance of 75% on a recent word analogy task. It also outperforms related models on similarity tasks and named entity recognition.”

Background:- Matrix factorization methods, such as LSA [11], create a term-document matrix which is a sparse matrix mapping terms to their occurrences in documents. Since frequent terms such as the, a will have a disproportionately large count without contributing much to the semantic analysis, a normalization technique e.g. tf-idf needs to be used to weigh the occurrences in proportion to the number of times the terms appear in each document. The term-document matrix is then decomposed to find a low-rank approximation. By contrast, the Hyperspace Analogue to Language (HAL) [12] method creates a term-term matrix in which words are mapped to the number of times they occur within the context of another word.

Shallow window-based methods, such as the Word2Vec model described previously, learn word representations by moving a context window across the corpus. A disadvantage of these methods over Matrix factorization methods is that they only consider the local context and do not take into account the large amount of repetition in the data.

The GloVe Model: - The GloVe model considers the statistics of word occurrences as the primary source of information. The authors illustrate how GloVe utilizes with a simple example. Consider a word-word co-occurrence matrix denoted by X. Let X_{ij} denote the number of times word j occurs in the context of word i. Let $X_i = \Sigma_k X_{ik}$ denote the number of times any word appears in the context of i. Finally, let $P_{ij} = p(j|i) = X_{ij}/X_i$. Now consider the words $i = ice$ and $j = steam$. The relationship between these words can be determined by observing their co-occurrence probabilities with respect to various other “probe” words k. For a word closely related to ice but not to steam, such as solid, the ratio of $P(solid|ice)/P(solid|steam)$ will be large.

Similarly, a word like gas will have a smaller ratio of $P(gas|ice)/P(gas|steam)$. Neutral words like water or fashion will have a ratio close to one. Indeed, we can see this relationship in the following figure.



Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 4. Co-occurrence probabilities for target words ice and steam [9]

Pennington *et al.* propose that on the basis of the above argument, the starting point for the word vector learning model should be the ratios of the co-occurrence probabilities rather than the probabilities themselves. They propose a general model of the form,

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ij}}{P_{jk}} \tag{5}$$

Where w $\in$ $\mathbf{R}^{\mathbf{d}}$ are word vectors and w $\in$ $\mathbf{R}^{\mathbf{d}}$ are context word vectors. In this equation, the right hand side comes from the corpus, and F is function which can encode information present in the ratio P_{ik}/P_{jk} in the vector space. This is done by taking the vector difference of wi and wj. Since the arguments are vectors while the right hand side is scalar, the dot product of $(w_j - w_i)^T$ with $\tilde{w}_k$ is taken. Eventually, after solving for $F = exp$ and adding a bias term b_k to restore symmetry, a simplified version of Eq. 5 is obtained.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \tag{6}$$

This model has a major disadvantage in that it weighs all occurrences equally, even the rare ones. These are noisy and not significant, yet account for "75–95% of the data in X, depending on the vocabulary size and corpus." To resolve this Pennington et al. proposed a new weighted least squares regression model, introducing a weighing function $f(X_{ij})$ into Eq. 6.

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij})\,) \tag{7}$$

Where V is the size of the vocabulary. The authors found the following class of functions to work well.

The performance of the model depends weakly on the cutoff x_{max} which is taken to be 100. It was found that taking α = 3/4 gave a small improvement over α = 1. The authors note that a similar fractional power scaling was found to give the best performance in Word2Vec.

Performance comparison between GloVe and Word2Vec

Pennington et al. performed their evaluation of the GloVe model in relation on other models based on the following criteria. The type of natural language processing tasks per- formed were 1) Word analogies 2) Word similarity and 3) Named entity recognition. Five corpora of various sizes were used: a 2010 Wikipedia dump with 1 billion tokens, a 2014 Wikipedia dump with 1.6 billion tokens, the Gigaword5 which has 4.3 billion tokens, a combination of Gigaword5 +

Wikipedia2014 which has 6 billion tokens, and on 42 billion tokens of web data, from Common Crawl. For all experiments x_{max} was set to 100 and $\alpha = 3/4$. The model was trained using AdaGrad [13], an adaptive gradient descent algorithm with a faster convergence than SGD, with an initial learning rate of 0.05 for 50–100 iterations. The size of the context window was taken to be ten words from the left and right. Using the word2vec tool, the skip-gram (SG) and the continuous bag-of-words (CBOW) models were trained on the same corpus. Negative sampling was used instead of the hierarchical softmax function for training the models since it was found to have better results.

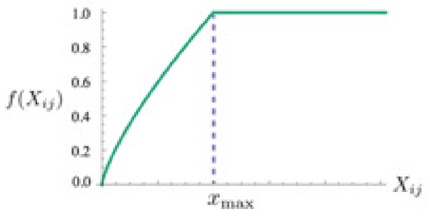


Figure 5. Weighing function with α = 3/4 [9]

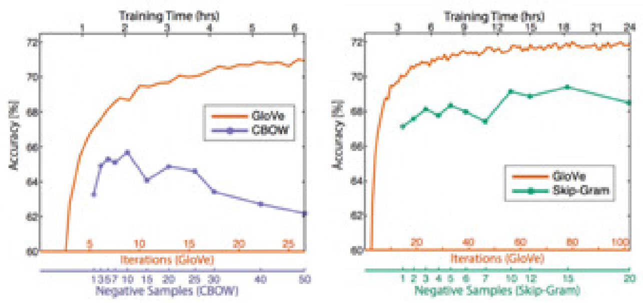


Figure 5. GloVe comparision with Word2Vec [9]

In most tasks, the GloVe model performed significantly better than other models, even with smaller vector sizes and corpora. Although a thorough quantitative comparison of GloVe with word2vec is difficult due to the existence of many parameters, the authors control for the major sources of variation viz. setting the context window size, vector length, corpus and vocabulary size to a specific configuration. Another important variable to control for is the training time. For GloVe the training time is a function of the number of training iterations. However, word2vec is currently designed for only one training epoch and modifying the number of iterations is a non-trivial task. Therefore the authors instead decided to increase the number of negative samples as that has similar results to increasing the number of epochs.

As we can see in Fig.6 GloVe consistently outperforms the word2vec model. Whereas the word2vec model starts showing a loss in accuracy after 10 additional negative samples, no such performance loss is detected in the GloVe model. Regardless of speed, it achieves dramatically better results. On the basis of these results, we have opted to use the GloVe model in our method.

3.2 Dimensionality Reduction

Dimensionality reduction is a big problem because many types of data have a large number of attributes (>30) and it is required to visualize high-dimensional data like word embeddings. This problem was recognized and while many algorithms have been built to tackle it, most had the drawback of large data loss while reducing to 2 dimensions. Algorithms like Chernoff faces [14] represented data in 3 or more dimensions and made it difficult for the researcher to understand the quality of the embedding. More importantly, the aim of dimensionality reduction was to minimize data loss while converting to lower dimensions. A significant algorithm - Principal Component Analysis (PCA) [15] that proved useful in many areas for dimensionality reduction, failed to apply well on word embeddings. It was accompanied with high data loss because the approach could not keep the lower dimension points of two similar n-dimensional data points close to each other. Hinton attributed this to a limitation of using linear mapping (PCA used linear mapping).

3.2.1. T-SNE

Laurens van der Maaten and Geoffrey Hinton (2008) proposed a solution to the problem of efficient dimensionality reduction called t-SNE (t-Distributed Stochastic Neighbor Em- bedding) which was a variation of another algorithm by Hinton and Roweis (2002) called Stochastic Neighbor Embedding. The newer approach was significantly easier to optimize and was claimed to produce better dimensionality reduction for e.g. to visualize word embeddings in 2D space.

Traditional SNE converts higher dimensional distances between two data points to conditional probabilities which express similarities between the pair of words. For two data points x_i and x_j, the conditional probability $p_{j|i}$ that x_i would select x_j as its neighbor based on their Gaussian probability distribution fixing x_i at the center is given by:

$$p_{j|i} = \frac{\exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{k\neq i} \exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)} \tag{6}$$

Where σ_i represents the variance of the Gaussian centered at the data point x_i. For the lower dimensional counterparts y_i and y_j of the higher dimensional data-points x_i and x_j respectively, we must similarly compute a conditional probability expressing the similarity between them. This conditional probability $q_{j|i}$ is calculated as:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k\neq i} \exp(-||y_i - y_k||^2)} \tag{10}$$

It is important to note that set $p_{i|i}$ and $q_{i|i} = 0$ since we are only interested in modeling pairwise similarities. For all other conditional probabilities, the aim is to find the low- dimensional representation that minimizes the difference between $p_{j|i}$ and $q_{j|i}$. The overall performance of the dimensionality reduction is expressed in terms of the Kullback-Leibler divergence. This is the cost of reducing dimensionality that signifies the loss of information while reducing from a higher to lower dimension model. This cost function C is calculated as:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{11}$$

The cost is minimized by minimizing the Kullback-Leibler divergences of all data points using a gradient descent method. In Eqn. 11, P_i denotes the conditional probability distribution over all other high-dimension data points given the data point x_i and similarly Q_i denotes the conditional probability distribution over all other low-dimension data points given y_i. It is important to note that the Kullback-Leibler divergence is not symmetric in nature. Thus, the SNE function retains the local structure while reducing the dimensionality of data. The value of σ_i too must change as it navigates through the data points since in dense regions, a smaller value of σ_i is more appropriate than in sparse regions. The importance of σ_i is that it induces a probability distribution P_i. There exists a parameter called perplexity which influences the value of σ_i and P_i selected by the SNE (using binary search) for each data point x_i. The perplexity is calculated as:

$$Perp(P_i) = 2^{H(P_i)} \tag{12}$$

Where H(Pi) is the Shannon entropy of Pi measured as:

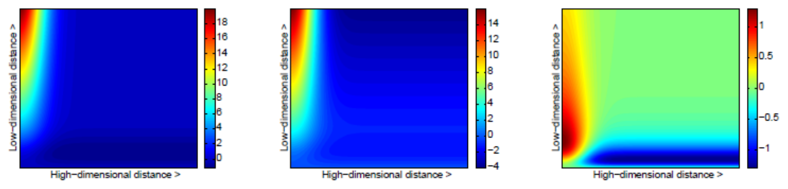
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{13}$$

The minimization of the cost function is done by means of simple gradient descent computed as:

$$\frac{\partial C}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{j|i} - q_{j|i})(y_i - y_j) \tag{14}$$

Which may be interpreted as an imaginary resultant force created by the collection of springs between y_i and all other low dimension points.

A very significant problem faced during reduction of dimensionality is the crowding problem. This happens because the pairwise distance in a 2D plane cannot effectively represent the distances in an n-dimensional hyperplane. A situation can arise where there exists n+1 data points in an n dimensional plane such that each point is mutually equidistant. In this situation we cannot express the data in a lower dimension without loss of data. The crowding problem states that the area of the 2D map available to plot distant data points will not be large enough compared to the area available to plot nearby data points. This explains that is we wish to model small distances effectively on the 2D plane, the moderate distances would require to be plotted too far away in the 2D plane. This means that for SNE, the 'spring' connecting an arbitrary data point i to the distant points in an n-dimension hyperplane, will exert an extremely small attractive force. t-SNE is an approach that intends to solve this problem. For this, t-SNE replaces the Gaussian function with a Student-t distribution to measure the similarity between two data points in the lower-dimensional space. Heavy-tailed distribution in a low-dimensional space removes both the crowding problem and optimization problems of traditional SNE. A comparison between the gradients of SNE, UNI-SNE and t-SNE is shown in Fig. 6. As we can see, t-SNE clearly outperforms its predecessors.



(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

Figure6. Gradients of SNE, UNI-SNE and t-SNE as a function of the Euclidean distances between two points in the high-dimensional and low-dimensional representations

In the high dimensional space, distances can be converted into probabilities using Gaussian distribution. In the Low dimensional space, heavier tails can be used than Gaussian tails to represent the distances as probabilities. A Student-t distribution is used in t-SNE as the heavy tailed distribution (instead of Gaussian) and the new joint probability are:

$$q_{j|i} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq i} (1+||y_i - y_k||^2)^{-1}} \tag{15}$$

Where q_{ji} is the low-dimension embedding similarity between y_i and y_j, two far apart data points, with $q_{ji} = 0$ for all i. The high-dimension embedding similarity p_{ji}, of an N data point model is calculated symmetrically by averaging over the condition probabilities as:

$$p_{ji} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{16}$$

The Kullback-Leibler divergence form for t-SNE is calculated using embedding similarities of an n-dimensional embedding ε as:

$$C(\varepsilon) = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{17}$$

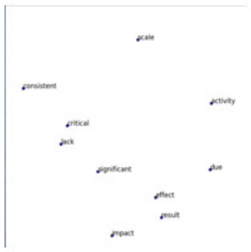
Thus, due to asymmetry of the Kullback-Leibler divergence, the objective function for an embedding ε is non-convex and minimized by a gradient descent approach as:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{i \neq j} (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j) \tag{18}$$

Where the normalization term

$$Z = \sum_{k \neq 1} (1 + \|y_k - y_l\|^2)^{-1} \tag{19}$$

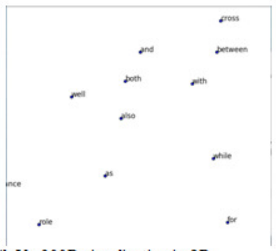
As we can see from the above equations for t-SNE, the model scales quadratically and has time complexity of O(N^2) since the normalization term sums over N(N −1) pairs. After a few thousand input objects, the learning becomes very slow and impractical on traditional systems. Laurens van der Maaten (2014) made contributions to t-SNE by accelerating learning to a time complexity of $O(N\ logN)$ by using the Barnes-Hut and the dual-tree algorithm, these algorithms limit the low dimensional embeddings to only two or three dimensions. This is because generalization to higher dimension causes exponential growth of the size of the tree is thus counterproductive for acceleration of t-SNE. Since two or three dimensional representations may suffer a high degree of information loss as compared to higher dimensional representations, we do not wish to restrict ourselves to merely the two and three dimensional representations. Therefore, for this work, we have limited ourselves to traditional t-SNE and only the 2000 most frequent words in the English language (due to CPU memory restraints).



(a) GloVe 50D visualization in 2D

(b) GloVe 100D visualization in 2D

(c) GloVe 200D visualization in 2D



(d) GloVe 300D visualization in 2D

Figure 8: Glove visualization in 2D (zoomed in)

This work discusses and compares the performance of t-SNE's dimensionality reduction on 50, 100, 200 and 300 dimensional word vectors that were taken from Stanford's GloVe (Global Vectors for Word Representation) project. Below are the zoomed in two-dimensional visualizations of a few words using the 50 (Fig. 8a), 100 (Fig. 8b), 200 100 (Fig. 8c) and 300 (Fig. 8d) dimension models for the 6B token corpus collected from Wikipedia 2014 and Gigaword5. For this model, the word vocabulary size was 400,000 and a symmetric context window of size 10 was used. As we can see, it is difficult to tell the quality of the word embeddings manually by observing the distance between a collection of pairs of words. To replace this subjective measure we get from the manual human observation of the word embeddings, we use commonly used measures for vector-vector distance calculation and similarity.

Next, the data loss is compared with the reduction ratio and find an optimal balance between the two.

3.3. Measure Of Text Similarity

Text Similarity measures play a deciding role in text relevant research and applications in tasks such as classification and clustering of texts and documents, information retrieval, topic detection and tracking, questions generation, essay scoring, question answering, text summarizing, machine translation and others. The primary stage for sentence, paragraph and document similarities is set by finding text similarity whose fundamental part is realising similarity between words.

Words can be similar in two ways -- **lexically** and **semantically** Words are similar lexically if their character sequences are similar. Words are semantically similar if they have the same thing, are opposite of each other, used in the same context, used in the same way and one is a type of another. Lexical similarity is introduced through various String-Based algorithms, Semantic similarity is established through Knowledge-Based and Corpus-Based algorithms. String-Based measures operate on string sequences and character composition. A string metric is a metric that measures similarity or dissimilarity (distance) between two text strings for approximate string comparison or matching. Corpus-Based similarity measures the semantic similarity between words according to information extracted from large corpora. Knowledge-Based similarity is a semantic similarity measure that determines the degree of similarity between words using information derived from semantic networks.

Some of the measures that are widely used in determining the word similarities are:

- **Cosine Similarity:** Cosine similarity is a measure of similarity between two vectors, t_1 and t_2, of an inner product space that measures the cosine of the angle between them.

$$t_1 . t_2 = \|t_1\| \, \|t_2\| \, \cos\theta \tag{20}$$

Given two vectors of attributes, T_1 and T_2, the cosine similarity S, $\cos\theta$, is represented using a dot product and magnitude as:

$$S = \cos\theta = \frac{T_1 . T_2}{\|T_1\|\|T_2\|} = \frac{\sum_{i=1}^{n} T_{1i} T_{2i}}{\sqrt{\sum_{i=1}^{n} T_{1i}^2}\sqrt{\sum_{i=1}^{n} T_{2i}^2}} \tag{21}$$

Where T_1 and T_2 are the components of vectors T_1 and T_2 respectively

- **Euclidian Distance:** Euclidian distance is a standard metric for geometrical problems. It is the ordinary distance between two points and can be easily measured with a ruler in two- or three-dimensional space. Euclidean distance is widely used in clustering problems, including clustering text. It satisfies all the above four conditions and therefore is a true metric. It is also the default distance measure used with the K-means algorithm. Euclidean distance or L2 distance is the square root of the sum of squared differences between corresponding elements of the two vectors. If $a = (a1, a2, \ldots., an)$ and $b = (b1, b2, \ldots, bn)$ are two points in Euclidean n-space, then the distance *d* from *a* to *b*, or from *b* to *a* is given by:

$$d(a,b) = d(b,a) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \ \ldots. \ (b_n - a_n)^2} \tag{22}$$

- **Jaccard Similarity**: Jaccard similarity is computed as the number of shared terms over the number of all unique terms in both strings.[16]

$$Similarity_{jaccard}(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{n} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)} \tag{23}$$

The numerator of the Jaccard function uses the *min* function, essentially computing the (weighted) number of overlapping features (since if either vector has a zero association value for an attribute, the result will be zero). The denominator can be viewed as a normalizing factor.

4. METHODOLOGY

In this section we will elaborate on the methodology used to cluster our sample datasets. We will attempt to explain here how the word embeddings from GloVe were extracted and how we employed T-SNE to reduce the dimensionality of the word embeddings.

4.1 EXTRACTION OF RELEVANT WORD EMBEDDINGS FROM THE GloVe DATASET

To improve the efficiency of the program at run time we found it optimal to segregate the word embeddings for the words present in the data set into a separate file. Doing so would significantly reduce the time taken during run time to search for the word embeddings. For those words for which there were no word embeddings available, we ignored the corresponding tuples. The algorithm proceeds by first parsing and loading the GloVe data set into the memory as separate arrays of labels and embeddings. It then loads the Zoo dataset pickle file and extracts the animal name, the only categorical attribute in the dataset, into a separate array .The algorithm then compares the animal names from the data set with every label in the GloVe label array and the corresponding embeddings are separated out into another array. If the animal name is not found in the GloVe dataset then is not added to the array and the tuple is discarded.

4.2. REDUCING THE DIMENSIONALITY OF THE WORD EMBEDDINGS

Following the extraction of the word embeddings for the words present in our dataset, we reduced the dimensionality using T-SNE. The particular implementation used was scikit-learn's T-SNE python implementation. The number of components preserved was varied between 2 to 11 dimensions. The pairwise distance was compared in each iteration and the mean sigma calculated as explained in section 3.2.1. Reducing the dimensionality was an important step since it allowed us to more easily process and visualize the data. It also helps us avoid the so-called "curse of dimensionality" [17] where the higher dimensional data would not have been relevant to our clustering task and would have instead interfered with it.

4.3. CLUSTERING WORDS BASED ON THE OBTAINED EMBEDDINGS

The final step involves building a k-means model based on the word embeddings obtained from the above step. The process was fairly straightforward as it involved applying the standard *k*-means algorithm on the data set. We used the scikit-learn's *k*-means implementation. An important point to determine, however, was the number of clusters to be selected. In our first sample data set, the choice was obvious. For our second sample dataset, on the other hand, certain techniques had to be used to determine the number of clusters. This will be explained further in the next section.

5. RESULTS AND DISCUSSION

For testing our algorithm, we used two datasets -- the Zoo dataset and the Flag dataset. The datasets were chosen based on the number of categorical attributes they had since our method was designed for primarily categorical datasets.

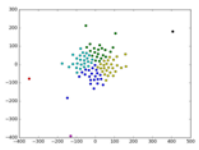
5.1 ZOO DATASET

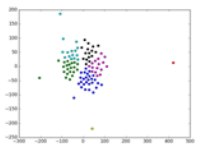
The Zoo dataset, created by Richard S. Forsyth, was obtained from the UCI machine learning repository. The dataset contains one categorical attribute (the name of the animal), two numerical attributes, and fifteen Boolean-valued attributes. It contains 101 such instances, and the animals are classified into 7 sets. This gives us an easy way to determine the number of clusters and made it easy to compare the results of the clustering.

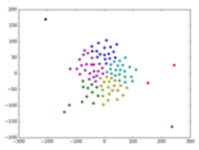
As we can see in Fig. 9, the data points are farther away at 12 dimensions, since the data loss is less than at 2 dimensions.

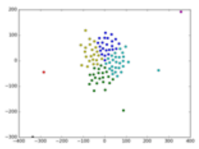
Table 1: Error while reducing dimensions of word vectors of Zoo dataset using t-SNE

No. of dimensions	No. of Iterations	Error rate (%)
2	150	1.779436
3	125	1.776164
4	350	1.200057
5	375	1.482583
6	125	1.563208
7	350	1.495372
8	125	1.547472
9	125	1.737374
10	125	1.646573


(a) Zoo clustering with 2 dimensions


(b) Zoo clustering with 4 dimensions


(c) Zoo clustering with 6 dimensions


(d) Zoo clustering with 8 dimensions

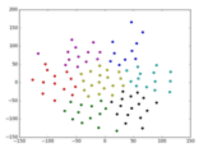
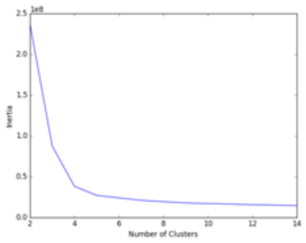

(e) Zoo clustering with 10 dimensions

Figure 9: Zoo Dataset Clustering

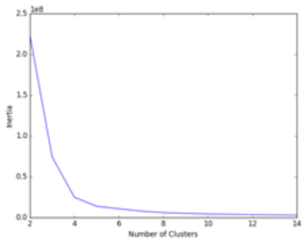
5.2. Flag Dataset

The Flag dataset, also obtained from the UCI repository, contained details of various nations and their flags. It had twenty categorical attributes and ten numerical ones. Of particular interest to us were the country's name, religion, language, the main hue of the flag, the top-left and the bottom-right hue of the flag.

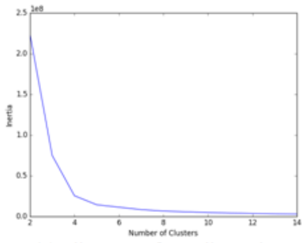
To help us determine the number of clusters, we used the elbow method [18] to obtain a rough estimate. This involved plotting the inertia (the within-cluster sum of squares distance) measure to the number of clusters. At the point where the variance drops sharply with an increase in the number of clusters, we used it to set the value of *k*.



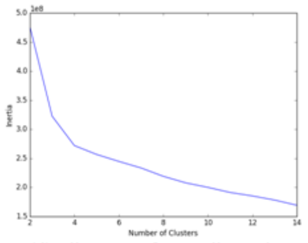
(a) Elbow test for 2 dimensions



(b) Elbow test for 3 dimensions



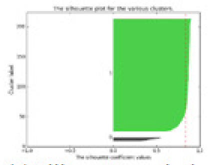
(c) Elbow test for 4 dimensions

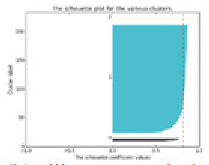


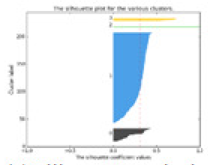
(d) Elbow test for 10 dimensions

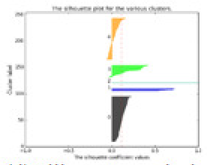
Figure 10: Elbow test in the Flag Dataset

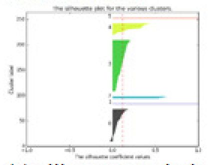
The elbow tests suggested a *k* value of 4. To test this hypothesis further we used silhouette analysis [19] to plot the silhouette scores against the number of clusters. Values near +1 indicate that the sample is far away from the neighbouring clusters. Negative values indicate that the samples may have been assigned to the wrong cluster. Ideally, the size and thickness of the plots should be similar for each cluster and the number of samples with below average scores (denoted by a red-dotted line) should be low.

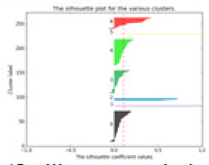

(a) Silhouette analysis with 2 clusters

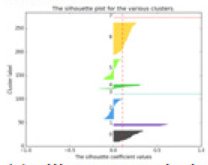

(b) Silhouette analysis with 3 clusters


(c) Silhouette analysis with 4 clusters


(d) Silhouette analysis with 5 clusters


(e) Silhouette analysis with 6 clusters


(f) Silhouette analysis with 7 clusters


(g) Silhouette analysis with 8 clusters

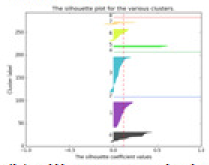
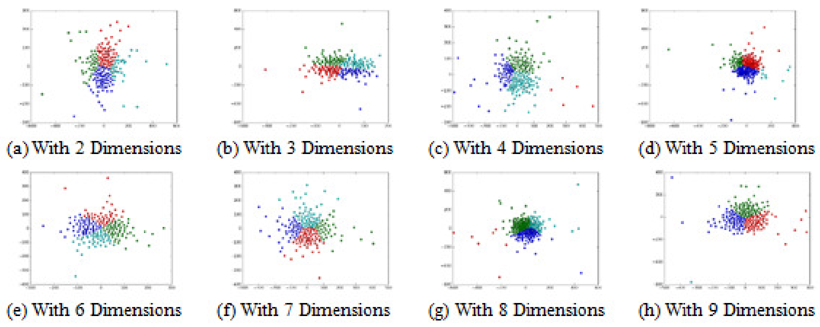

(h) Silhouette analysis with 9 clusters

Figure 11: Silhouette analysis for Flag dataset

In Fig.11 plots (a) and (b) have uneven sizes and few samples that have an above-average silhouette score. Plots (d) through (h) have samples with negative silhouette scores. Only plot (c) does not suffer from these problems. This suggests the setting of *k* to 4 for the flag dataset. We continued the clustering using this number and the results are shown below in the following plot. Fig. 12 shows the plot of the clustering results of the flag dataset when reduced to 2 to 9 dimensions.


(a) With 2 Dimensions (b) With 3 Dimensions (c) With 4 Dimensions (d) With 5 Dimensions

(e) With 6 Dimensions (f) With 7 Dimensions (g) With 8 Dimensions (h) With 9 Dimensions

Figure 12: Flag Dataset Clustering

The dataset was again reduced to two dimensions in order to show this plot. Samples belonging to different clusters were assigned different colors.

6. CONCLUSIONS

Context aware computing helps in creating devices, systems and applications that are easy to use by understanding the context of use. It enables the means of considering the situation of use not only in the design process, but in real time while the device is in use. In Human-Computer Interaction (HCI), we traditionally aim to understand the user and the context of use and create designs that support the major anticipated use cases and situations of use. In Context-aware computing on the other hand, making use of context causes a fundamental change: We can support more than one context of use that are equally optimal. At run time – when the user interacts with the application — the system can decide what the current context of use is and provide a user interface specifically optimized for this context. With context-awareness, the job of designing typically becomes more complex as the number of situations and contexts which the system will be used in usually increases. In contrast to traditional systems, we do not design for a single - or a limited set - of contexts of use; instead, design for several contexts. The advantage of the approach described in this work is that it can provide optimized and automated designs for a range of contexts.

The quality of context-aware systems, as perceived by the user, is directly related to the awareness mismatch, and a good design aims at designing systems with minimal awareness mismatch. A prerequisite for creating a minimal awareness mismatch is that the user understands what factors have an influence on the system.

In this work, we have proposed the idea of a self-supervised clustering called context aware clustering where the contextual information is used as feedback to improve clustering results. We have used the data from the GloVe corpus to cluster the Flag and Zoo datasets. As compared to traditional k-means clustering, our proposed context-aware clustering method considers dependencies in data at a higher contextual level. Thus clustering results are not just better but also give us an estimate of the context of the categorical attributes that contain words. This is important because it can be used to reveal the hidden patterns in all kinds of data sets.

The approach explained in this work is unique in the way it introduces context to clustering. Due to the lack of a method for measuring the context of a word, our approach cannot be compared or stated to be clearly superior to previous approaches. Any superiority, if present, shall be made clear only through empirical evidence in the future. The advantage lies in setting a standard for clustering categorical data (while retaining the context of their non-numerical attributes) by the use of a global representation of words made available by GloVe.

7. Acknowledgments

We extend our heartfelt gratitude to Dr. B. K. Tripathy, Senior Professor, VIT for his guidance and insight. His suggestions and feedback were invaluable and greatly assisted us in the writing of this paper.

We are also indebted to Jeffrey Pennington, Richard Socher and Christopher D. Manning of Stanford University for their work on Global Vectors (GloVe) for Word Representations. Without their research our work would not have been possible.

References

[1] P. Berkhin, A survey of clustering data mining techniques, in: Grouping multidimensional data, Springer, 2006, pp. 25–71.

[2] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics) 28 (1) (1979) 100–108.

[3] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data mining and knowledge discovery 2 (3) (1998) 283–304.

[4] S. Guha, R. Rastogi, K. Shim, and Rock: A robust clustering algorithm for categorical attributes, in: Data Engineering, 1999. Proceedings. 15th International Conference on, IEEE, 1999, pp. 512–521.

[5] J. Yuan, Y. Wu, Context-aware clustering, in: Computer Vision and Pattern Recognition, 2008. CVPR2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[6] A. M. Posonia, V. Jyothi, Context-based classification of xml documents in feature clustering, Indian Journal of Science and Technology 7 (9) (2014) 1355–1358.

[7] D. Ienco, R. G. Pensa, R. Meo, Context-based distance learning for categorical data clustering, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 83–94.

[8] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv: 1301.3781.

[9] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.URLhttp://www.aclweb.org/anthology/D14-1162

[10] X. Rong, word2vec parameter learning explained, arXiv preprint arXiv: 1411.2738.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American society for information science 41 (6) (1990) 391.

[12] K. Lund, C. Burgess, Producing high-dimensional semantic spaces from lexical co-occurrence, Behaviour Research Methods, Instruments, & Computers 28 (2) (1996) 203–208.

[13] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (Jul) (2011) 2121–2159.

[14] H. Chernoff, The use of faces to represent points in k-dimensional space graphically, Journal of the American Statistical Association 68 (342) (1973) 361–368.

[15] H. Hotelling, Analysis of a complex of statistical variables into principal components. Journal of educational psychology 24 (6) (1933) 417.

[16] P. Jaccard, Etude comparative de la distribution florale dans une portion des Alpes et du Jura, Impr.Corbaz, 1901.

[17] R. Bellman, R. Corporation, Dynamic Programming, Rand Corporation research study, Princeton University Press, 1957.URLhttps://books.google.it/books?id=wdtoPwAACAAJ

[18] D. J. Ketchen, C. L. Shook, The application of cluster analysis in strategic management research: an analysis and critique, Strategic management journal 17 (6) (1996) 441–458.

[19] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of computational and applied mathematics 20 (1987) 53–65.

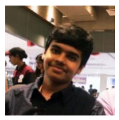
AUTHORS

Pulkit Juneja – B.Tech. Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (2017 Batch) Currently working at McKinsey and Co.

Hemant Jain – B.Tech. Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (2017 Batch)*Joining* University of Washington, MS in Data Science (2017-19)

Tanay Deshmukh – B.Tech. Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (2017 Batch)*Joining* University of Southern California, MS in Computer Science conc. in Data Science (2017-19)

Siddhant Somani – B.Tech. Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (2017 Batch) *Joining* Columbia University, MS in Computer Science (2017-19)



B. K. Tripathy – Senior Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India Areas of practice – Machine Learning, Data Mining, Artificial Intelligence, Clustering, Fuzzy Sets