Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic

Conference Paper · December 2020

DOI: 10.1109/BigData50022.2020.9378337

2 authors:

Mahdi Naser Moghadasi
Texas Tech University
11 PUBLICATIONS **80** CITATIONS
SEE PROFILE

Yu Zhuang
Texas Tech University
83 PUBLICATIONS **1,179** CITATIONS
SEE PROFILE

Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic

1st Mahdi Naser Moghadasi
Computer Science Department
Texas Tech University
Lubbock,TX, USA
Mahdi.Moghadasi@ttu.edu

2nd Yu Zhuang
Computer Science Department
Texas Tech University
Lubbock,TX, USA
Yu.Zhuang@ttu.edu

***Abstract*—Text classification is considered as one of the primary task in many Natural Language Processing (NLP) applications. In industrial applications of NLP, sentimental analysis is a task to understand how satisfied a user is after receiving a service or buying a product. The traditional approach is to convert a text into a format of numeric vector before feeding into machine learning algorithm. This representation of a word refers to word embedding. However the traditional embedding methods often model the syntactic context of words but ignore the sentiment information of text [1]. This can impact on the accuracy of a classification model to predict the correct sentimental score for a text. In this paper, we present Sent2Vec, an alternative embedding representation that includes the sentimental semantic of a sentence in its embedding vector. We utilized the unsupervised Smoothed Inverse Frequency (uSIF) sentence embedding method in the Sent2Vec neural network over a multi million samples dataset. The new sentence embedding presented, can be used as features in downstream (un)supervised tasks, which also leads to better or comparable results compared to sophisticated methods. Furthermore, with a simple logistic regression classifier, Sent2Vec reaches competitive performance to state-of-the-art results on several datasets when combined with GloVe(6B).**

***Index Terms*—Word Embedding, Sentimental Analysis, NLP**

I. Introduction

Recent research on deep learning to learn a distribution representation vector over words has achieved popularity in Natural Language Processing (NLP). The technique to learn this distributed representation vector is referred to word embedding. In a solid definition word embedding is a technique to learn continuous low-dimensional vector space representations of words by leveraging the contextual information from large corpora. Word2Vec [2] is a neural network model to learn the center word by its surrounding (context) words in a statement. It makes words of similar meanings closer over continuous vector space. Despite its simplicity, the model has generated high quality in NLP tasks such as language modeling, text understanding and machine translation. Other popular representations range from the simplest BOW and its term-frequency based variants [3], language model based methods [4]–[6], topic models [7], [8], Denoising Autoencoders and its variants [9], [10], and distributed vector representations [11], [12], [13]. Another prominent line of work on word embedding includes GloVe [14] and fastText [15]. However, despite the rising popularity regarding the use of word embeddings in different natural language processing task, they often fail to capture the emotional semantics the words convey. To overcome this challenge and include effective information into the word representation, Tang et al [1] proposed Sentiment-Specific Word Embeddings (SSWE) which encodes both positive/negative sentiment and syntactic contextual information in a vector space. This work demonstrates the effectiveness of incorporating sentiment labels in a word level information for sentiment-related tasks compared to other word embeddings. Yet, it only focuses on binary labels, which weakens its generalization ability on other tasks. Yu et al. [16] instead proposed to refine pre-trained word embeddings with a sentiment lexicon, observing improved results. Felbo et al. [17] achieved good results on affect tasks by training a two-layer bidirectional Long Short-Term Memory (bi-LSTM) model, named DeepMoji, to predict emoji of the input document using a huge dataset of 1.2 billions of tweets. However, collecting billions of tweets is expensive and time consuming for researchers. Emo2vec [18] is a word-level representation that encodes emotional semantics into fixed-sized, real-valued vectors. Labutov and Lipson [19] proposed a method that takes an existing embedding and labeled data as input and produces an embedding in the same space, but with a better predictive performance in the supervised task.

Due to the progress of the word embedding as the skeleton for many NLP tasks, researchers have tried to compute embeddings that capture the semantics of word sequences (phrases, sentences, and paragraphs), with methods ranging from simple additional composition of the word vectors to sophisticated architectures such as convolutions neural networks and recurrent neural networks. [20]–[22]. Arora et. al. [23] proposed a strong baseline for computing sentence embeddings: take a weighted average of word embeddings and modify with Singular Value Decomposition (SVD). This simple method outperforms complicated approaches such as Long Short Term Memory (LSTM) on similarity tasks. The technique is called *smooth inverse frequency (SIF)*. In a more sophisticated version of SIF method, Ethayarajh et al. [24] introduced *unsupervised smoothed inverse frequency (uSIF)*. uSIF is based on the principle that more frequent words should be down-weighted because they are typically less informative.

In this paper we present Sent2Vec, a sentimental embedding model. Our method is different from previous approaches mainly due to 1) Sent2Vec trained to generate a sentence embedding in one-shot while previous methods are mainly based on word embeddings and an intermediate method is needed to convert word embedding to sentence embedding. 2) We apply Glove embedding in creating the new sentiment embedding. The Glove data representation provides a richer vocabulary list compared to other methods. This is important because if the dictionary of the dataset is only limited to one domain it will not be feasible to evaluate it on other datasets. 3) the size of the dataset used to train the Sent2Vec is smaller than the previous works. For example in Deepmoji algorithm, the author utilized over 1 billion tweets to train the network while we applied 1 million records of Amazon dataset which substantially smaller. We explain different word and sentence embedding in sections II and III. We discuss about the algorithm of Sent2Vec in section IV and evaluate our method over different datasets in section V.

II. Word Embedding

Statistical NLP has been developed as the dominant option for modeling complex natural language tasks. However, at its beginning, it often suffers from the major curse of dimensionality while learning joint probability functions of language models. This led to the motivation of learning distributed representations of words existing in low dimensional space. Word embedding methods are a set of algorithms to represent a word in a format of a vector with fixed length. That is a continuous vector in a low dimension space that embeds the semantic and lexical features of a word. In this section, we elaborate different word embedding techniques following by the mathematical formulas behind them.

A. Word2Vec

There are two *Word2Vec* models: skip-gram and continuous bag-of-words (CBOW). They both manage to capture interactions between a centered word and its context words. However, they do it differently, and somehow oppositely. While skip-gram models the distribution of context words given the centered word, CBOW is concerned about predicting the centered word given its context. The probability of the context word w_t given the centered word $\hat{r}$ is calculated by a softmax function:

$$P(w_t|\hat{r}) = \frac{exp(w_t^T \hat{r})}{\sum_{w \in W} exp(w^T \hat{r})} \tag{1}$$

where W is the set of all context word vectors. Notice that $\hat{r}$ is neither an element of W nor computed from elements of W. Elements of W will be called output vectors and $\hat{r}$ will be computed from a different set consisting of input vectors. *Word2Vec* models' cost functions (for one target word) minimize the negative log-likelihood of the target word vector given its corresponding predicted word:

$$\mathcal{L}(w_t, \hat{r}) = -\log P(w_t|\hat{r}) = \log \left(\sum_{w \in W} exp(w^T \hat{r}) \right) - w_t^T \hat{r} \tag{2}$$

To illustrate better understanding of the cost functions, the gradient with respect to w of $\mathcal{L}(w_t, \hat{r})$ is:

$$g_1(w, w_t, W, \hat{r}) = \frac{\partial}{\partial w} \mathcal{L}(w_t, \hat{r}) = \hat{r}(P(w|\hat{r}) - I\{w = w_t\}) \tag{3}$$

where $I\{.\}$ is the indicator function.

The gradient with respect to $\hat{r}$ is:

$$g_2(w_t, W, \hat{r}) = \frac{\partial}{\partial \hat{r}} \mathcal{L}(w_t, \hat{r}) = \sum_{w \in W} [P(w|\hat{r})w] - w_t \tag{4}$$

B. GloVe

Global Vectors for Word Representation (GloVe) is a word embedding model developed by Pennington et al. [14]. The authors illustrated how the training objective of skip-gram uses information from the occurrence matrix of a corpus, which explains why the skip-gram model, although only uses local context window, can produce word embeddings that capture global relationships.

Concretely, let X be the word-word co-occurrence counts matrix of the corpus where the (i, j) entry is number of times word i and word j co-occur and entries X_{ij} define the number of times word j occurs in the context of word i. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of 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 relationship of word i and word j is determined based on the ratio of the probabilities of their co-occurrences with a context word k. The ratio P_{ik}/P_{jk} depends on three words i, j, and k, the most general model takes the form,

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

Function F aimed to encode the information in the ratio P_{ik}/P_{jk} in the word vector space. According to linear algebra the author enforced a restriction on the (5), such a way that F depends only on the difference of the two target words.

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{P_{ik}}{P_{jk}} \tag{6}$$

Since in the occurrence matrix, the roles of the rows and columns are interchangeable, symmetric has been enforced to the above equation:

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{F(w_i^T w_k^{'})}{F(w_j^T w_k^{'})} \tag{7}$$

From (6) and (7):

$$F(w_i^T w_k^{'}) = P_{ik} \tag{8}$$

The answer to the (8) is F = exp, which entails to:

$$w_i^T w_k^{'} = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{9}$$

From (9), it sets the following weighed least square objective:

$$J = \sum_{i,j} f(X_{ij})(w_i^T w_j^{'} + b_i + b_j^{'} - \log X_{ij}) \quad (10)$$

where $f(.)$ is a weight function and $b, b^{'}$ are bias terms.

According to the training objective of *skip-gram*:

$$J = -\sum_i \sum_{j \in context(i)} \log Q_{ij} \quad (11)$$

where $Q_{ij} = \frac{\exp(w_i^T w_j^{'})}{\sum_k exp(w_i^T w_k^{'})}$, $X_i = \sum_k X_{ik}$ and $P_{ij} = X_{ij}/X_i$, J can be rewritten as:

$$J = -\sum_{i,j} X_{i,j} \log Q_{i,j} = \sum_i X_i H(P_i, Q_i) \quad (12)$$

where H is the cross entropy between two distributions.

The *GloVe* model is more advanced than (12) in two ways:

(a) The weight function is not X_i but an arbitrary function $F(X_{ij})$.

(b) The cross entropy has several drawbacks. First, it assigns more probability mass to unlikely event. Second, calculating the normalizing constant involves summing over the entire vocabulary, which is costly. To address these problems, the *GloVe* model chooses a quadratic loss instead.

C. fastText

Joulin et al. [15] introduced an embedding based on the N-gram features. fastText architecture is a simple linear model with rank constraint, Fig. 1. The input to the network is a look up table over the words. This builds a weight matrix, next the word representations are averaged into a text representation, which is in turn fed to a linear classifier. The model uses a softmax function f to compute the probability distribution over the predefined classes. Considering N number of documents, this causes minimizing the negative log likelihood over the classes:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log\left(f\left(BAx_n\right)\right) \quad (13)$$

where x_n is the normalized bag of features of the n-th document, y_n is the label, A and B are the weight matrices. The linear classifier computation explodes when the number of class is large. In other words, the computational complexity is $O(kh)$ where k is the number of classes and h the dimension of the text representation. Hence, to reduce the computation complexity the author explained applying a hierarchical softmax to reduce the training time. During training, the computational complexity drops to $O\left(h \log_2(k)\right)$. Each node is associated with a probability that is the probability of the path from the root to that node. If the node is at depth l + 1 with parents n_1, . . . , n_l , the probability is

$$P\left(n_{l+1}\right) = \prod_{i=1}^{l} P\left(n_i\right) \quad (14)$$

This means that the probability of a node is always lower than the one of its parent.

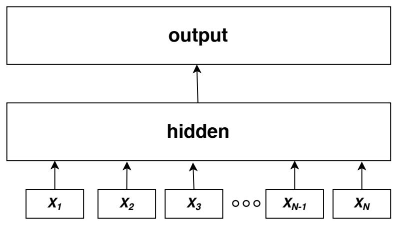


Fig. 1: Model architecture of fastText for a sentence with N ngram features $x_1, ..., x_N$.

III. Sentence Embedding

Sentence embedding in an ideal definition which can be explained as the representation of the semantic of a sentence in a format of a single numeric vector. Sentence embedding assists to understand the intention of the sentence without calculating individually the embeddings of the words. In this section we illustrate the sentence embedding algorithms to apply over word embeddings to build a single numeric vector representing the sentence.

A. Smooth Inverse Frequency (SIF)

The main idea behind SIF [23] is to compute the weighted average of the word vectors in the sentence and then remove the projections of the average vectors on their first singular vector. As it is shown in the Algorithm 1, the model accepts a word embedding v_w per each word in a sentence, a set of sentence S, an scalar parameter α and $p(w)$ the (estimated) word frequency as a probability. Here the weight of a word w is calculated according to $a/(a+p(w))$. This weight multiplied over the original word embedding and averaged to build a vector v_s. Finally, the projection of v_s over its first singular vector (presented as u in the Algorithm 1) is removed to build the new updated v_s as the sentence embedding for sentence s.

Algorithm 1 SIF - Sentence Embedding

```
Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$,
       parameter $a$ and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$
       of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
for sentence $s$ in $\mathcal{S}$ do
    $v_s \leftarrow \frac{1}{|s|}\sum_{w \in s} \frac{a}{a+p(w)} v_w$
Form a matrix $X$ whose columns are $\{v_s : s \in \mathcal{S}\}$ , and let $u$ be its
first singular vector
for sentence $s$ in $\mathcal{S}$ do
    $v_s \leftarrow v_s - uu^\top v_s$
```

Different values for a as the hyper-parameters yielded to be closer to the best results and an even wider range can achieve significant improvement over unweighted average [23].

B. unsupervised Smoothed Inverse Frequency (uSIF)

uSIF builds upon the random walk model proposed in [25] by setting the probability of word generation inversely related to the angular distance between the word and sentence embeddings. The author discussed about the interesting effect

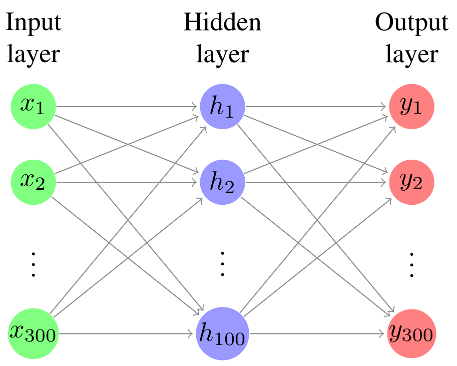


Fig. 2: Sent2Vec Neural Network

of word vector length on the probability of a sentence being generated in SIF [23]. To address the effect of word vector length, it was proposed a random walk model where the probability of observing a word w at time t is inversely related to the angular distance between the time-variant discourse vector $c_t \in \mathbb{R}^d$ and the word vector $v_w \in \mathbb{R}^d$:

$$p(w \mid c_t) \propto 1 - \frac{\arccos(\cos(v_w, c_t))}{\pi}$$
$$\text{where } \cos(v_w, c_t) \triangleq \frac{v_w \cdot c_t}{\|v_w\|_2 \cdot \|c_t\|_2}$$

where $\arccos(\cos(v_w, c_t))$ is the angular distance. uSIF and SIF are all based on the principle that more frequent words should be down-weighted because they are typically less informative. We refer the readers to [24], [25] and [23] for more mathematical proofs of SIF and uSIF methods.

IV. Sent2Vec

In this section, we present the Sent2Vec neural network that transforms a traditional GloVe embedding to a sentimental embedding representation. The skeleton of Sent2Vec neural network is displayed in Fig. 2.

The model receives a 300-dimension word embedding vector $\mathcal{X}$ as the input, where each dimension in $\mathcal{X}$ is a numeric value denoted by x_i and $1 < i < 300$:

$$\mathcal{X} \triangleq \{x_1, x_2, \ldots, x_{300}\} \tag{15}$$

The network turns $\mathcal{X}$ into a sentimental 300-dimension vector $\mathcal{Y}$ where y_i corresponds to a numeric value in the sentimental vector. We refer to $\mathcal{Y}$ as the true vector.

$$\mathcal{Y} \triangleq \{y_1, y_2, \ldots, y_{300}\} \tag{16}$$

We set the hidden layer as a fully connected dense layer with 100 neurons after several configuration evaluation. We used *relu* and *sigmoid* activation functions for the hidden and the output layers, respectively. This can be explained as below:

$$\mathcal{H} = \mathcal{F}_{relu}(\mathcal{W}_1\mathcal{X} + \beta_1) \tag{17}$$

$$\mathcal{Y} = \mathcal{F}_{sigmoid}(\mathcal{W}_2\mathcal{H} + \beta_2) \tag{18}$$

where $\mathcal{H}$ is the output of the hidden layer, $\mathcal{W}$ and β are weight matrices and bias vector respectively. We have:

$$\begin{aligned} \mathcal{F}_{relu}(z) &= \left\{ \begin{array}{cc} z & z > 0 \\ 0 & z <= 0 \end{array} \right\} \\ \mathcal{F}_{sigmoid}(z) &= \frac{1}{1+e^{-z}} \end{aligned} \tag{19}$$

The loss function to be optimized is the cross entropy for the 300 dimension output, defined as below:

$$\text{Loss} = -\frac{1}{300}\sum_{i=1}^{300} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{20}$$

$\hat{y}_i$ is the i-th predicted scalar value by the model and y_i is a numeric value in the output vector $\mathcal{Y}$. Accordingly, the output of the neural network $\mathcal{Y}$, is a vector of numbers between 0 and 1. The example below illustrates a hypothesis case that a sentimental embedding vector $\hat{\mathcal{Y}}$ is predicted by Sent2Vec neural network model for an input sentence embedding vector $\mathcal{X}$.

$$\underbrace{-0.10, \overbrace{0.013}^{x_i}, ..., 0.068, -0.164}_{\mathcal{X}} \Rightarrow \underbrace{0.01, \overbrace{0.5}^{\hat{y}_i}, ..., 0.06, 0.13}_{\hat{\mathcal{Y}}}$$

We selected the Amazon review dataset that has been mostly used as a large dataset classification benchmark in [26]–[28] for our training dataset. The dataset included 3 million reviews for training and 650,000 for testing. For each review there is a rating label that corresponds to user satisfaction for the product under review. These labels construct a set of target classes labeled from 1 to 5 where 1 is correspondent to reviews with least satisfaction and 5 with most satisfaction. We built a look up table according to these labels with a score associated to it, displayed in Table I. The scores in the Table I are optimized values to maximize distance margin between classes in the vectors $\mathcal{Y}$. As shown in Fig. 3, to preprocess the data we required to construct the sentence embedding for each review. We employed uSIF algorithm over word-embedding GloVe(6B) (refer to Table II) to build sentence embedding required by the neural network. In this experiment, all the word embedding representations mentioned in the Table II have 300 dimensions. Each sentence embedding vector $\mathcal{X}$ represents an embedding for a review from the training dataset. In this research, a review is treated as a sentence that is fed to the Sent2Vec neural network. Eventually, to build the corresponding vector $\mathcal{Y}$, we multiplied vector $\mathcal{X}$ with an score from look up Table I and passed as input to a sigmoid function to present values in the same range the network expected. Consequently, an embedding vector $\mathcal{Y}_+$ with positive sentiment has a larger dissimilarity to a vector with negative sentiment $\mathcal{Y}_-$. Fig. 4 illustrated original and sentimental embedding for the first 20 dimensions predicted by the Sent2Vec for a negative and a positive review. We will analyze the Sent2Vec results in more detail in the following section.

TABLE I: Sentimental Scores

Label	Score
1	-1000
2	-100
3	+1
4	+100
5	+1000

TABLE II: Word Embedding Description

Word embedding	Description
GloVe(6B)	Pre-trained vectors based on Wikipedia. (6B tokens)
GloVe(840B)	Pre-trained vectors based on Common Crawl (840B tokens)
GloVe(42B)	Pre-trained vectors based on Common Crawl (42B tokens)
Word2Vec	Pre-trained vectors trained on a part of the Google News dataset
fastText	Pre-trained vectors over 1 million word vectors trained on Wikipedia
Sent2Vec-1M	Sentimental vector for GloVe(6B) over 1,000,000 samples of Amazon dataset
Sent2Vec-300K	Sentimental vector for GloVe(6B) over 300,000 samples of Amazon dataset
SSWE	Sentiment-Specific Word Embedding for Twitter Sentiment Classification [1]
DeepMoji	Using millions of emoji occurrences to learn any-domain representations for detecting sentiment emotion and sarcasm [17]
Emo2Vec	Emo2Vec: Learning Generalized Emotion Representation by Multi-task Training [18]

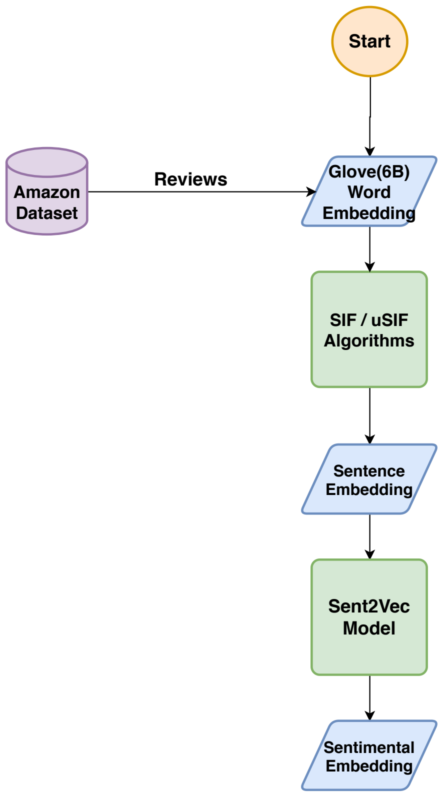


Fig. 3: Training Process of Sent2Vec Neural Network

V. Experiment

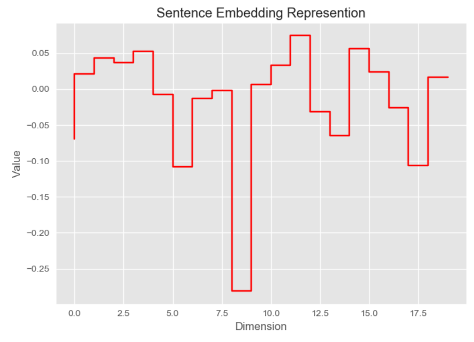
In this section, we discuss the performance of the Sent2Vec neural network with the other competitive representations. We applied different word embedding representations as shown in the Table II to build a sentence embedding through uSIF algorithms. As Fig. 5 details the test processing steps, the Amazon review data converts to word tokens to construct a word embedding per each word. In the next step the word embedding transforms to sentence embedding according to uSIF algorithms. We trained a logistic regression classifier over those sentence embeddings and measured the performance of the classification. We employed 3 million reviews for the training and 650,000 reviews for the testing in the logistic regression with different embedding representations shown in Table II. The reader is referred to (Zhang et al [26]) for more details on the construction of the Amazon dataset. In our implementation we created two Sent2Vec models based on the different training data size. Sent2Vec-1M and Sent2Vec-300K are neural networks with the same topology but trained over 1,000,000 and 300,000 samples respectively. Table III shows the logistic regression classification results of Sent2Vec against different embedding representations. We measured the performance of the logistic regression through F1 score [29].

In Table III, we collected the performance metrics for the logistic regression using uSIF and noticed the Sent2Vec representations have achieved the highest accuracy among all sentence representations. The Sent2Vec-1M has the highest accuracy across all candidates. Another observation of the results of Table III is that while Sent2Vec-300K trained with only 10% of the full Amazon dataset it increases the GloVe(6B) representation by 4% with uSIF sentence embedding. Furthermore, it was showing that performance is further raised once we employed more training data for Sent2Vec model in Sent2Vec-1M.

TABLE III: Logistic Regression performance with uSIF sentence embedding

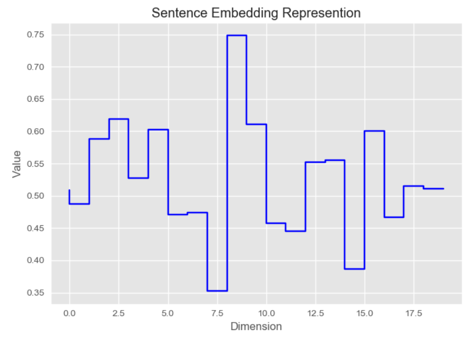
Representation	Accuracy
GloVe(6B)	48
GloVe(840B)	47
GloVe(42B)	47
Sent2Vec-1M	**55**
Sent2Vec-300K	50
Word2Vec	43
fastText	51
SSWE	39
EMO2Vec	39
DeepMoji	53

For the second experiment, we created sentence embedding of a subset of samples using uSIF for each candidate in Table II and passed the embeddings into the K-Means clustering algorithm with K=5 as illustrated in Fig. 5. Finally we measured the silhouette scores [30] per each cluster. The silhouette score is a metric to measure cohesion between samples of a cluster compared to other clusters. The silhouette ranges from -1 to $+1$, where a high value indicates that the sample is

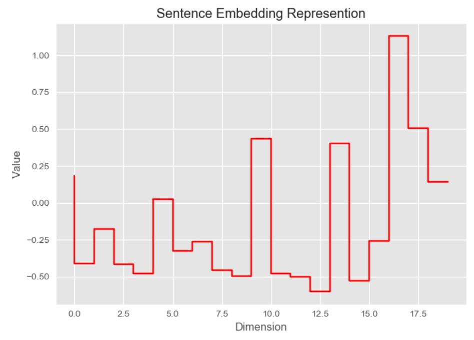


Original Negative Embedding

$\Longrightarrow$

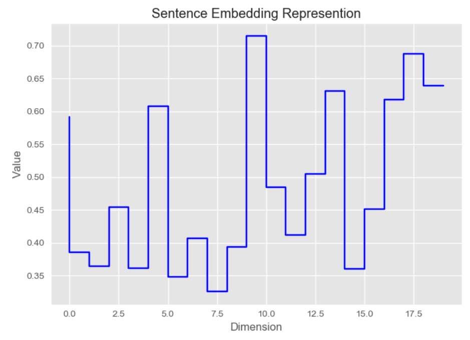


Sentimental Negative Embedding



Original Positive Embedding

$\Longrightarrow$



Sentimental Positive Embedding

Fig. 4: Embedding Transformation to Sentimental Embedding

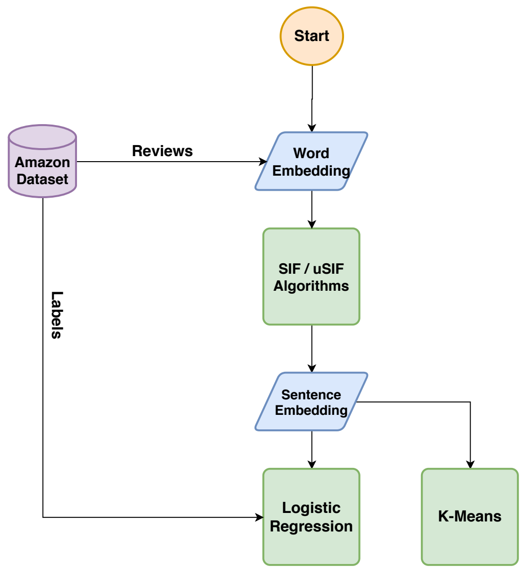


Fig. 5: Testing process of different word embeddings

well matched to its own cluster and inadequately matched to neighboring clusters. For detail on the silhouette scores, readers may refer to [30].

Table IV shows the silhouette score per each representation. We notice the Sent2Vec-1M has achieved the highest silhouette score. Furthermore, we plotted each of these clusters in the Fig. 6. The figures demonstrate how each cluster has been formed from its embedding representation of 10000 samples. Furthermore, our analysis according to the Table IV and Fig. 6 indicates the Sent2Vec-1M provides sentence embedding representations with a better cohesion for the data, separable boundaries between classes and highest silhouette score for clusters. The data showed that in the clustering experiment, Sent2Vec model produces representation that yields to a better clustering form and consequently a higher silhouette score comparing other embedding representations.

TABLE IV: Silhouette score per cluster

Representation	Silhouette score
GloVe(6B)	0.0394
GloVe(840B)	0.0640
GloVe(42B)	0.0348
Sent2Vec-1M	**0.2536**
Sent2Vec-300K	0.2228
Word2Vec	0.0393
fastText	0.1138
SSWE	0.0453
EMO2Vec	0.0634
DeepMoji	0.0486

To extend our analysis beyond the scope of traditional logistic regression we trained the embedding representations over a neural network algorithm as neural networks receive

TABLE V: Neural Network Classifier

Representation	Accuracy
Glove(6B)	51
Sent2Vec-1M	53
Sent2Vec-300K	50
WORD2VEC	43
Glove(840B)	50
Glove(42B)	51
FastText	51
SSWE	51
EMO2Vec	45
DeepMoji	53

TABLE VI: Dataset Description

Dataset	Labels	Train set	Test set
Amazon	5	3M	650K
DBPedia	14	560K	70K
Yahoo	10	1.4M	60K

a lot of attention among researchers as popular algorithms for different machine learning problems. We tested all the embeddings over the neural network as shown in Table V. The Sent2Vec-1M and DeepMoji algorithms surpassed all the other representations as the top performers in this problem.

Furthermore, to investigate the performance of all the embeddings outside of the Amazon dataset we chose different dataset as it is explained in the Table VI and trained a logistic regression classifier with the different embedding representations. The selected dataset have different label classes and different sizes. We trained the Sent2Vec model (Figure 2) over the training data per each dataset. Table VII illustrated the performance of the logistic regression over different dataset. We investigated if the number of label classes had any impact on performance of the Sent2Vec representation embedding. Our results shown that the Sent2Vec embedding representation had the top performance in the dataset selected from Table VI.

TABLE VII: Logistic Regression over different dataset

Representation	Accuracy
Dbpedia	
Glove(6B)	63
Sent2Vec	**77**
WORD2VEC	74
Glove(840B)	70
Glove(42B)	53
FastText	53
SSWE	25
EMO2Vec	49
DeepMoji	50
Yahoo	
Glove(6B)	46
Sent2Vec	**65**
WORD2VEC	48
Glove(840B)	47
Glove(42B)	44
FastText	52
SSWE	30
EMO2Vec	31
DeepMoji	41

VI. Conclusion

In this paper, we presented the Sent2Vec a neural network model that boosts the performance of sentimental analysis by generating a new sentence representation. We ran our experiment over a big dataset of Amazon reviews with 3 million records for training and 650,000 records for testing. Our results indicate that the traditional embedding representation are not sufficient in specific domain when there is a latent sentiment inside the sentence. Sent2Vec magnified the sentiment representation of a sentence and produces a vector of numbers between 0 and 1 that replicated the original embedding characteristics yet with a taste of the sentiment of the sentence. Furthermore, our observation implies that the new embedding increases the performance of NLP algorithms for downstream tasks such as sentimental analysis, classification and sentence clustering. Our future plan is to train Sent2Vec models with different type of dataset and apply transfer learning across different corpus. Furthermore, in another planned comprehensive study, we will work toward another set of experiments to observe the behaviour of the network while the vocabulary of texts are different but sentimental contexts remain fixed.

Acknowledgment

The work reported in this paper was supported in part by National Science Foundation under Grant No. CNS-1526055.

References

[1] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523, Jan 1988.

[4] Language modeling for information retrieval. 2003.

[5] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[6] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models, 2015.

[7] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, Sep 1990.

[8] David M. Mimno, Matthew D. Hoffman, and David M. Blei. Sparse stochastic inference for latent dirichlet allocation. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008.

[10] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation, 2012.

[11] Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews, 2014.

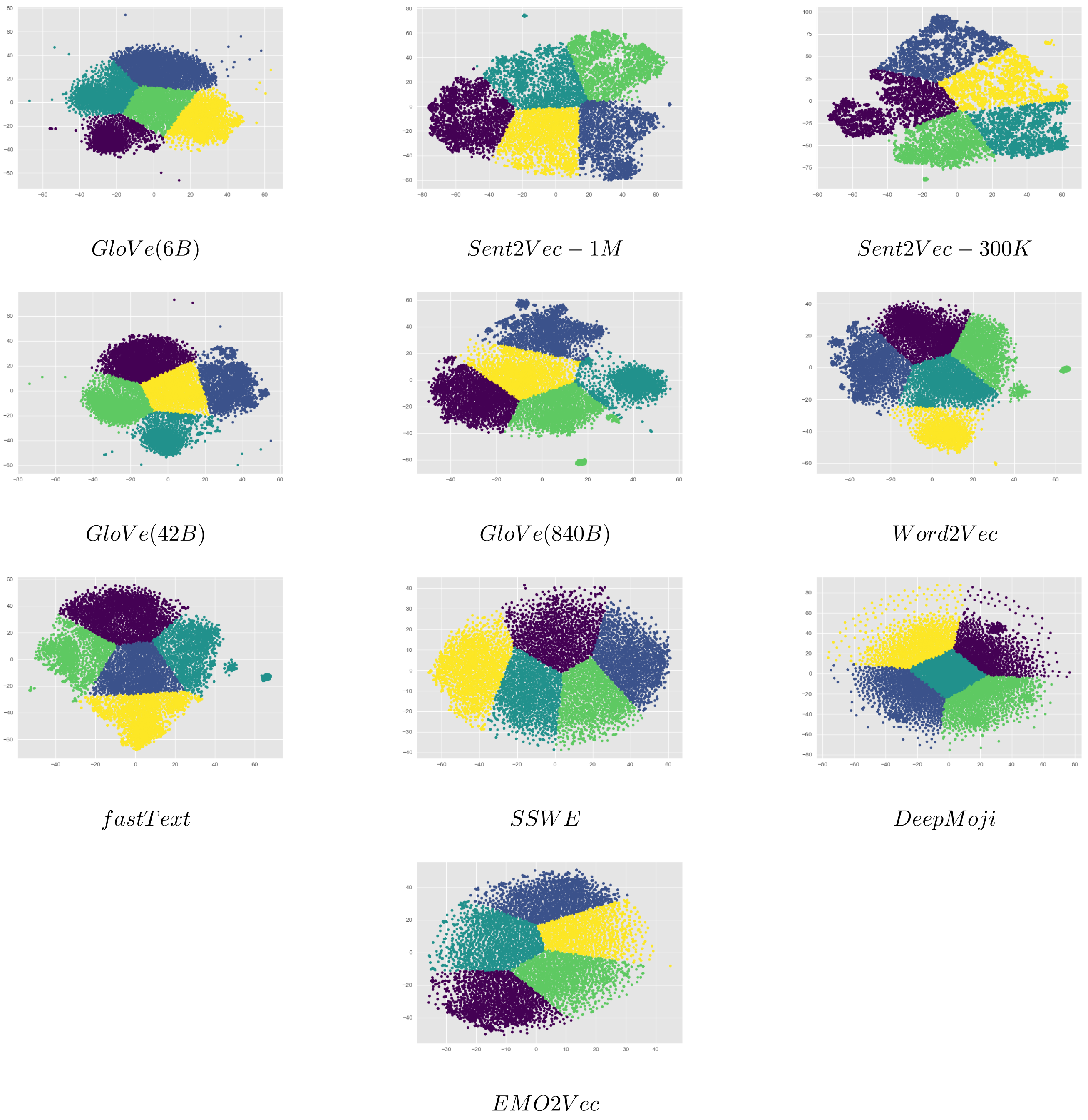


Fig. 6: Cluster Representation for each Sentence Embedding

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[13] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors, 2015.

[14] 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.

[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[16] Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 534–539, 2017.

[17] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*, 2017.

[18] Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. Emo2vec: Learning generalized emotion representation by multi-task training. *arXiv preprint arXiv:1809.04505*, 2018.

[19] Igor Labutov and Hod Lipson. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 489–493, 2013.

[20] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691, 2015.

[21] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks, 2015.

[22] Yashen Wang, He-Yan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. Cse: Conceptual sentence embeddings based on attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 505–515, 2016.

[23] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[24] Kawin Ethayarajh. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 91–100, 2018.

[25] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[26] Xiang Zhang and Yann LeCun. Text understanding from scratch, 2015.

[27] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[28] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[29] Yutaka Sasaki et al. The truth of the f-measure. 2007, 2007.

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