2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)
Manipal University Jaipur, Dec 22-23, 2017

Vector Representation of Words for Sentiment Analysis Using GloVe

Yash Sharma, Gaurav Agrawal, Pooja Jain, *Member, IEEE,*Tapan Kumar *Senior Member IEEE*
Indian Institute of Information Technology, Kota
{*2014kucp1020,2014kucp1017,pooja.jain,tapan*}*@iiitkota.ac.in*

***Abstract*—Sentiment Analysis is one of the application of Natural Language Processing(NLP) methodology. The NLP enables to understand the common day to day language of the people. This understanding is extended to decipher the sentiments of the users and hence interpret the liking and disliking of the people. Vector representation of the words using unsupervised technique like Glove proves to be very effective in interpreting the meaning and hence the sentiments.**

***Keywords*—Glove, vector representation, Sentiment Analysis, Natural Language Processing, Machine learning**

I. Introduction

Natural Language Processing (NLP) is required for the computers to understand the common language of the people. The field of NLP is the backbone for robotics, speech recognition systems and sentiment analysis systems. The common day to day language of the people needs to be understood by the computer so that the latter analyses the sentiments of the users. The major steps in NLP are lexical analysis, syntactic analysis, semantic analysis, disclosure integration and pragmatic analysis. For the syntactic and semantic analysis, its necessary to decipher the meaning of the word and its usage. For this, it needs to be converted into vector space model. These vectors can be used as features in many applications like information retrieval [1], question answering [2] or document classification [3] and sentiment analysis as discussed in this paper. So, the current research work focuses on effective representation of words in vector space. A Vector Space Model translates collections of text into quantifiable data, a common way of vectorizing text documents [4] .The word context matrix captures word co-occurrences, taking into account the surrounding words within a window size. The distributional hypothesis states that words that occur in similar contexts have somewhat same meaning [5]. Neural network-based models have been proved to give better word representations over traditional distributional semantic methods such as LSA (Latent Semantic Analysis) [6].

The next section deals with the related work and the subsequent section summarizes the Glove. The section 4 deals with the proposed method for sentiment analysis and the fifth section gives the implementation results. Finally the sixth section concludes and gives the future work.

II. Related Work

The two learning models for vector representation of words generally used are: global matrix factorization methods, like latent semantic analysis (LSA) [7] and local context window methods, like the skip-gram model by Mikolov et al. [8].Both the models have drawbacks. The methods like latent semantic analysis efficiently utilize statistical information, they are not very efficient with word analogy tasks. Methods like skip-gram may have advantage of better analogy task, but they are unable to leverage the statistics of the corpus since they are trained on separate local context windows rather than on global co-occurrence counts. Glove and Word2vec are two models of low-dimensional vector representation of words that incorporates context.

A. Skip-gram Model

The vast majority of rule based and statistical NLP work considers words as atomic symbols: dictionary, meaning or run. In the vector space notation, it is a sparse vector: that is a vector with one at only a single position and zeroes at other remaining positions: [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]. This representation is known as one-hot vector representation whose disadvantage is space complexity and its failure to represent similarity between two words.

For e.g. consider two one-hot word vectors :

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0
That is, there doesn't exist a reliable method to determine the similarity between two words using this representation.

Word2Vec model(Skip-gram and CBOW), uses distributional similarity based approach for representing each word as a vector of d-dimension, where each element in vector is a real number.Skip-gram model tries to predict the surrounding word of every word , i.e. for each word(a.k.a. center word) "t", it predicts surrounding words(context words) in a window of radius m. This approach generates an objective function as to maximize probability of context word given the center word , i.e. ,

$$\prod_{t} \prod_{-m \leq j \leq m; j \neq 0} P(w_{t+j}|w_t; \theta),$$

where θ is the matrix of words in vector form. Maximizing above equation is same as maximizing the log of above equation. Hence, generally the log of the above equation is

treated as objective function. So, the equation becomes:

$$\sum_{t} \sum_{-m \leq j \leq m; j \neq 0} \log(P(w_{t+j}|w_t; \theta)),$$

Now, probability in skip-gram model is defined by usage of soft-max function, i.e. , for context word u_w and center word v_w, the probability is defined as:

$$P(u_w \mid v_w; \theta) = \frac{\exp(u_w.v_w)}{\sum_{u_c \epsilon C} \exp(u_c.v_w)},$$

where "c" ϵ Corpus "C". One of the disadvantage of using skip-gram model is that is fails to exploit the statistical information of corpus that is already present or can be easily computed due to which its complexity includes |C| , i.e. corpus size as a factor.

B. Continuous Bag of Words (CBOW)

This model tries to predict the center word using context words. It uses statistical information such as co-occurrence matrix to create vector for each word.
Notations of CBOW Model:

- $w^{(i)}$: Word (i) from the vocabulary (V).
- $W^{(1)} \in R^{n \times |V|}$: Input word matrix.
- $u^{(i)}$: ith column of $W^{(1)}$ i.e. the input word vector representation of the word $w^{(i)}$
- $W^{(2)} \in R^{n \times |V|}$: Output word matrix
- $v^{(i)}$: ith row of $W^{(2)}$,the output word vector representation of word $w^{(i)}$.
- n : Size of embedding space

Steps

- Generate one hot vectors ($x^{(i-C)}$, . . . , $x^{(i-1)}$, $x^{(i+1)}$, . . . , $x^{(i+C)}$) for the input context which is of size C.
- Get embedded word vectors for the given context

$$\begin{aligned} u^{(iC)} &= W^{(1)} x^{(i-C)}, \\ u^{(i-C+1)} &= W^{(1)} x^{(i-C+1)}, \\ &\ldots, \\ &\ldots, \\ &\ldots, \\ u^{(i+C)} &= W^{(1)} x^{(i+C)}. \end{aligned}$$

- Averaging those vectors

$$h = \frac{u^{(i-C)} \ + \ u^{(i-C+1)} \ + \ ... \ + \ u^{(i+C)}}{2C}$$

- Generating the score vector

$$z = W^{(2)} h$$

- Turning the score into probability

$$\hat{y} = softmax(z)$$

- To measure of the distance between two distributions y and $\hat{y}$. Cross entropy H($\hat{y}$,y) is used. The intuition for use of cross-entropy in discrete case can be derived from loss function:

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

- Considering our case, since y is one hot vector. Thus the above loss simplifies to simply:

$$H(\hat{y}, y) = y_i \log(\hat{y}_j).$$

- The optimization objective:

  minimize the value of J

$$= -\log P(w^{(i)} \mid w^{(i-C)}, ..., w^{(i-1)}, w^{(i+1)}, ..., w^{(i+C)})$$

$$= logP(v^{(i)} \mid h)$$

$$= \frac{exp(v^{(i)T} h)}{\sum_{j=1}^{|V|} exp(v^{(i)T} u^{(j)})}$$

$$= -v^{(i)T} h + \log \sum_{j=1}^{|V|} exp(v^{(i)T} u^{(j)})$$

C. Global Vectors (GloVe) model

Proposed by researchers at Stanford, the GloVe model tries to generate the vector representation of word by using similarity between words as an invariant. The model exploits approaches given by two different models, namely, Skip-gram and Continuous Bag of Words Model (CBOW).

Issue with the former model is computational time(although good accuracy) , while latter had low accuracy(although computationally efficient). What Glove tries to do is to combine the approaches presented by two model and it has proved to be more accurate and efficient than those two.

Before understanding GloVe model, lets describe somewhat more about vector representation of words: Each of the models

are used to generate a vector of a fixed dimension (say d) for each word. Each model uses similarity between two words as an invariant i.e. they assume that words which occur in similar contexts are more likely to share same meaning. For e.g. , given two sentences :
"Tell me the time from the watch you have."
"Can you please check the clock for what time is it?"
One thing to observe from these two sentences is that words watch and clock share some common words in their contexts like time etc. Hence, it is more likely that both share same meaning too.

Natural Language Processing involves machine learning tools like SVM(Support Vector Machine) or ANN(Artificial Neural Network). SVM plays an important role in classification of input data [9].

III. Details of GloVe

Lets' denote some terminologies first before moving to formulation of GloVe :

- Let matrix of word to word co-occurrence counts be denoted by X, whose values X_{ij} stores the number of times the word j has occurred in context of the word i.
- Let X_i = $\sum_k X_{ik}$ denote the number of times any word appeared in context of the word i.
- Finally, let P_{ij} = $P(j \mid i)$ = X_{ij}/X_i be the probability that word j appear in the context of word i.

The authors of the original paper began with a example that shows how certain aspects of meaning could be extracted directly from their co-occurrence probabilities. Let us consider two words i and j that are related to each other in a context; for instance, let us suppose that we pick cricket as a matter of subject, so let i = duck and j = boundary. The closeness between these words can be observed by examining the ratio of their co-occurrence probabilities with various probe words, k. For words k related to duck but not boundary, say k = out, we expect that the ratio P_{ik}/P_{jk} will be large. Similarly, for words k related to boundary but not duck, say k = six, the ratio should be small. For words k like score or captain, that are either related to both duck and boundary, or to neither, the ratio should be closer to 1.

Former argument indicates that ratio of co-occurrence probabilities can be used as a starting point to determine the similarity between those words. The ratio P_{ik}/P_{jk} depends on three words i, j, and k, therefore most general model will look like,

$$F(w_i, w_j, \hat{w_k}) = \frac{P_{ik}}{P_{jk}},$$

where w$\in R^d$ are word vectors and $\hat{w} \in R^d$ are separate context word vectors.

Since vector spaces are inherently linear structures, considering vector differences.

$$F((w_i - w_j)^T, \hat{w_k}) = \frac{P_{ik}}{P_{jk}}$$

By using group theory and algebraic equations they concluded to the fact that the above equations can be simplified to :

$$F((w_i - w_j)^T \hat{w_k}) = \frac{F(w_i^T \hat{w_k})}{F(w_j^T \hat{w_k})},$$
$$where, F(w_i^T \hat{w_k}) = P_{ik} = \frac{X_{ik}}{X_i}$$
$$w_i^T \hat{w_k} = log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$
$$\exp(w_i^T . \hat{w_k}) = P_{ik} = \frac{X_{ik}}{X_i}$$

where '.', is the dot product between two vectors w_i and w_k and exp is the exponent function. Simplifying above equation leads to :

$${w_i}^T . \hat{w_k} = log(P_{ik}) = log(X_{ik}) - log(X_i),$$

and assuming log(X_i) as a constant or a bias term of word i we modify above equation to:

$${w_i}^T . \hat{w_k} + b_i + \hat{b_k} = log(X_{ik}),$$

where b_i is the bias for word i and $\hat{b_k}$ (we also have to add bias for k if we have added it for i to maintain symmetry) is the bias for word k.

Now if we see the final equation from machine learning perspective, we can see that RHS of the above equation can be determined from corpus and we only need to update LHS in order to make it equal to RHS. Hence LHS can be thought of as a hypothesis(h) and RHS can be regarded as known output(y).

The cost function then becomes (using least squared method) :

$$J = \sum_{i,k} ({w_i}^T . \hat{w_k} + b_i + \hat{b_k} - \log(X_{ik}))^2,$$

and there is a need to minimize this cost function (now any gradient descent technique can be used to achieve it, but generally stochastic gradient descent or adaptive gradient descent is used).

But before applying gradient descent the cost function need to be modified a little bit by giving some weights to each pair of words (i.e. there are some words known as stop words , which co-occur with every other word but constitute very less in representing meaning of the word they are with, for e.g.

word 'the' occur with most of the vocabulary words), hence the cost function becomes :

$$J = \sum f(X_{ij})(w_i{}^T.\hat{w_k} + b_i + \hat{b_k} - \log(X_{ik}))^2,$$

where f(Xij) is some weight associated with co-occurrence of word i with j. Normally f is defined as :

$$\begin{array}{l} (\frac{x}{x_{max}})^{\alpha}, \quad if\ x\ <\ x_{max}\ , \\ 1,\ otherwise \end{array}$$

Now if partial derivative of J is computed with w_i it will done as follows :

$$\frac{\partial J}{\partial w_i} = \sum_{k=1}^{d} \frac{\partial J}{\partial w_{i,k}},$$

where d is the dimension of word i.

If w_i is a vector for example of (x_1, x_2, x_3) , then $\frac{\partial J}{\partial w_i}$ will be :

$$(\frac{\partial J}{\partial w_{i,1}} = (1,0,0)) + (\frac{\partial J}{\partial w_{i,2}} = (0,1,0)) + (\frac{\partial J}{\partial w_{i,3}} = (0,0,1)),$$

which thus is a vector of (1,1,1) .

Now we know how to compute the derivative with respect to each word we can easily apply gradient descent to with some learning rate alpha(= 0.5), to train our word2Vec model.

Results : After training the model, we tried to extract the words with similar meaning by providing a random word as an input and here are the results : For given word company predicted similar words are : Market , business, products, industry, stock, inc. etc.

For more in depth explanation refer this paper [].

IV. Proposed Method for Sentiment Analysis

Sentences have temporal aspect i.e. Each word in a sentence depends greatly on what came before and comes after it. In order to account for this dependency, Recurrent neural network is used.

In RNN, each word in an input sequence will be associated with a specific time step. In effect, the number of time steps will be equal to the max sequence length. The idea behind RNNs is to make use of sequential information.

In a traditional neural network it is assumed that all inputs (and outputs) are independent of each other, but for predicting the next word one should know that which words have came before that word. RNN are called recurrent as they perform the similar task for each and every element of the sequence, with the output depending on previous computations. Another way to see RNN is that they have memory that captures information about what has been already calculated till now.

Thus after obtaining word vectors, set of sentences are taken (these will be the sentences that have either positive or negative sentiments) and converted to word vectors which are then given input to RNN(Recurrent Neural Networks).

Notation

- x_t is a input at a time step t. For example, x_t can be a one hot word vector corresponding to the second word of the sentence.
- s_t is hidden state at time step t. Its the memory of the network.
- o_t is the output at time step t. For example,if we want to predict the next word in the sentence it would be a vector of probabilities across our vocabulary.

s_t is calculated based on the previous hidden state and the input at current step.

$$s_t = f(Ux_t + Ws_{t_{-1}}).$$

The function f is generally a non-linearity like tanh or ReLU(Rectifier Linear Unit). s_{-1}, which is used to calculate the first hidden state and is typically initialized to all zeroes.

$$o_t = \mathrm{softmax}(Vs_t)$$

V. Implementation and results

GloVe model is implemented as given in its paper and word vectors are created. To calculate similarity between two words dot product of their vectors is done. To get most similar words, the dot product of vector is taken with all other vectors and words with vectors giving maximum dot product is considered similar.

These resulting vectors are much accurate at answering analogy questions of the form I is to J as K is to ?. Let us say for example, uncle is to aunt as man is to ? (woman) using vector offset method based on the cosine similarity distance. This vector representation can also answer Queen + Man - Woman = ? question and it arrives at the result King ! and all this knowledge is simply coming from looking at lots of word in context and with no any other information provided about their semantics.

For example: Here pre trained vectors are used for getting higher accuracy: the vectors for (Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab)) is used with dimension = 50.

Since the vectors have high dimensions, Here t-Distributed Stochastic Neighbor Embedding (t-SNE) is used which is a (prize winning) technique for dimensionality reduction and it is well suited for the visualization of high dimensional data, here the vector of 50 dimension is converted to 2 dimension

vector and it can be easily visualised on a graph.

For sentiment analysis: First a corpus is given to this model, the GloVe then generates word vectors for each word (here dimension = 50), then the sentence is converted to their corresponding word vectors and a 3 dimensional vector is given to Recurrent neural network (RNN) layer.

The output label classes for RNN is 2(positive or negative). The model is than trained to give sentiment analysis for the query.Given word = January, the GloVe shows the similar words that are months of year and can be visualized using t-SNE as shown in the Figure 1. t-SNE is used to reduce the dimensions of the vector like input vector is of 50 dimensions and the output vector is of 2 dimensions as shown in the Figure 1. The X axis represents Feature 1 weight and Y axis represents feature 2 weight.

VI. Conclusion and Future Work

The current research paper gives a comparative study of the different models like skip gram and Continuous Bag of Words.Then the authors implement Glove and give the possible use of it in sentiment analysis. The word vectors obtained from Glove method is fed into RNN.
Currently sentiment analysis is doing binary classification(Positive and Negative Sentiments), but this approach can be extended to perform multi class classification such as for e.g. it can be used to determine the mood of a person based on the sentence provided. Also since these word vectors capture meaning of words they can be used to other Natural Language Processing problems like Machine Translation, Summarization, Question Answering , Information Extraction.

References

[1] Maya Carrillo, Chris Eliasmith, and Aurelio López-López. Combining text vector representations for information retrieval. In *International Conference on Text, Speech and Dialogue*, pages 24–31. Springer, 2009.

[2] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM, 2003.

[3] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[4] Paula Lauren, Guangzhi Qu, Guang-Bin Huang, Paul Watta, and Amaury Lendasse. A low-dimensional vector representation for words using an extreme learning machine. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1817–1822. IEEE, 2017.

[5] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[7] Peter Wiemer-Hastings, K Wiemer-Hastings, and A Graesser. Latent semantic analysis. In *Proceedings of the 16th international joint conference on Artificial intelligence*, pages 1–14, 2004.

[8] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751, 2013.

[9] Joey Pinto, Pooja Jain, and Tapan Kumar. Hadoop distributed computing clusters for fault prediction. In *Computer Science and Engineering Conference (ICSEC), 2016 International*, pages 1–6. IEEE, 2016.

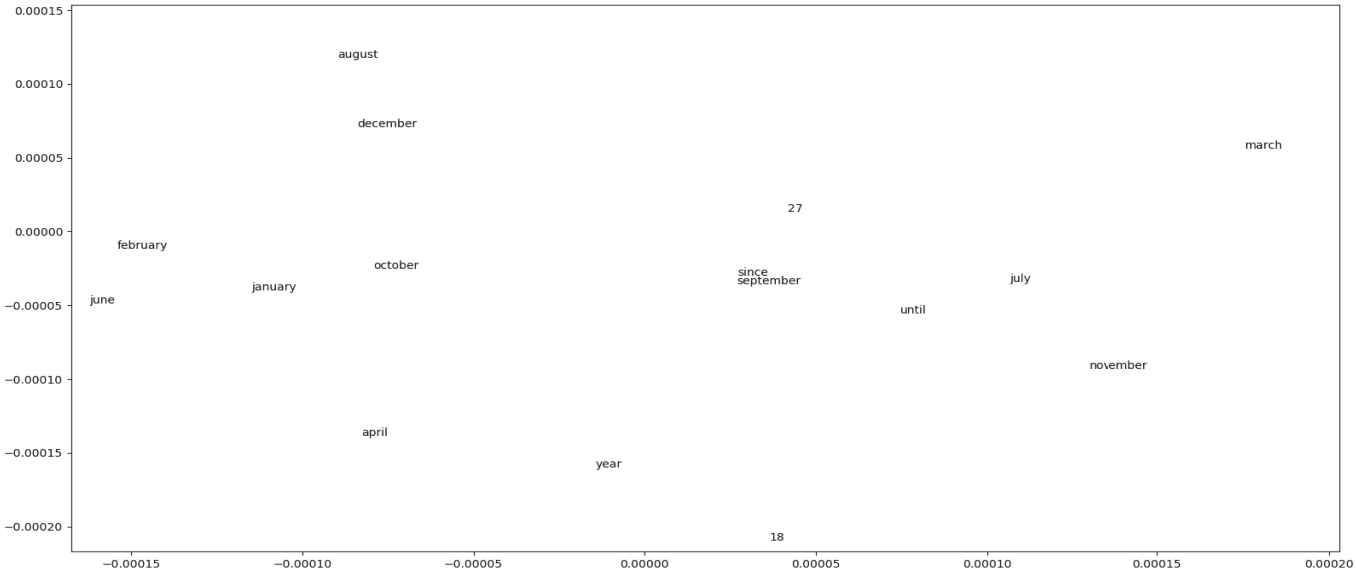
0.00015
0.00010
0.00005
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
august
december
march
27
february
october
since
september
july
january
june
until
november
april
year
18
−0.00015
−0.00010
−0.00005
0.00000
0.00005
0.00010
0.00015
0.00020

Fig. 1.