Sentimental Semantic Classification using Decomposed LSTM over Big Data

by

Mahdi Naser Moghadasi, M.Sc

A Dissertation

In

Computer Science

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

Approved

Yu Zhuang

Victor Sheng

Shuo Yu

Mark Sheridan
Dean of the Graduate School

December, 2020

TABLE OF CONTENTS

ABSTRACT

Text classification is considered as one of the primary task in many Natural Language Processing (NLP) applications. Traditional text classifiers often depends on human interaction for feature design, such as building dictionaries or knowledge databases. The range of text classification research goes from designing the best features to choosing the best possible machine learning classifiers. In industry fields, such as marketing and product management, they are already leveraging the process of text analyzing and extracting information from textual data. For example text classification of content on the website using tags helps Google crawl the website easily or a faster emergency response system can be made by classifying panic conversation on social media. In text classification a challenge is the feature engineering over the input to extract characteristics and find their associations to the output classes. With the evolve of deep neural network, the task of feature extraction is mostly taken care with dense layers of neural network where each layer extracts a representation of the input data.

However, success of deep learning is associated to our ability to train large neural networks for large datasets. Long training times for deep neural networks (DNNs) affect research in new DNN by slowing the development of DNNs.Hence, faster training enables increasingly larger models to be trained on large datasets in feasible amounts of time. Another challenge in text classification problems for sentimental analysis tasks is traditional feature engineering of data presented as embedding representation are not sufficient in specific domain when there is a latent sentiment inside the sentence.

To solved these challenges, we propose LSTM Decomposing Method (LDM) to reduce training time of DNN(s) and present Sent2Vec, a new representation of input text that includes the sentiment. LSTM Decomposing Method (LDM) is based on disintegrating the internal unit of a Recurrent Neural Network (RNN) into sub-units. Unlike previous similar researches, our approach does not need an active re-training during the parameter reduction phase. Additionally, we introduce Sent2Vec, a sentimental representation for text classification. Sent2Vec magnified the sentiment representation of a sentence and produces a vector of numbers where positions replicated the original embedding characteristics yet with a taste of the sentiment of the sentence. In this report, we utilized our techniques over multiple dataset including Amazon dataset which has 3,000,000 records for training and 650,000 for testing. We also explain two use cases of NLP techniques in chatbot implementation and Facebook comment analysis.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

In many industries the advent of Big Data has resulted in a need to process and analyze vast amounts of data in a fast, efficient and scalable manner to extract business insights and improve customer experience. Sentiment analysis is a computational approach toward identifying opinion, sentiment, and subjectivity in text. Sentiment classification methods classify a document associated with an opinion to be positive or negative. 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. As part of a pre-processing task in NLP, the text needs to be transformed into a format that is understandable by computer. 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. 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 [56, 106, 114]. Arora et.al. [8] 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. [36] 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. A challenge in the text data representation is the generalization of the word embedding methods. 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.

Deep learning has emerged as a powerful machine learning tool for data representation and prediction. Mimicking the structure of the brain, neural networks consist of a number of so-called neurons organized in a certain way for processing information. It learns to execute tasks by changing the connection between neurons through an iterative train-

ing process. Deep learning approaches in industry frequently have the need to process very large data sets. Two factors drives the growth of neural network sizes. One is the desire to achieve good accuracies for predictions, making people to turn to larger network models with large number of internal parameters to be learned. The second is the deluge of large datasets, which could takes days of high performance computing power to train their machine learning models. Although a number of large-scale distributed machine learning frameworks have adapted algorithms to distributed and scale up architectures as mentioned in [97] to process the data efficiently, there are examples like the BERT and ResNet-50 models, respectively taking 3 days on 16 TPUv3 chips and 29 hours on 8 Tesla P100 GPUs [33], [48] to train. As mentioned the training of large models is often both computation and memory intensive. Deploying such heavy size models in data centers incurs high power consumption, leading to high utility charges from data centers.
As deep neural networks grow in popularity for a wide range of problems across many scientific and engineering domains, they also grow in internal structure complexity. One class of DNNs is Recurrent Neural Network (RNN), which is a class of neural networks whose connections between neurons form a directed cycle. The application of RNN over sequential data makes RNN a suitable algorithm for tasks such as text mining, time series analysis etc. However, the number of parameters or weights in sophisticated networks makes them expensive to be trained, and even infeasible to be deployed on devices with limited resources such as memory, storage capacity, network bandwidth or power [44]. For example, the deep neural network used for acoustic modeling in [46] had 11 million parameters which grew to approximately 67 million for bidirectional RNNs and further to 116 million for the latest forward only GRU models in [5].

CHAPTER II
MOTIVATION

The success of deep learning is associated to our ability to train large neural networks for large datasets. Long training times for deep neural networks (DNNs) affect research in new DNN by slowing the development of DNNs. Faster training enables increasingly larger models to be trained on for large datasets in feasible amounts of time. Furthermore, the traditional word embedding methods often model the syntactic context of words but ignore the sentiment information of text. This can impact on the accuracy of a classification model to predict the correct sentimental score for a text. These problems can be broken down in the following smaller items:

- **Problem 1. The training time for models on the large scale data is long.**
  This is a problem that often unrealized till the testing stage if the choosing of the algorithm is solely made by studying previous research works on a smaller data set. For example, research papers [6], [7],[47] use *Reuters-21578 Text Categorization Collection Data Set* [1] as a benchmark for evaluation of their classification algorithms. Reuters-21578 has a total of 21578 instances. Although Reuters-21578 has common features to other larger data set but in our opinion; it has a limitation to explore other characterizes of text classification algorithms specially when the objective is to compare a proposed algorithm on an industrial size problem which is bigger in term of number of instances.

- **Problem 2. Traditional word embedding partially includes sentimental characteristics of the sentence.**
  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. This can impact on the accuracy of a text classification model to predict the correct sentimental score for a text.

[1]https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection

Innovations

Regarding the specified problems this research makes the following contributions:

- LSTM Decomposing Method (LDM).
  To address the first problem, we introduce a method to decompose the processing unit of LSTM neural network to reduce the training time, and experimental study of the implemented decomposed LSTM method on a large document classification problem will be evaluated. The decomposed LSTM method has fewer trainable parameters than the original method, and we believe this might impact on reducing training time. The closest to our techniques are Neural Architecture Search (NAS) and Pruning algorithms - we will explain them in section III - however, both techniques required training in-advance to find the optimal topology for a neural network. In the simplest form of the LDM, the neural network LSTM layer decomposed to sub-layers before training. Furthermore, the neural network is decomposed in such a way that subunits of the decomposed LSTM allow the use of pre-trained LSTM unit to start as the subunits. Leading to further reduction of training time for the decomposed LSTM network.

- **Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic**
  To discuss new feature embedding, we present Sent2Vec, an alternative embedding representation that includes the sentimental semantic of a sentence in its embedding vector. We utilized unsupervised Smoothed Inverse Frequency (uSIF) sentence embedding methods in the Sent2Vec neural network over a multi million samples dataset. We explain sentence embedding in section IV. 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 a large Amazon dataset when combined with GloVe(6B).

CHAPTER III
RELATED WORKS

As our research work divided into two major subjects we discuss literature review into two separate sections. In first section the previous works related to optimization of neural network to improve the training time over large dataset will be discussed, next we follow by explanation of different representation methods for words and sentences. There have been numerous studies done on how to accelerate the training and testing process of a model, and Neural Architecture Search and model Pruning are two of the well-studied approaches, and we will investigate in this research for their effectiveness.

Neural Network Optimization

Neural network optimization techniques in general refer to approaches to reduce the training or testing time. Most of these approaches focus on the trainable parameters in a network or the network structure. For example, in language modeling the size of the parameters in the recurrent layers have increased dramatically even with various techniques explored for optimization in [60] [21].

Neural Architecture Search (NAS)

Identifying good neural network architectures took years of research by experts through examining different congurations. Recently, there has been an increase of trends in using algorithms to automate the manual process of architecture design. . Representative architecture search algorithms can be categorized as evolution [116, 74, 93, 105], weights prediction [14] , Monte Carlo Tree Search [84] and reinforcement learning [123, 124, 121] where among which reinforcement learning algorithms have stood out with strongest performance. Despite the progress in NAS algorithms a major challenge of intensive computation still exists as each evaluation typically requires re-training the whole neural network.

Pruning

Model pruning requires scaling down the number of parameters or weights of a neural network, ultimately leading to lower memory and computation costs but with the least possible loss of the predictive power [44]. By eliminating connection weights with small values from a trained neural network, pruning algorithms can produce sparser networks that keep a reduced number of the connections but maintain similar performance compared to the original network. By retaining the original weight initialization values, these sparse networks can even be trained from scratch to achieve a higher test accuracy [39, 67] than the original network. In model pruning mechanisms they either individually remove less important parameters (with small values) [44, 45] or structurally remove a group of such parameters [115, 38, 4, 49, 50]. A drawback of pruning is to require prior training of the full network to obtain useful information about each weight in advance, hence increasing end-to-end training time as a result. In this research we introduce an approach of decomposition to disintegrate an LSTM unit into sub-units without going through a searching process for the network architecture. Our work is different from traditional NAS and pruning approaches mainly in not relying on in-advance training of the selection of network architecture and parameters, and we achieve this through LSTM unit decomposition. Furthermore, we have investigated and found that small LSTMs trained in earlier studies can be used as pre-trained sub-units for the decomposed LSTM. Although the pre-train unit are not a necessary step in order to build the neural network model, the use of pre-trained units leads to further reduction of training time.

Word Emebdding

The text classification problem in computer science is a challenging problem by converting the text in a format that is understandable by the computer in a sort of numerical format. Many machine learning algorithms use a fixed-length vector that represented the features. Bag-of-words or BOW is among most common representations in the text domain. Although its popularity among researchers because of its simplicity it has some drawbacks. The BOW representation lose the ordering of the words which it entails to ignoring the semantic of the context in the text. Recent research on deep learning to learn a distribution representation vector over words has achieve more popularity in NLP. Word2Vec [78] is a neural network model to learn to the center word by its surrounding (context) words in a statement, It gives words of similar meanings to close points over continuous vector space. Although its simplicity the model has generated quality words in NLP tasks such as language modeling, text understanding and machine translation.

An extension to Word2vec is Paragraph Vectors [66] generalize the concept of Word2vec to learn vector representation for a set of documents. It is an unsupervised algorithm that learns fixed-length vector representation of variable length text in the corpus. Paragraph Vectors uses the similar model of word2vec with a document vector (such as a document id) that is unique for each document. It has surpassed traditional model such as BOWs, Latent Dirichlet Allocation[80] and text understanding tasks [28]. However there is not easy clarification on how to retrieve the unique document id using an inference process in the paper. Also, the number of parameters to be learned increased as the size of the training corpus and makes it computationally expensive at the testing for unseen data. It is observed that simple methods of algebra such as additional composition on the word embedding learned by Word2Vec can keep meanings of a phrase or a sentence, for example a popular example is shown that vec("king") - vec("man") + vec("women") is close to vec("queen") [75]. In Doc2vecC [19], it represents each document as a average of the word embedding vectors of all the words in the documents; this process is during the learning phase as opposite to other methods which post process the learning word embedding [103], [73]. To speed of the algorithm, Doc2VecC removes words from document randomly.
The methods discussed so far are belonging to group of word embedding called context-free representation. These methods only provide a single global representation for each word, ignoring their context. In more recent methods, the representation for each word depends on the entire context in which it is used. This new representation called contextual representation as the embedding of a word is associated to the context it is applied. Perhaps a major contribution to contextual representation is the concept of paying Attention [109] to each word according to their meaning in a sentence. The idea behind attention in neural network is to use parts of the input sentence where most relevant information is concentrated for a word. **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) [32] is a contextual representation methods that implemented the attention mechanism through a stack of encoder/decoder layers. BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific ar-

chitecture modifications. Different variations of BERT also proposed **R**obustly **O**ptimized **BERT** **A**pproach (RoBERTa) that trained over 160 GB of data which is significantly more the training data size used for the original BERT. **A** **L**ite **BERT** (ALBERT) architecture that incorporates two parameter-reduction techniques: factorized embedding parameterization and cross-layer parameter sharing to reduce the size of the model. Beside BERT and its variation methods, **E**mbeddings from **L**anguage **MO**dels (ELMO) is a character based contextual embedding method that applies several layers of bidirectional LSTMs to capture context between words. ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training. ELMo comes up with the contextualized embedding through grouping together the hidden states (and initial embedding) in a certain way (concatenation followed by weighted summation). Although its several advantages, yet it is not as popular as BERT for embedding representation.
Irrespective of contextual and context-free approaches, the generalization of word embedding representation often fails to capture the emotional semantic between words in a sentence. To overcome this challenge and include effective information into the word representation, Tang et al [107] proposed Sentiment-Specific Word Embeddings (SSWE) which encodes both positive/negative sentiment and syntactic contextual information in a vector space. His 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. Felbo et al. [37] 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 [117] is a word-level representation that encodes emotional semantics into fixed-sized, real-valued vectors. Labutov and Lipson [64] 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. Other popular representations range from the language model based methods ([65]; [79]; [62]), topic models ([31];[80]), Denoising Autoencoders and its variants ([112]; [20]), and distributed vector representations ([72]; [77]; [63]). Another prominent line of work includes learning task-specific

document representation with deep neural networks, such as CNN ([120]) or LSTM based approaches [106],[27].
Most of discussed methods applied one sort of deep learning algorithms to create a new representation. However, there some challenges in running complex deep learning algorithms. A major challenge is, size of some of models is too large for an experiment without high computing infrastructure. For examples the BERT model took 3 days on 16 TPUv3 chips [33] to train. Considering this limitation, we focused on pre-trained models that are available to be executed over a desktop computer to calculate embeddings of large dataset such as Amazon reviews [120]. Furthermore, the scope of this research is to improve the quality of a traditional word embedding (Glove) representation and build a sentence embedding model that includes sentimental latent patterns in the new representation. We aim to present different comprehensive evaluations of our method to explain that although in this experiment our method Sent2Vec utilizes a form of Glove representation - a lesser complex model compared to BERT - to generate the sentimental embedding, yet it can also be applied over more complicated models to generate a sentence embedding. Additionally, our method is different from Emo2Vec, SSWE as the outcome of Sent2Vec is a sentence representation applied over multi-class labels however these techniques are restricted for either binary class problems or word embedding representation. Finally, in our evaluation we compared our method to pre-trained models that are available under open-source license, we noticed that fast sentence embedding module [12] implemented many of those pre-trained models to be accessible for public. We explain the data representation methods in more detail in the following section.

CHAPTER IV
FEATURE REPRESENTATION

In this section we discuss over different embedding methods to represent a word in a format of a vector of numbers. The dimension of the vector varies according to different representation methods. The traditional word representation discuss firstly and as we go further more sophisticated algorithms will be explained. Finally, we will explain sentence embedding algorithms that run over the word embedding representation to produce a single vector corespondent to a representation of a sentence.

Bag of words (BOW)

Bag of Words (BOW) is known arguably a very common document representation according to its simplicity and an easy implementation. BOW uses frequency of occurrence of each word as features that passed as input to a classifier label documents. Although its popularity it fails often to find similarity between words and phrases due to sparsity nature of the generated vector in a high dimensional representation. The major weaknesses of the BOW are: Firstly it loses the ordering between words in a sentence and secondly due to the first issue introduced the semantics between words are ignored.

TF-IDF

In order to make the text to be understand by computers some sort of representation in the format of numbers proposed in the format of vector. The process of vectorization or more specifically document vectorization is to convert text content to a numeric vector that applied as features. Machine learning algorithm use these features to build a model. Term frequency - inverse document frequency (TF-IDF) is a numerical statistic representation that aims to reflect importance of a word in a text document among all other document in the corpus [104]. TF-IDF stills consider to be a BOW model that loses word ordering.

$$\text{TF-IDF}_{t,d} = (1 + \log \text{TF}_{t,d}) \cdot \log \frac{N}{\text{DF}_t}$$

- TF = Term Frequency, which measures how frequently a term occurs in a document.
- IDF = Inverse Document Frequency, which measures how important a term is.

Word2Vec

Word embedding methods are a set of algorithms to represent a word in a format of a vector with fixed length. That is continuous vectors in a low dimension space that embedding the semantic and lexical features of a word. This continues vector representation of a word can be obtained from the internal hidden layer of the neural network models used to train.

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.

Given a predicted word vector $\hat{r}$ and a target word vector w_t. The probability of the target word conditional on the predicted word is calculated by a softmax function:

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

where W is the set of all target word vectors.

Notice that $\hat{r}$ is neither an element of W nor computed from elements of W. As we will see later, 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:

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

The gradient with respect to w of $\mathcal{L}(w_t, \hat{r})$ is:

$$g_1(w, w_t, W, \hat{r}) = \frac{\partial}{\partial w}\mathcal{L}(w_t, \hat{r}) = \hat{r}(P(w|\hat{r}) - I\{w = w_t\}) \tag{4.1}$$

where $I\{.\}$ is the indicator function.
The gradient with respect to $\hat{r}$ is:

$$g_2(w_t, W, \hat{r}) = \frac{\partial}{\partial \hat{r}}\mathcal{L}(w_t, \hat{r}) = \sum_{w \in W}[P(w|\hat{r})w] - w_t \tag{4.2}$$

Skip-gram

For an index i and a window size c, skip-gram predicts the context words $\{w_j\}$, $(i - c \leq j \leq i + c, j \neq i)$ given the centered word r_i. Hence, in the general model, $w_t = w_j$ and $\hat{r} = r_i$ for this case. Fig. 4.1. The cost function is derived as follows:

$$\mathcal{L}_{skipgram}(c, i) = \sum_{i-c \leq j \leq i+c, i \neq j} -\log P(w_j|r_i)$$

The gradients of this function are:

$$\frac{\partial}{\partial w}\mathcal{L}_{skipgram}(c, i) = r_i \sum_{i-c \leq j \leq i+c, i \neq j} g_1(w, w_j, W, r_i) \tag{4.3}$$

$$\frac{\partial}{\partial r_i}\mathcal{L}_{skipgram}(c, i) = \sum_{i-c \leq j \leq i+c, i \neq j} g_2(w_j, W, r_i) \tag{4.4}$$

Continuous bag-of-words

Intuitively, this model reverses the modeling mechanism of skip-gram. CBOW predicts a word given its context. The target word vector is now the output vector of the word at index i, w_i; the predicted word vector is the sum over all context input vectors: Fig. 4.2.

$$\hat{r} = \sum_{i-c \leq j \leq i+c, i \neq j} r_j$$

The CBOW's cost function is as follows:

$$\mathcal{L}_{CBOW}(c, i) = -\log P(w_i|\hat{r})$$

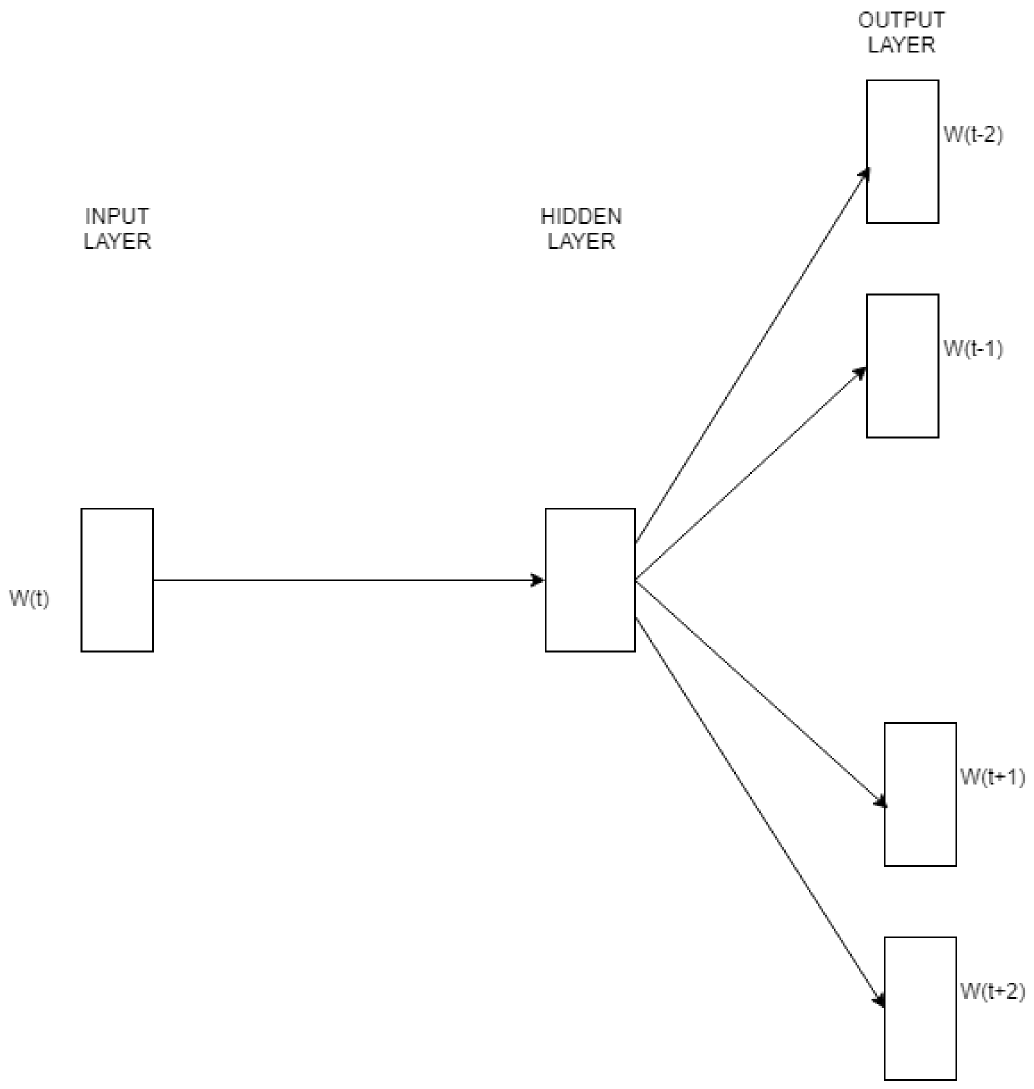


Figure 4.1. The architecture for the Skip-Gram model.

and the gradients are: student

$$\frac{\partial}{\partial w}\mathcal{L}_{CBOW}(c,i) = g_1(w, w_i, W, \hat{r}) \tag{4.5}$$

$$\frac{\partial}{\partial r_j}\mathcal{L}_{CBOW}(c,i) = g_2(w_i, W, \hat{r}) \tag{4.6}$$

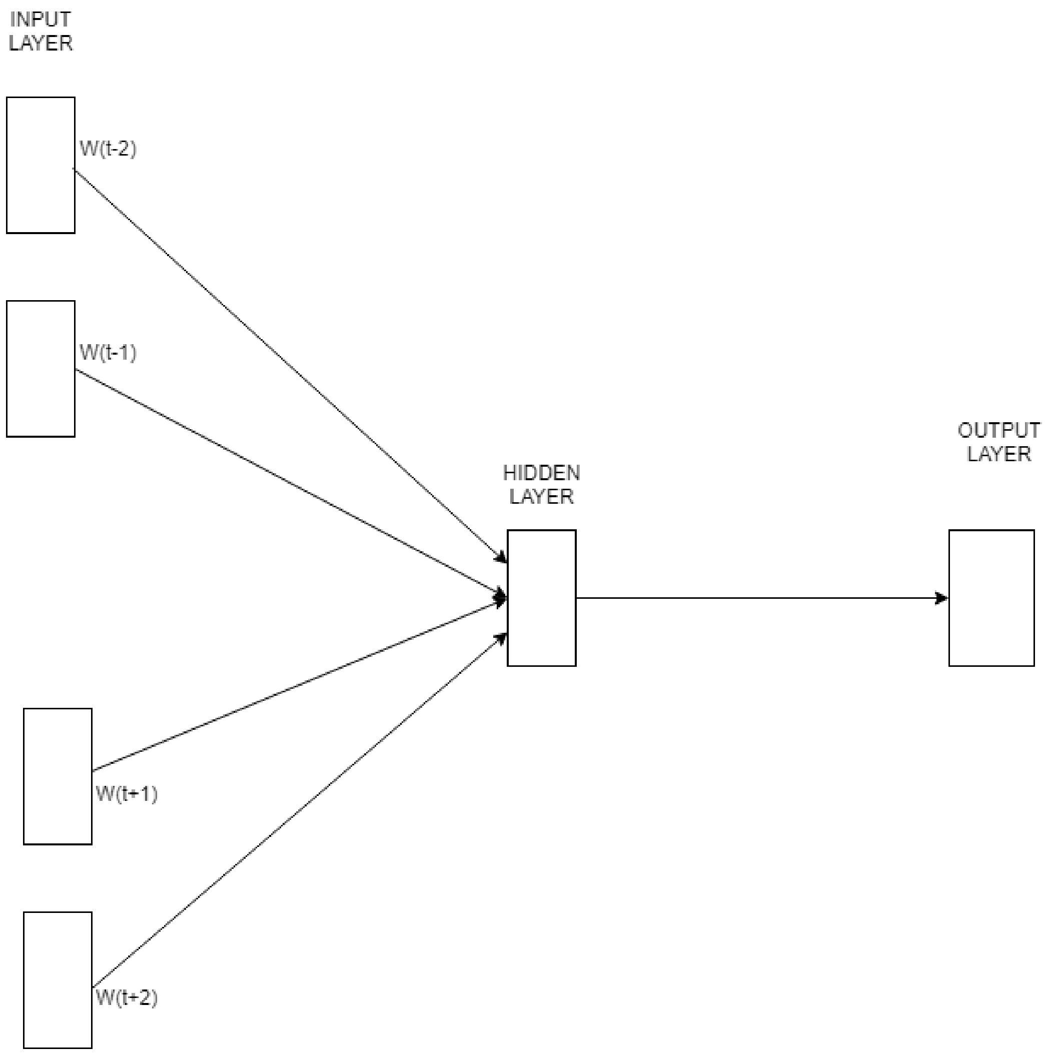


Figure 4.2. The architecture for the Continuous Bag Of Words.

for $i - c \leq j \leq i + c, i \neq j$. Otherwise,

$$\frac{\partial}{\partial r_j} \mathcal{L}_{CBOW}(c, i) = 0 \tag{4.7}$$

Negative sampling

Training a neural network takes a train data and adjusting the neuron weights so that it predicts that test sample more accurately. In other words, each training sample will tweak

all of the weights in the neural network.
The size of our word vocabulary means that our skip-gram neural network has a tremendous number of weights, all of which would be updated slightly by every one of our billions of training samples.
Negative sampling addresses this by having each training sample only modify a small percentage of the weights, rather than all of them. With negative sampling, we are instead going to randomly select just a small number of *negative* words to update the weights for.

GloVe

GloVe is a more recent word embedding model developed by [87]. In GloVe , the model was compared directly to *word2vec*. There was a debate about how that experiment was set up. The final version of the paper still claimed that *GloVe* outperformed *word2vec* on the task of word analogy [77] if we let it run long enough. The authors illustrated how the training objective of *skip-gram* uses information from the occurrence matrix of a corpus, which explains why the model, although only uses local context window, can produce word embeddings that capture global relationships. Hence, the *skip-gram* model can also be viewed as a least square problem where only simple matrix decomposition is required to compute the solution. Thus *GloVe* yields a better performance than *skip-gram*.
Concretely, let X be the occurrence matrix of the corpus where the (i, j) entry is number of times word i and word j co-occur. Denote by $P_{ij} = \frac{X_{ik}}{X_i}$ the probability that word i and word j co-occur. 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:

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{P_{ik}}{P_{jk}} \tag{4.8}$$

Since in the occurrence matrix, the roles of the rows and columns are interchangeable, we enforce symmetric to the above equation:

$$F\left((w_i - w_j)^T w_k^{'}\right) = \frac{F(w_i^T w_k^{'})}{F(w_j^T w_k^{'})} \tag{4.9}$$

From Eqn. 4.8 and Eqn. 4.9, we have:

$$F(w_i^T w_k^{'}) = P_{ik} \tag{4.10}$$

or

$$w_i^T w_k^{'} = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{4.11}$$

From Eqn. 4.11, we set the following weighed least square objective:

$$J = \sum_{i,j} f(X_{ij})(w_i^T w_j^{'} + b_i + b_j^{'} - \log X_{ij}) \tag{4.12}$$

where $f(.)$ is a weight function and $b, b^{'}$ are bias terms.
Let's recall the training objective of *skip-gram*:

$$J = -\sum_i \sum_{j \in context(i)} \log Q_{ij} \tag{4.13}$$

where $Q_{ij} = \frac{\exp(w_i^T w_j^{'})}{\sum_k exp(w_i^T w_k^{'})}$, which can be rewritten as:

$$J = -\sum_{i,j} X_{i,j} \log Q_{i,j} = \sum_i X_i H(P_i, Q_i) \tag{4.14}$$

where H is the cross entropy between two distributions.
The *GloVe* model is more advanced than Eqn. 4.14 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.

fastText

Joulin et al. [58] introduced an embedding based on the N-gram features. fastText architecture is a simple linear model with rank constraint, Fig. 4.3. 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

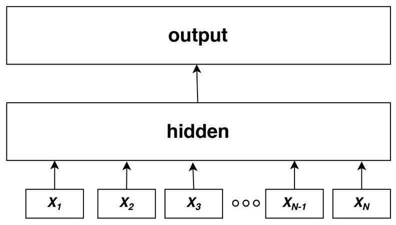


Figure 4.3. Model architecture of fastText for a sentence with N ngram features $x_1, ..., x_N$.

log likelihood over the classes:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log\left(f\left(BAx_n\right)\right) \tag{4.15}$$

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 $n_1, \ldots, n_l$, the probability is

$$P\left(n_{l+1}\right) = \prod_{i=1}^{l} P\left(n_i\right) \tag{4.16}$$

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

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.

Paragraph Vector

Paragraph Vectors [66] is a unsupervised learning model that every paragraph is mapped to a unique vector. This could be consider as an extension to a word2vec model that a new variable added to map paragraph as it is shown in Fig. 4.4. In a classic implementation the paragraph vector and word vectors are averaged to predict the next word in a context.In [76] unweighted averaging is found to do well in short length phrases.

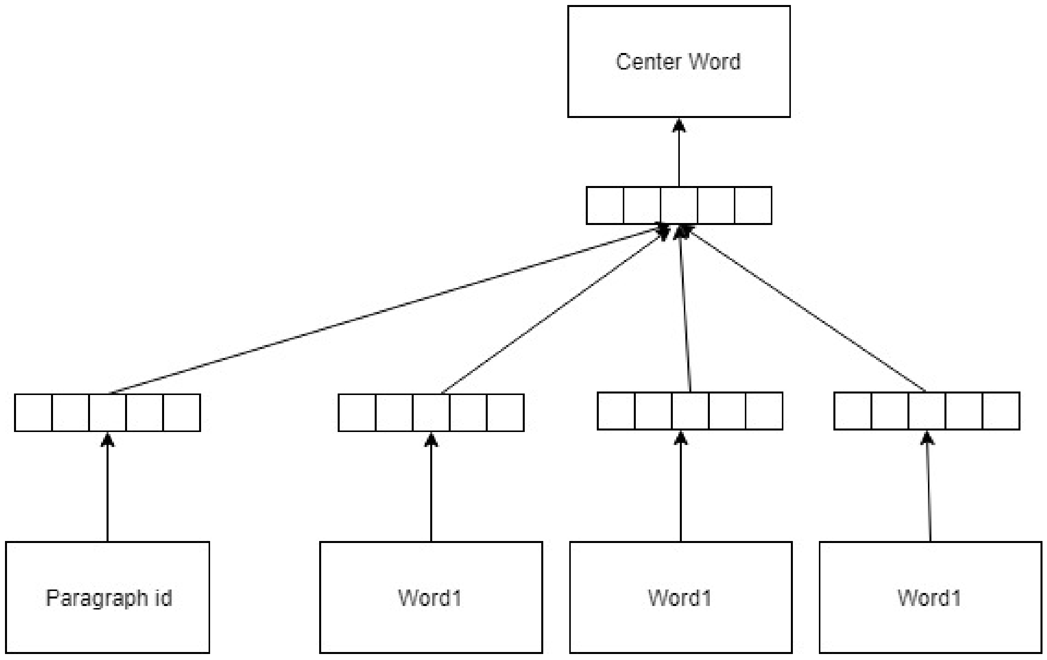


Figure 4.4. A framework for learning word vectors. Context of Word1, Word2 and Word2 are used to predict the Center Word.

The algorithm has two key stages:

- A training phased to get words and paragraph vectors.
- An *inference stage* to get a paragraph vectors for unseen document.

Although the first step is well understandable from the paper but the inference stage is not very explained in [66].

Smooth Inverse Frequency (SIF)

The main idea behind SIF [8] 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 IV, 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 IV) is removed to build the new updated v_s as the sentence embedding for sentence s.

[H] 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. Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

sentence s in $\mathcal{S}$ $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$ Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector

sentence s in $\mathcal{S}$ $\quad v_s \leftarrow v_s - uu^\top v_s$ SIF - Sentence Embedding 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 [8].

unsupervised Smoothed Inverse Frequency (uSIF)

uSIF builds upon the random walk model proposed in [9] 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 of word vector length on the probability of a sentence being generated in SIF [8]. 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 [36],[9] and [8] for more mathematical proofs of SIF and uSIF methods.

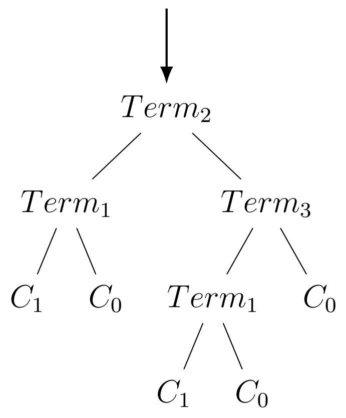
CHAPTER V
TEXT CLASSIFICATION ALGORITHMS

The problem of classification has been widely studied in information retrieval literature. The idea of classification is to use a set of training records T = $\{T_1,...,T_N\}$, such that each record is labeled with a class label from a set of C different discrete *Classes* = $\{C_{1..k}\}$. The training data is used in order to construct a classification model, which relates the features in the underlying record to one of the class labels. For a given test instance for which the class is unknown, the training model is used to predict a class label for this instance. The problem of text classification discussed in different domains such as : Document Organization and Retrieval [18], News filtering and Organization[95], Opinion Mining[119] and Email Classification and Spam Filtering [24],[26], [101]. One of the challenges in text classification is text produces high dimensional sparse feature input because of low frequencies on most of the words. It is important in the text classification to consider such sparsity characteristics.
Feature selection is an important problem for text classification. In feature selection, The objective is to apply the features which are most correlated to a class during the classification process. In the other words, some of the words in the text are has a higher likelihood to be correlated to the class distribution than others.

Decision Tree

A decision tree in simple word is a graph that uses a dividing method to illustrate every possible outcome of a decision.In the text data, a condition on attribute to divide the data space are often based on the presence and absence if a term in the document occurs or not. A leaf node is represent corresponding class labels. In the other words, Decision Tree recursively partitions the training data set into smaller subset based on a set of predicates at each node or branch [35][88]. Other type of predicates or conditions for partitioning are possible. Single Attribute Splits, Similarity-based multi-attribute split and Discriminant-based multi-attribute split are different partitioning approaches. [98] used decision tree in combination with boosting techniques to improve the accuracy of the classification as well.
As it is shown in the figure below each class is abbreviated with C_n and depending on the

condition of absence or occurring of a *Term*$_t$ the input document categorized to one the classes of the set $\{C_0,C_1\}$.



Naive Bayes Classifiers

The Naive Bayes classifier models the distribution of the documents in each class using a probabilistic model. It is based on an independence assumptions about the distributions of different terms. Although this assumption is clearly false in many real world applications, Naive Bayes performs surprisingly well. Two type of models are commonly known to be studied in text in naive Bayes classification are *Multivariate Bernoulli Model* and *Multinomial Model*. These models make assumption about how the data are generated and propose a probabilistic model based on these assumptions. Bayes rule in (5.1) is used to classify test data and picks the class that is most likely has generated the example as mentioned in [70].

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{5.1}$$

Both models essentially calculate the posterior probability of a class according to distribution of the words in the document. These models are also has a common weakness of BOW feature that ignores the actual position of the words. A major difference between Multivariate Bernoulli Model and Multinomial Model these models is considering to take word frequencies into account and sampling the probability space. Both methods suffer on assumption that input features have multinomial distribution which is a generalization of the binomial distribution shown in the Fig. 5.1. Neither binomial nor multinomial distributions can contain negative values. This can be a major draw back later once we need

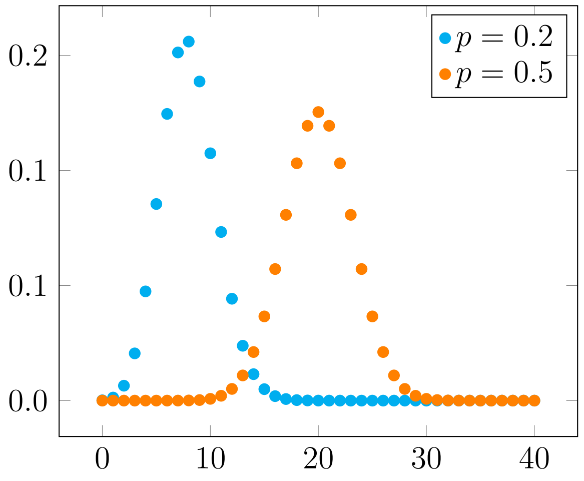


Figure 5.1. Probability mass function for Binomial distribution over 40 samples.

to pass a representation of a document in vector format that included negative dimensions. (For example average of word2vec representations of all words in a document may have negative values)

Support Vector Machine

Support Vector Machine (SVM) classifier determine separators in the search space which entails separating the different classes. SVM in the context of text documents are models that classifies document based on the value of the linear combinations of the documents features [108]. A single SVM is a binary classifier [53] such as positive and negative classes.

The algorithm attempts to find a hyperplane with the maximum *margin* from the positive and negative examples. The document with margin of the hyperplane are called *support vector*, shown in Fig. 5.2. For a training data set of n points of the form $(\vec{x}_1, y_1), ...(\vec{x}_n, y_n)$ a hyperplane can be written as the set of points $\vec{x}$ satisfying the equation:

$$\vec{w}.\vec{x} - b = 0$$

.

- $\vec{w}$ is the normal vector to the hyperplane.
- y_i is the label for class *i*.

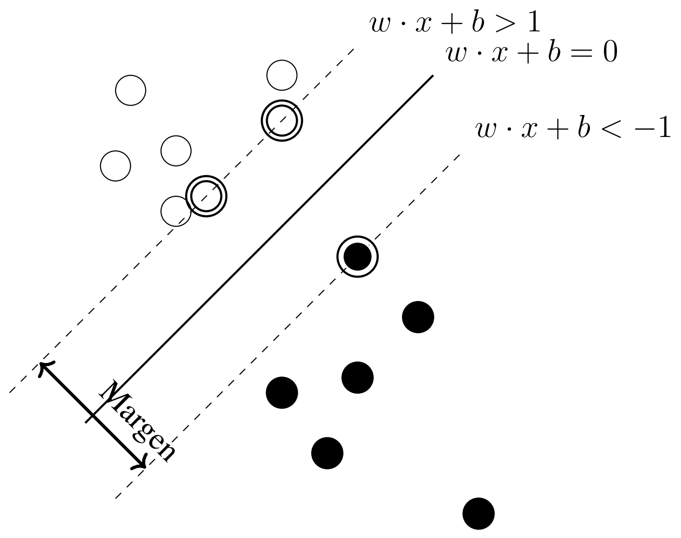


Figure 5.2. A linear SVM.

- $\vec{w} \cdot \vec{x} - b = 1$ anything on or above this boundary is of one class, with label 1.
- $\vec{w} \cdot \vec{x} - b = -1$ anything on or below this boundary is of the other class, with label -1.

The performance bounds on which the maximal margin classifier is based are independent of the dimension of the feature space [57], but instead depend on the margin. This is the reason that the SVM with high dimensional feature space can still give good generalization performance. However, this requires careful tuning of the regularization parameters. [83], [85].

Recurrent Neural Network

Neural network based methods have obtained great progress on a variety of natural language processing tasks. The primary role of the neural models is to represent the variable-length text as a fixed-length vector. These models generally consist of a projection layer that maps words, sub-word units or n-grams to vector representations (often trained before hand with unsupervised methods), and then combine them with the different architectures of neural networks.

A recurrent neural network (RNN) is able to process a sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector $h^{(t)}$ of the input sequence. The activation of the hidden state $h^{(t)}$ at time-step t is computed as a function f of the current input symbol $x^{(t)}$ and the previous hidden state $h^{(t-1)}$.It is common to use the state-to-state transition function f as the composition of an element-wise nonlinearity with an affine transformation of both $x^{(t)}$ and $h^{(t1)}$. However, There is a problem with RNNs with transition functions of this form is that during training, components of the gradient vector can grow or decay exponentially over long sequences.

Long short-term memory network (LSTM) was proposed by [51] to resolve the issue in learning long-term dependencies. Fig. 5.3 shows an example of LSTM. The LSTM maintains a separate memory cell $c^{(t)}$ internally that updates and exposes its content only when expected. At time step t, LSTM first decides what information to erase from the cell state. This decision is made by a σ function, called the forget gate. The function takes h_{t-1} output from the previous hidden layer and x_t (current input), and generates a number in [0, 1], where 1 means ”hold” and 0 means erase as it is shown in (5.2). Then LSTM decides what new information to store in the cell state. This includes two steps. First, a σ function, called the input gate as Equation (5.3) illustrates decides which values LSTM will update. Next, a tanh function computes a vector of new possible values $\widetilde{C_t}$,which will be added to the cell state in (5.4). LSTM merges these two to create an update to the state. To update the old cell state C_{t-1} into new cell state C_t as (5.5) presented. Finally, LSTM decides the output according to the cell state. In order to do that, LSTM first runs a σ layer which decides which parts of the cell state participate in the final output. This process occurs in so called output gate in (5.6). Then, LSTM puts the cell state through the tanh function and multiplies it by the output of the σ gate, so that LSTM only outputs the parts it decides to as (5.7).

$$f_t = \sigma(W^f x_t + W^f h_{t-1}) \tag{5.2}$$

$$i_t = \sigma(W^i x_t + W^i h_{t-1}) \tag{5.3}$$

$$\widetilde{C_t} = tanh(W^n x_t + W^n h_{t-1}) \tag{5.4}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{(}C_t) \tag{5.5}$$

$$o_t = \sigma(W^o X_t + W^o h_{t-1}) \tag{5.6}$$

$$h_t = o_t * tanh(C_t) \tag{5.7}$$

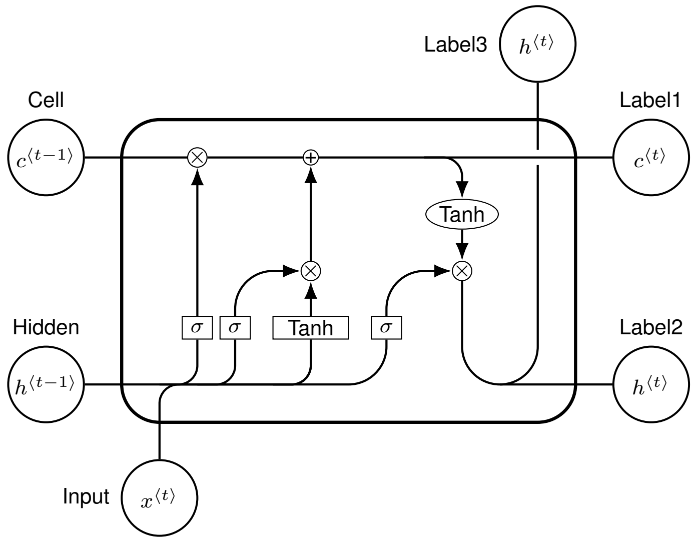


Figure 5.3. Long Short-Term Memory (LSTM) architecture

Recent research on LSTMs has focused in two directions: Finding variations of individual LSTM memory unit architecture [10, 22, 59, 43], and designing new ways of connecting LSTM layers into a network [23, 61, 122]. Although both approaches have improved performance over vanilla LSTMs still LSTM is hard to deploy due to its requirement on high computation complexity and large memory. Our target goal in this research is to address LSTM high complexity by decomposing an LSTM layer into sub-layers before training the network. We refer to words neural network and network interchangeably during this work.

CHAPTER VI
CHATBOT

In this section [1], we present one of the application of deep learning and text classification in developing chatbot. According to a report published by National Institute on Drug Abuse on 2017 (NIDA,2017) [34] the death toll of overdosing opioids is 115 in the US per day. Furthermore, in 2016 - the most devastating year of drug overdose epidemic - near 64,000 people died of drug overdoses and the estimation is over 2 million people suffer from disorders related to pain relievers for opioid in the United States. A school of research to address the opioid addicts has tended to focus on proposing a platform to identify either prescription drug abuse or finding opioid addicts via social media [11]. Relapse prediction if combined with mental support can be prevented or extended the duration that an addict stays clean. During this period addicts tend to be very emotional due to affects of the drugs in his/her brain. Hence, support of families, communities and medical professionals are as important as other medical treatments. However due to lack of knowledge or certainly of the best action to provide such a support from families and friends, social media has played as a platform to express addicts problems and confusions when he/she encounters one [99]. Addicts can post their questions or seek for answers in specific forums related to opioid subjects. For example on Reddit social media website, some discussion groups allow users to post their opinion or past experience regarding their opioid addiction and rehabilitation process and other users can either seek for an answer or educate themselves. The study [3] has done an analysis on various linguistic aspects of conversation and their correlation with conversation outcome. Their report shown quantifiable differences between a more successful and less successful counselors in how the conduct a conversation.

The evolution of big data opens a new door of the possibility to create a conversational agent empowered by data-driven methods. The system builds upon the idea of collecting a large number of human-human conversations on the social forums and use machine learning and information retrieval techniques to build agents that mimic humans in a mutual conversation between AI and human. Advances in machine learning, particularly in neural networks, has allowed for more complex dialogue management methods and more

[1]This section has accepted in ICNLP 2020 conference

conversational flexibility [29, 68, 110, 111, 40, 69].
Mental health applications that are in the market use of conversational agents and text-based dialogue systems. Often these agents can be deployed on messenger systems (eg, Facebook, Twitter) and are designed to present mental health materials in a more interactive and mutual conversational style. In the health field, such agents could be beneficial in helping users by providing relevant information about a disease or the clinicians to identify symptoms of a disease from a conversation[81]. Koko [82] - a chatbot for mental health intervention - provides information and education on responsible alcohol use. SHIHbot [13] is another application deployed on Facebook, which answers a wide variety of sexual health questions on HIV/AIDS.
At the time of this research, despite the variation of chatbots available for mental health patients, there is no chatbot available for opioid patients to inquire their questions in privacy using deep pretrained neural networks. We present Robo - a chatbot based on pretrained deep averaging network (DAN) to address such demands instantly and accurately. Our experiment demonstrated that our chatbot retrieves answers semantically by understanding the user's query first and matches to the best answer that has highest semantic score.
The paper is structured as follows: we define the methodology and data collection process in section VI. Technical detail and implementation of the Robo are explained in sections VI and VI, respectively. We illustrate real use cases where Robo retrieves responses from Reddit social media forums in section VI.

Methodology

Natural Language Understanding is a branch of NLP to convert human text into a format that is understandable for the computers. Considering this objective many techniques have been presented in the computer science communities on the concept of encoding a word to a vector of numbers. Consequently geometric operations can apply over these encoded numeric vectors to measure similarities between words. The same concept extends to a larger scale corpus such as a sentence, paragraph or a document to be encoded as a numeric vector respectively. Therefore, A geometric mathematical function such as cosine function can potentially being used to return similarity score between two documents if they encoded into numeric vectors.

In this section we explain the web resource to collect the data and store it in the chatbot knowledge service component. Next, we elaborate the chatbot model which is based on Query Semantic Understating (*QSU*). QSU is the machine learning component of the chatbot application that responsible to return the most relevant answer to a user query. The QSU encodes the user query into numeric vectors and use cosine similarity score to measure semantic distance between the user query and potential responses through two filtering processes discussed in the next section.

Dataset Collection

At the first step, We noticed the essence of having a collection of QA dataset to answer user's query. For this propose we chose Reddit [30] social media forum as our resource to assemble the data. Using the Reddit crawler module, we collected data from over 14 weeks via a daily automatic process to pull the posts, comments and their replies from subreddits topics such as drugs and opioid and created a questions/answers repository in the MySQL database as our *Knowledge Service* component. The format of the data in MySQL is a dictionary of QA pairs where each question can have multiple answers. We refer to these pairs as the *QA* or simply *dataset* thoroughly this paper.
An analysis over the collected data is shown in Table 6.1. According to Table 6.1 - (a) the size of dataset is 20,494 including questions and answers where out of 20,494 there are 7,596 questions and 12,898 answers replied to the questions. Furthermore, we extend our investigation on QA dataset and acknowledged for each question in the dataset there is an average 1.7 number of answer. While 1.7 replies for a question was an average number, We noted for some questions the reply's number is above 150 answers. Part (b) of Table 6.1 shows a statistical report over the questions and answers length in characters unit. We observed that on average each question in the dataset has over 100 characters long while for the answers this number is above 600 characters. That indicates users replied with a longer comments rather than a short response. The length of a sentence can impact on the encoded representation of it. The analysis of dataset in this section helped us to have a better understanding on what model to use to encode the dataset into numeric vectors. We explain about the process of encoding in the sentence encoder in the following chapter.

Table 6.1. Reddit Dataset Analysis

Total	*Question*	*Answer*
20494	7596	12898

Table 6.2. Dataset size of Questions and Answers

Percentile	*Question*	*Answer*
25%	48	242
50%	112	613
75%	250	1494
95%	718	5421

Sentence Encoder

Sentence encoder is a component of QSU to convert QA dataset and user's query which, are often larger than a single word, into numeric vectors. Google introduced Universal Sentence Encoder [16] in which two different encoders were implemented. First model is the Transformer based encoder [109] which aims for high-accuracy but has larger complexity and uses more computational resources. The second model uses a deep averaging network (DAN) [55] where embeddings for words and bi-grams are averaged together and then used as input to a deep neural network that computes the sentence embeddings. Likewise to the Transformer encoder, the DAN encoder takes as an input a lowercased string and produces a a 512 dimensional sentence embedding vectors. DAN utilize the efficiency with slightly reduced accuracy. As shows in Fig. 7.5, DAN constructed on several layers of deep feed-forward neural networks. The objective of stack of layers is that each layer learns a more abstract representation of the input than the previous one. We applied the TensorFlowHub python library to load the pretrained model[2] of a deep averaging network (DAN) architecture to encode textual format to numeric vector representations in the sentence encoder.

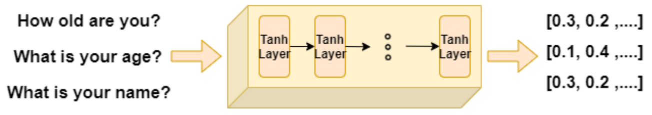


Figure 6.1. Deep Averaging Network (DAN) architecture

[2]https://tfhub.dev/google/universal-sentence-encoder/3

Chatbot Model

We created the chatbot model based on the single turn response matching [113]. In the single turn response matching the context and the candidate response are encoded as vectors respectively, and then the matching score is computed based on those two vectors. We introduce Query Semantic Understating (*QSU*) as the machine learning component in the chatbot model that implemented the single turn response matching logic. The QSU retrieval engine is defined as follows: given a set of QA pairs P=$\{q_i, a_i\}_{i=1}^{n}$ and a user's question q, it finds the most similar q_j based on highest cosine similarity score in P and return the corresponding a_j as the reply. All the chatbot implemented components are developed using Python 3.7 programming language. We applied TensorFlowHub [3] to use the pretrained models in sentence encoder component.

At the first step in Query Semantic Understanding, the user's query and Reddit QA dataset are encoded into numeric vectors using the sentence encoder illustrated in Fig. 6.2. Sentence encoder module through the pretrained NLP model[4] generates vectors for QA dataset as well as user's query. To better refer to these vectors we denote the following symbols A_v, Q_v, Qu_v for answer, question and query variables respectively. The next step in QSU is the *question filtering* process in which the cosine similarity score between each question encoded vector (Q_v) and the query encoded vector (Qu_v) calculated. The output of this process is the id of a question with the largest cosine score. We refer to this question as Q_{max} as it has the maximum cosine similarity score to the user query. In the *answer filtering* process all the answers associated to the Q_{max} in the dataset will be selected and the filtering process calculates a cosine similarity score between those answers (A_v) and the query vector Qu_v. Finally, the answer with the largest cosine score returns as the response of the answer filtering process which is presented to the user query. In the following chapter we demonstrate the chatbot platform and several real use cases for opioid users.

Platform Demonstration

In this section we present our proposed chatbot technical components in more details. Next, we describe the interface of the chatbot following by experiment chapter.

[3]https://www.tensorflow.org/hub

[4]https://tfhub.dev/google/universal-sentence-encoder/3

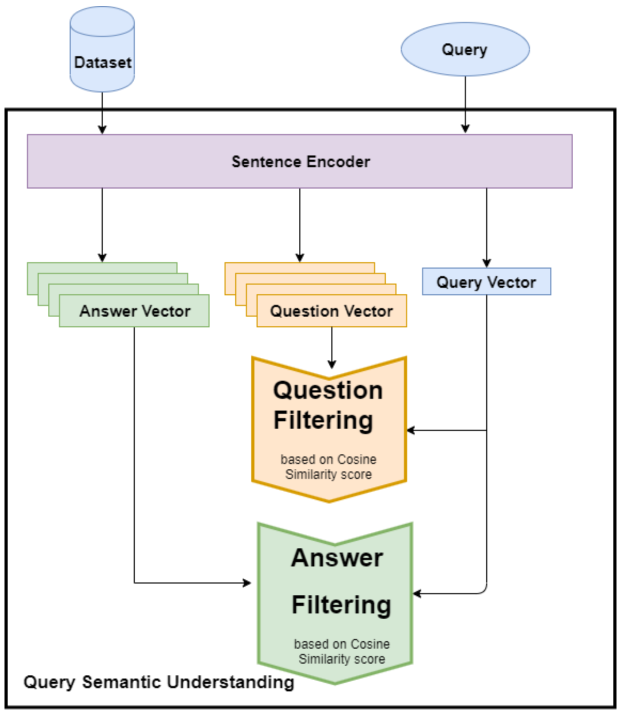


Figure 6.2. The Query Semantic Understanding in chatbot model

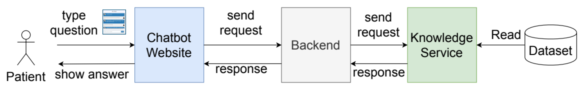


Figure 6.3. The overview design of opioid prototype

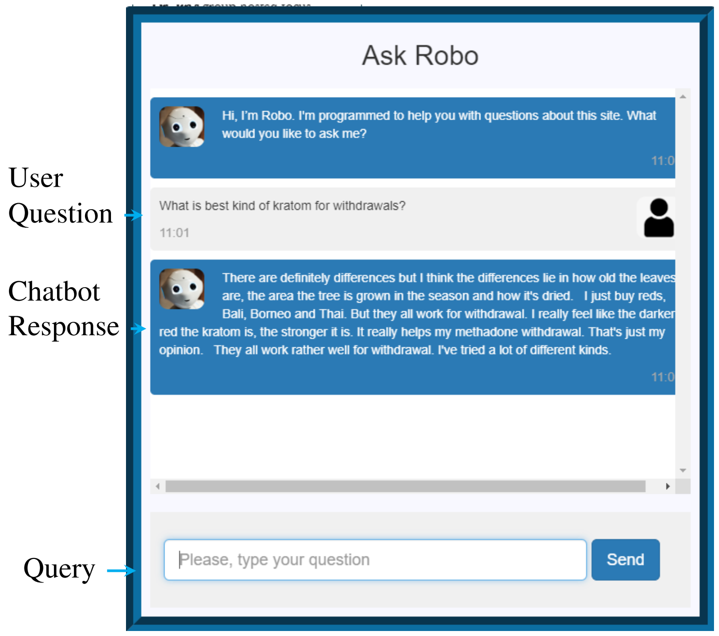
Technical Framework

The architecture of the chatbot is shown in Fig. 6.3. The individual (e.g., a patient, a patient's family member or friend) enters their query in the website through their device (e.g., computer, laptop, smartphone). The website sends a HTML request to the back-end server which checks the content of the request and forwards it to the service if it is valid. The knowledge service analyzes the query semantic using the machine learning algorithms to search the optimal response for the question and sends back the response. The back-end server gets the response and forwards it to the website as the message to the end user.

The chatbot web application is created using the cutting-edge technologies to provide quick and efficient responses to clients. The website is implemented using front-end technologies (i.e., AngularJS framework[5], Node.js[6]). and Express.js (i.e., a web application framework to handle REST API) [7] which accelerates the response time for multiple requests.

Chatbot Interface

Fig. 6.4 illustrates the landing page of the chatbot. The client can enter their question in the query field and press the enter key or the send button. The chatbot shows the user message and sends their query to API website to provide a suggestion using the machine learning techniques. After that, the chatbot response displays in the dialogue chatbot.



Hi, I'm Robo. I'm programmed to help you with questions about this site. What would you like to ask me?
11:0

What is best kind of kratom for withdrawals?
11:01

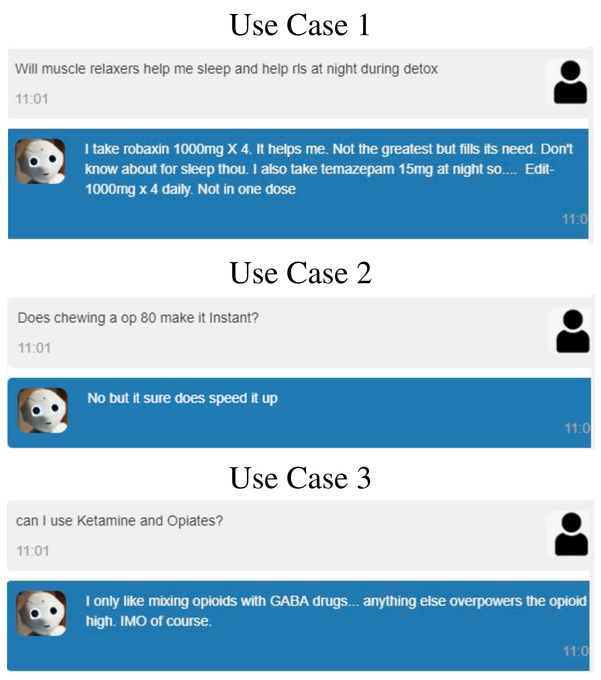
There are definitely differences but I think the differences lie in how old the leaves are, the area the tree is grown in the season and how it's dried. I just buy reds, Bali, Borneo and Thai. But they all work for withdrawal. I really feel like the darker red the kratom is, the stronger it is. It really helps my methadone withdrawal. That's just my opinion. They all work rather well for withdrawal. I've tried a lot of different kinds.
11:0

Figure 6.4. A chatbot interface with a sample of questions and answers

Use Case Studies

In this section, we examined three real scenarios in which patients inquired different questions on opioid. We explain these use cases on how effectively our chatbot responded in these cases.

[5]https://angularjs.org/
[6]https://nodejs.org/en/about/
[7]https://expressjs.com/


Use Case 1

Will muscle relaxers help me sleep and help rls at night during detox

11:01

I take robaxin 1000mg X 4. It helps me. Not the greatest but fills its need. Don't know about for sleep thou. I also take temazepam 15mg at night so.... Edit- 1000mg x 4 daily. Not in one dose

11:0

Use Case 2

Does chewing a op 80 make it Instant?

11:01

No but it sure does speed it up

11:0

Use Case 3

can I use Ketamine and Opiates?

11:01

I only like mixing opioids with GABA drugs... anything else overpowers the opioid high. IMO of course.

11:0

Figure 6.5. Three use cases of the chatbots

Use Case 1: The first picture of Fig. 6.5 shows a sample of an individual asking a question whether muscle relaxing medicine helps them to sleep and relax during night or not. The chatbot analyzed the query and searches the dataset to find the best answer using the Query Understanding Module discussed earlier. A quick and effective reply is shown on the browser using previous replies from the dataset. This opinion gives the user a useful response about the two medicines to take to get relax; also the chatbot clarifies that, the person who tried these medicines was not sure if the medicines helped for sleeping. Chatbot provided a details answer included the tablet mass in mg.
Use Case 2: Another example is illustrated in the middle picture of Fig. 6.5. In this example, an individual asks about the usage of opiates. They need to know if chewing the medicine impacts positively to observe the effective contents. The answer for this question is no. We can see how the chatbot assists in providing non-medical suggestion to use the drug based on the explicit knowledge crawled from previous data from the social network website $(reddit)$.
Use Case 3: The last use case is about mixing drugs. The third end-user wonders if using Ketamine and opiates together is safe or not. It is known that Ketamine is used for sev-

eral reasons such as maintaining anesthesia. Using our knowledge base, chatbot shares an experience of another individual who have used GABA drug. GABA is used also for maintaining anesthesia for relapse prevention. This response is correct advice which complies to the research in GABA and Ketamine. This scenario is one of quite often possible examples when an individual searches for side effects or usage misuse of mixing drugs.

CHAPTER VII
ANALYSIS OF FACEBOOK COMMENTS

Social media posts and their comments are rich in variation of subjects and an interesting pool for opinion mining. Individuals engage in social communications on Facebook through three behaviors: like, share and comment on a Facebook post. Responses to comments are grouped under the respective comment as a conversation thread. Conversation threads become interesting when users have conflicting views with the article posted, or with the opinion of another user. In this section[1] our research goal is to answer questions such as why some posts in Facebook receive more attention than others? Are conversation threads following a similar pattern between subjects like sport and politics? Are there any harmony between conversation threads of different subjects? We investigated how individuals react to different conversation subjects in the Facebook through a comprehensive analysis. our aim is to discover semantic and sentimental patterns in conversation threads categories. Finally, we employed Natural Language Processing techniques such as semantic and sentimental analysis and a statistical method like average response time (ART) or average comment length (ACL) of a post and observed that there are interesting patterns exists among different conversation threads.

Motivation

Researchers have studied comments in the various places YouTube videos comments, blog comments, comments on media releases websites [102, 100]. Comment mining is to study comments in computational linguistics and NLP literature to analyze attitude toward the comment, identify rumors or thread topics or categorizing unique comments, and much more. It aims to address questions on the sentiment analysis or opinion mining of comments [15, 92]. Social networks are growing at a fast rate without a break. Most importantly, the unstructured data that is being stored on these social medias is a pool of information pertains to a host of several categories containing governments, entertainment, news, travel, education, businesses, and health. In the health domain as an example, Medina et al. in their research [71], have focused on the sentiment of tweets to identify a correspondence to health issues and to gain a fresh outlook in analyzing health data.

[1]This research submitted for the special session in IEEE-Big Data 2020

Similarly, on the Facebook social media, researchers worked on sentiment classification and sentiment change detection on Facebook comments using the lexicon and machine learning-based approach[86]. The rising popularity of social media has radically changed the way news content is propagated, including interactive attempts with new dimensions. Praboda et al.[90] explored a set of news media page that originate content by themselves in the social media network.

In this research, we selected the CNN[2] news Facebook page as an enrich resource to analyze the comments. It has been discussed that the CNN followers are more interactive and gregarious and the most frequently shared posts regarding the USA elections were tackled by CNN followers [102, 54]. This research aims to provide a complete analysis of users reaction in a form of replying to Facebook posts in controversial threads for different topics.

In section VII, we discuss on data collection and introduce a proposed framework for the quantitative analysis in this research. Next, we explain the observation of our investigation in sections VII and VII which included sentimental and semantic analysis. We employed several machine learning and NLP algorithms in our experiment. Result and discussion will be explained in section VII.

Proposed Framework

Data Collection

In this section, we discuss the architecture framework and programming libraries we used in our experiment to pull and process the unstructured comments from the Facebook page. We selected 9 different topics from the CNN news Facebook page. As displayed in Fig.7.1, to extract comments from a post we employed a third-party tool named *export comments*[3] which is a web application that exports Facebook comments to a comma-separated file (CSV), thereafter we stored comments in a local database. We filtered the duplicated users to remove the bias and give each user an even opportunity to analyze his/her opinion. We implemented the pre-processing module in Python programming language as Python includes wide range of libraries to perform different NLP tasks such as removing non-English words or nested comments. The refined comments transferred to

[2]Cable News Network

[3]https://exportcomments.com/

Table 7.1. Topics and Captions pulled from Facebook.

Topics	Title	Total Comments
Entertainment	Kim Kardashian West and Kanye West welcome fourth child	676
Technology	Japan tests world's fastest bullet train	468
Travel	Thailand bay made popular by 'The Beach' to remain closed for two more years	162
Sport	Chicago Cubs ban a fan who was seen making a white power gesture behind a black reporter	442
Crime	El Chapo's attorney asks a judge to intervene over 'cruel and unusual' prison conditions	718
Health	More than 700 cases of mumps in the US this year, CDC says	312
Politics	Joe Biden once said a fence was needed to stop 'tons' of drugs from Mexico	67
Finance	The US just raised tariffs on Chinese goods. China says it will hit back	62
Weather	Houston gets drenched by overnight storms, and the region braces for another round later	88

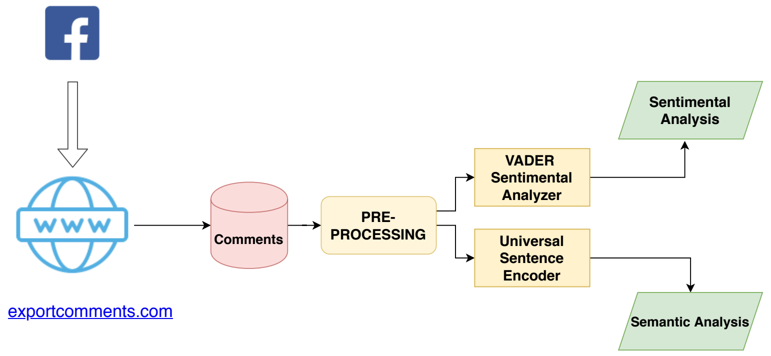


Figure 7.1. Architecture of framework to process comments.

the next modules for the sentiment and semantic analyses. Table 7.1 displays the topics, post title and the number of extracted comments.

Data Exploration

A quantitative percentile description of the Table 7.1 is illustrated in Fig. 7.2. This is a representation of users engagement over different subjects. The distribution implies the number of comments in subjects of crime and entertainment include the highest percentage of the collected records. However, it is opposite for the weather subject. We consider several factors including users engagement in commenting to a post to measure user activity to a particular topic.

VADER Sentimental Analyzer

We applied **V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner or VADER library in python programming language to calculate the semantic score of each comment. VADER [41] uses a lexicon and rule-based sentiment analysis tool that is mainly attuned to sentiments expressed in social media. The output of the sentiment of a text using the VADER

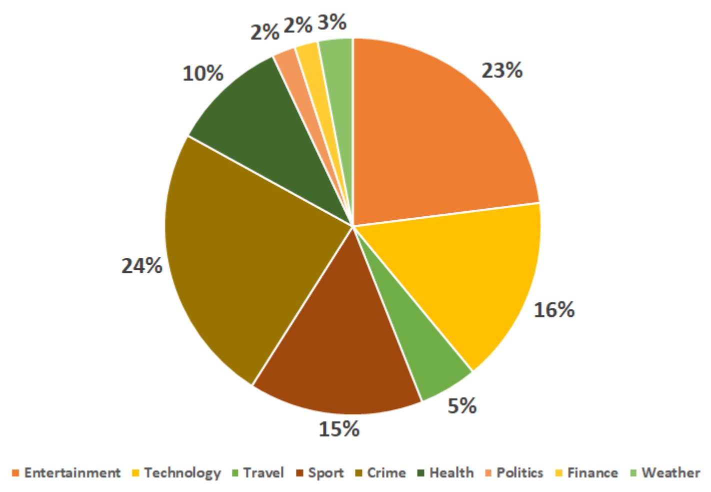
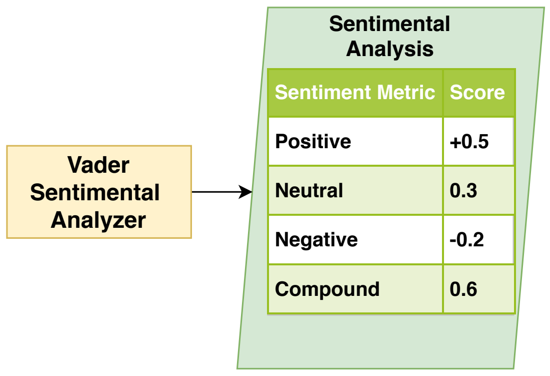


Figure 7.2. Total comments distribution over subjects.

would be subtle (Positive, Neutral, Negative, Compound) rating scores. The first three, positive, neutral, and negative, represents the proportion of the text that falls into those categories. The final metric, the compound score, is the sum of all of the lexicon ratings which have been standardized to range between -1 and 1 as is illustrated in Fig. 7.3 . VADER can handle informal writing - multiple punctuation marks, acronyms, and an emoticon which users quite often follow in the social medias when users write comments. we utilized the VADER over the collected comments of all topics to measure their compound score and refer it as the sentimental score in this work. A compound score was used to count how many comments are positive or negative depending on the sign of their compound score. The percentage of positive and negative comments is displayed in Fig. 7.4, 56% and 44% of total comments have positive and negative sentiment scores respectively.

Semantic Analyzer

The semantic analysis in a simple form is the process of interpreting the contextual clues surrounding a word to better understand the implication of the content of a sentence. In the semantic analysis, the text encoded into embedded vectors in the form of numeric representation to be understood by the computer. The idea behind the numeric representation is to encode the text in a lower dimension format that can treat a word using algebraic operations [42]. In our research, we applied available text encoders from Universal Sen-



Sentiment Metric	Score
Positive	+0.5
Neutral	0.3
Negative	-0.2
Compound	0.6

Figure 7.3. Sentimental module uses VADER to calculate sentimental score.

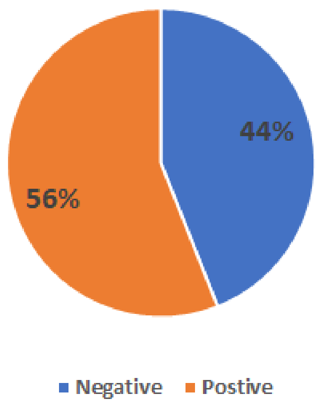


Figure 7.4. Sentimental scores of all topics.

tence Encoder (USE) [17] and Deep Average Network (DAN) [55] model over collected data. As shown in Fig. 7.5, DAN is the backbone engine of USE, it constructed on several layers of deep feed-forward neural networks. The objective of the stack of layers is that each layer learns a more abstract representation of the input than the previous one. They are trained on a large corpus in advance and can be used in a variety of tasks (sentimental analysis, classification, and so on). We employed this pre-trained model to identify any latent semantic patterns between each topic. A model takes a word, sentence, or paragraph as an input and outputs a 512-dimensional vector. All comments from a post were concatenated to build a single document. Next the document were encoded to a 512-dimensional vector. Once the 512-dimensional vectors created, we measured similarities between documents with cosine similarity function.

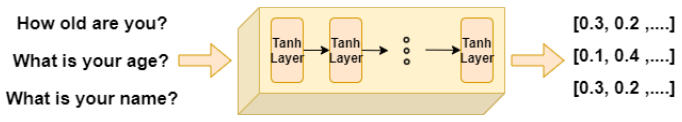


Figure 7.5. Deep Averaging Network (DAN) architecture

Cosine Similarity function

Cosine similarity function is a judgment method of orientation of two vectors. Basically if two vectors have a same orientation they will have a cosine similarity of 1 and if they are oriented at 90° relative to each other they will have a similarity of 0. In a case if they are diametrically opposite each other they will have a similarity of -1, independent of their magnitude. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded in $[0, 1]$. Fig. 7.6 shows two embedding vectors of $Crime$ and $Sport$ topics as an example. According to the Eq.7.1 we can calculate the α° which represents a score of semantic similarity between these two topics. Suppose q and r are the vectors representation of these two topics. The matching function between q and r can be formulated through the cosine similarity with a replacement of $Crime$ as q and $Sport$ as r.

$$score = cos(\boldsymbol{q}, \boldsymbol{r}) = \frac{\boldsymbol{q} \cdot \boldsymbol{r}}{||\boldsymbol{q}|| \cdot ||\boldsymbol{r}||} \tag{7.1}$$

As the comments of a topic converted to a vector for each category we utilized cosine similarity operation to calculate how similar are two topics to each other. The output of the cosine function for some sample topics are displayed in Fig.7.7. We will discuss of the semantic analysis result in the section VII.

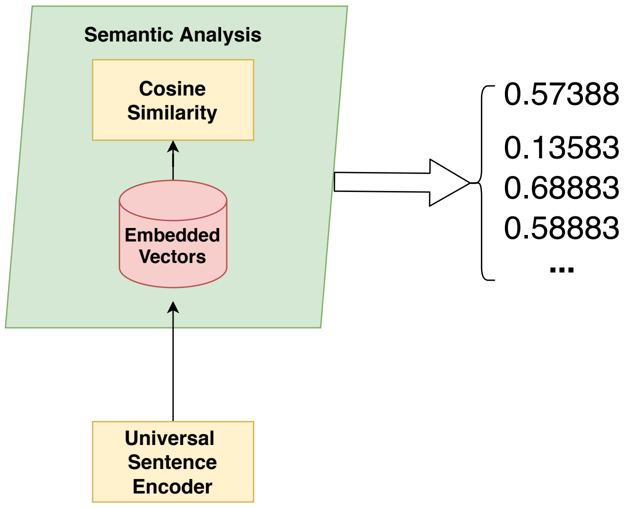


Figure 7.7. Semantic analysis module to calculate semantic score.

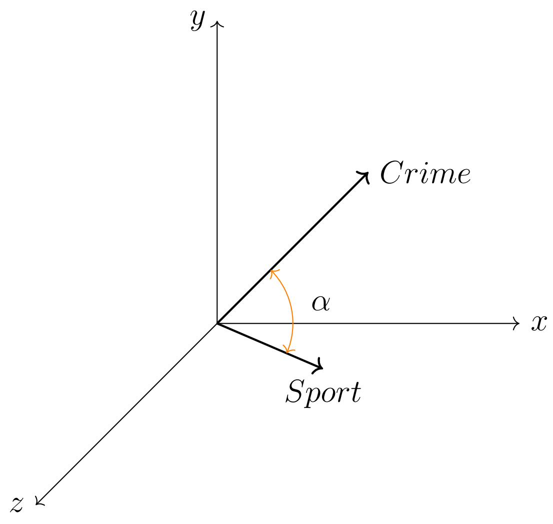


Figure 7.6. Cosine Similarity - α is the angle between two encoded vectors of $Crime$ and $Sport$.

Results and Discussion

In this section we discuss the sentimental analysis over the collected data and introduce several statistical methods we applied over the data to measure user engagement in replying to Facebook posts.

Table 7.2. Sentimental and Statistical Analysis over topics.

Subject	% of Positive Comments	% of Negative Comments	ACL	ART (min)	MRT (min)
Entertainment	0.659509	0.340491	50.3033	81.4038	23
Technology	0.638498	0.361502	69.2222	118.813	23
Travel	0.688525	0.311475	70.6235	171.166	25
Sport	0.487973	0.512027	93.8122	65.2305	5
Crime	0.346591	0.653409	96.5641	122.679	11
Health	0.605634	0.394366	98.9519	257.212	23
Politics	0.542222	0.457778	112.398	63.9517	67
Finance	0.466222	0.543478	128.613	405.526	62
Weather	0.490196	0.509804	149.295	145.088	6

We denoted σ as normalized sentiment score that is defined in VII. It calculates the ratio of number of comments with positive or negative sentiment to total number of comments:

$$\sigma = \frac{\mid s \mid}{\mid T \mid}$$

, s $\in$ $\{\mathcal{P}, \mathcal{N}\}$(7.2)where σ is the normalized sentimental score, $|T|$ is the total number of comments, $\mathcal{P}$ and $\mathcal{N}$ are the positive and negative comments. Fig. 7.8 illustrated the normalized positive and negative sentimental scores of each topic. As not all the comments of a particular topic are negative or positive, the figure implies that even on a subject such as crime there are some positive comments while the majority classified as comments with negative scores. The σ states that individuals in our study expressed a positive/negative reaction toward certain subjects. Crime followed by finance and sport are the categories with higher negative σ . On the other hand, technology, entertainment, and travel are categories with the highest positive σ. A high positive or negative sentimental score for a topic is a sign of attitude of users toward a post of that topic. For example in topics with positive sentiment score such as technology and entertainment commenters express their joy when replied back a comment and it increased the positive sentiment score. However, in the topic of crime their attitude is opposite. Hate, rage, fear are often noticeable in comments with crime subjects and they increased the negative sentiment for the topic in general. The higher value for σ means the intensity of reaction to a post as more people are expressing their feeling similarly. More especially, we observed that the magnitude of the emotion irrespective of its sentimental type is a higher value in positive subjects. It can imply that people expressed their joy with more emotion than their hate according to this result.

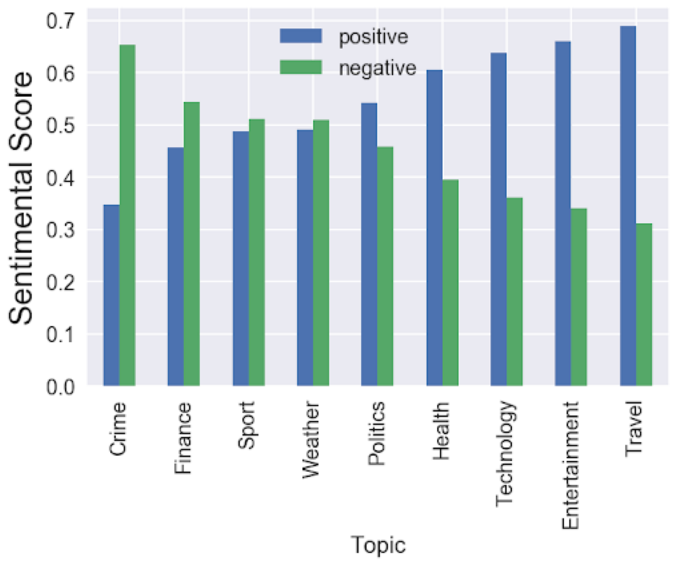


Figure 7.8. Distribution of sentimental scores per each topic

1. **Minimum Response Time (MRT)**
   We define the **M**inimum **R**esponse **T**ime (MRT), as the shortest time interval in minute of replying to a comment in a topic. Sentimental and statistical analysis in Table 7.2 illustrate that there are similar patterns between MRT and the sentimental score. For example, in topics of entertainment, travel, and technology with high positive sentimental percentage, they have similar MRT times. Additionally, a similar pattern is observed on topics with negative sentimental percentage such as sport and weather and their corresponding MRT times. This pattern can suggest that users react to comment in a shorter time when they read a post that has high negative or positive sentimental score [52, 2]. This could be through an empathy to a sad news or a joy for a happy event.

2. **Average Response Time (ART)**
   We define **A**verage **R**esponse **T**ime (ART) as the average time in minute that it took a comment posted on a topic. Topics of politic and sport have close sentimental score with lowest ART number and it can indicate the possibility of ongoing discussions in these topics. It can also infer that these topics are very debatable that drive to continuous engagement of users for commenting. Readers can study more

on the subject of engagements of users on sport and politics in social media in [1, 89]. The other interesting observation is in the compound score (average of positive and negative scores) for topics of sport and politics. The score is converged to a neutral value close to +49. It suggests that in debatable subjects that people usually have different opinions about a topic to argue ultimately the sentimental score has a potential to converge to a neutral value if both parties participate at the similar tone.

3. **Average Comment Length (ACL)**
   Average **C**omment **L**ength (ACL) is the average length of comments collected for a topic in character units. We did not notice a strong correlation between the sentimental score and ACL of a topic. However, in topics of health, crime, and sport the ACL values are close. Intuitively, when people are expressive they tend to reply to a comment with a lengthy response. Our research indicates that people are less expensive in topics of entertainment that has the lowest value of ACL and more expressive in weather that has the highest ACL value.

4. **Semantic Analysis**
   According to wikipedia, the semantic is "the linguistic and philosophical study of meaning". More specifically in Natural Language Processing, the semantic is how to let computers understand the meaning of a text. In our research, we were interested to know if there was any semantic similarity between the conversation of different topics irrespective if the exact keywords overlapped between topics. To measure the similarity between topics we contacted all the comments of a topic into a single document per each topic and fed into semantic analyzer described in section VII.

   The similarity score is displayed in Table 7.3. We applied a heat-map to visualize semantic correlation between topics and plotted in Fig. 7.9. In the figure the darker the cell, the higher the score is between two topics. The figure illustrated that there is a high semantic correlation between subjects of finance and politics with a score of 0.593097. A high semantic score between topics represent that commenters follow a similar pattern in expressing their thoughts in the comments for these two topics.

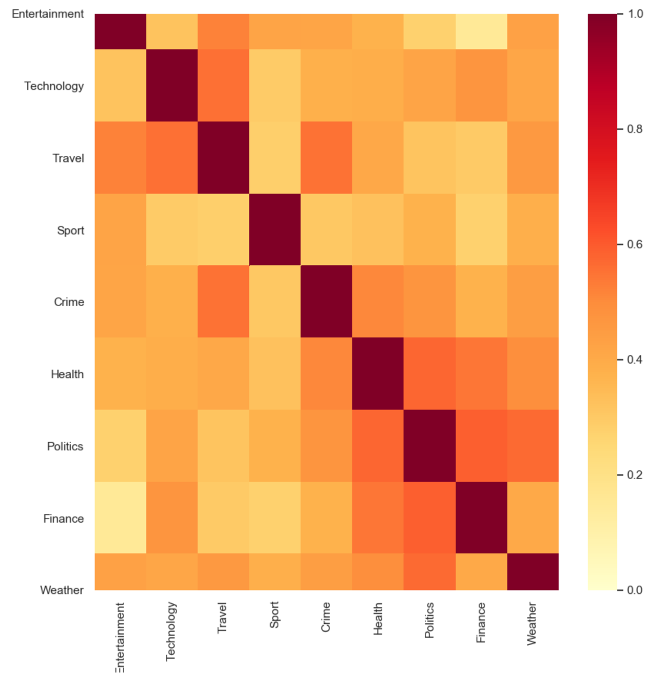


Figure 7.9. Semantic textual similarity

Table 7.3. Semantic score similarity

Entertainment	1	0.323587	0.520708	0.423074	0.418464	0.37774	0.27633	0.155075	0.432649
Technology	0.323587	1	0.557559	0.29630	0.381155	0.389669	0.422396	0.476173	0.414557
Travel	0.520708	0.557559	1	0.283387	0.553418	0.408158	0.317341	0.29717	0.457837
Sport	0.423074	0.296306	0.283387	1	0.304773	0.325181	0.376066	0.275898	0.383634
Crime	0.418464	0.381155	0.553418	0.304773	1	0.509343	0.473935	0.378542	0.441936
Health	0.37774	0.389669	0.408158	0.325181	0.509343	1	0.57659	0.544429	0.492934
Politics	0.27633	0.422396	0.317341	0.376066	0.473935	0.57659	1	0.593097	0.56898
Finance	0.155075	0.476173	0.29717	0.275898	0.378542	0.544429	0.593097	1	0.402763
Weather	0.432649	0.414557	0.457837	0.383634	0.441936	0.492934	0.56898	0.402763	1
	Entertainment	**Technology**	**Travel**	**Sport**	**Crime**	**Health**	**Politics**	**Finance**	**Weather**

CHAPTER VIII
PROPOSED METHODS

In this work we introduce two methods for text classification over big data, one to address sentimental latent characteristic of the text (Sent2Vec) and second to reduce the training time by decomposing an LSTM layer into two sub-layers (LDM) to build a network with less weights to train.

Sent2Vec: Sentimental Embedding

In this section [1], 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. 8.1.
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 $1 < i < 300$:

$$\mathcal{X} \triangleq \{x_1, x_2, \ldots, x_{300}\} \tag{8.1}$$

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.

$$\mathcal{Y} \triangleq \{y_1, y_2, \ldots, y_{300}\} \tag{8.2}$$

[1]This section has been submitted as a conference paper in IEEE Big Data 2020

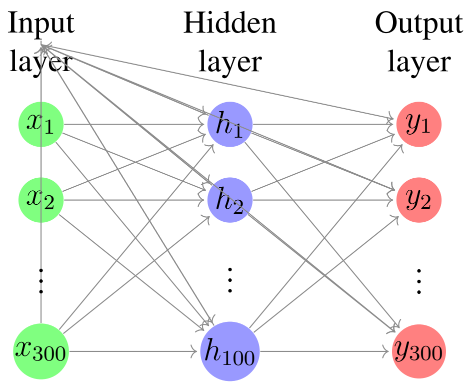


Figure 8.1. Sent2Vec Neural Network

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:

$$\mathcal{H} = \mathcal{F}_{relu}(\mathcal{W}_1\mathcal{X} + \beta_1) \tag{8.3}$$

$$\mathcal{Y} = \mathcal{F}_{sigmoid}(\mathcal{W}_2\mathcal{H} + \beta_2) \tag{8.4}$$

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

$$\begin{gathered}\mathcal{F}_{relu}(z) = \left\{\begin{matrix} z & z > 0 \\ \\ 0 & z <= 0 \end{matrix}\right\} \\ \mathcal{F}_{sigmoid}(z) = \frac{1}{1+e^{-z}}\end{gathered} \tag{8.5}$$

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

$$\text{Loss} = -\frac{1}{300}\sum_{i=1}^{300} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \tag{8.6}$$

$\hat{y}_i$ 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 $\mathcal{X}$.

$$\underbrace{-0.10, -0.15, \overbrace{0.013}^{x_i}, ..., 0.068, -0.164}_{\mathcal{X}} \Rightarrow \underbrace{0.01, 0.05, 0.07, \overbrace{0.5}^{\hat{y}_i}, ..., 0.0006, 0.13}_{\hat{\mathcal{Y}}}$$

LSTM Decomposing Technique

LSTM Decomposing Method (LDM) has advantage over the pruning and neural network architecture search algorithms since it does not require re-training and effectively considered as an off-line method. Consequently, our approach can save time and computing power if readers decide to implement and deploy our technique over cloud platform. Fig. 8.2 presents an original form of the RNN neural network with an LSTM layer for typical text classification problems. During an encoding process the input characters sequence needs to be transformed into a numerical format to be understandable by computer before running any mathematical operations. The encoding process is to form a vector representation in the embedding layer of an RNN. The embedding layers tailored the sequence to create a fixed input length. An LSTM layer consists of gated memory cells that can integrate information over longer time scales (as compared to simply using recurrent connections in a neural network). Each LSTM layer has a constant named $units$ which represents with a positive number and declared as the dimensionality of the output space. We refer to this parameter with the annotation N In this research. Finally a softmax function in the last layer computes a probability distribution over the 5 target classes. Fig. 8.3 presents the decomposed LSTM layer architecture where the bigger LSTM layer with N units is decomposed into two smaller LSTM with $[N/2]$ units each. The output of each decomposed layer concatenated before passes to the softmax layer. The Equation below demonstrates the formula for calculating number of weights in each LSTM layer.

$$4NM + N^2 + N \tag{8.7}$$

Where:

M = Embedding vector dimension

N = Units size

As shown in the (8.7) units size (N) impacts with the squared magnitude in the formula. Table 8.1 explains this impact, a higher value for N increases number of weights considerably in spite the value of M be fixed to 32 in our experiment. We will explain the advantage of the new topology in reducing the training time in decomposed LSTM versus the original RNN in section IX.

Table 8.1. LSTM Trainable Weights

Units	**LSTM Weights**
10	1720
20	4240
40	11680

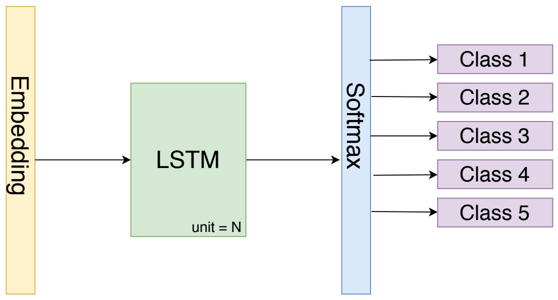


Figure 8.2. The original RNN model

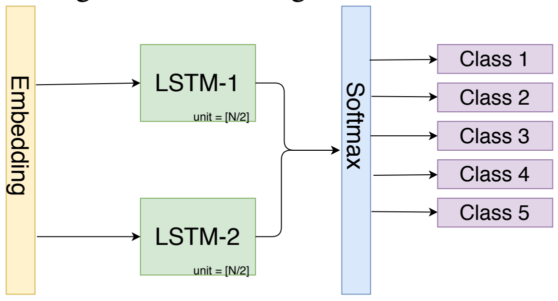


Figure 8.3. The decomposed RNN model

CHAPTER IX
EXPERIMENTS

Dataset

To analyze our model, we applied the neural networks designed in the previous steps over the Amazon review dataset that has been mostly used as a large dataset classification benchmark in [120, 25, 118]. 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 constructs a set of target classes labeled from 1 to 5. The reader is referred to (Zhang et al [120]) for more details on the construction of the data sets. We built a look up table according to these labels with a score associated to it, displayed in Table 9.1.

Implementation

In this section, we explained implementation of both methods Sent2Vec and LDM. We chose Python programming language for developing both techniques and collect our results on a desktop computer with 32GB RAM with i7 Intel processor.

Sent2Vec

The scores in the Table 9.1 are optimized values to maximize distance margin between classes in the vectors $\mathcal{Y}$. As shown in Fig. 9.1, 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 9.2) to build sentence embedding required by the

Table 9.1. Sentimental Scores

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

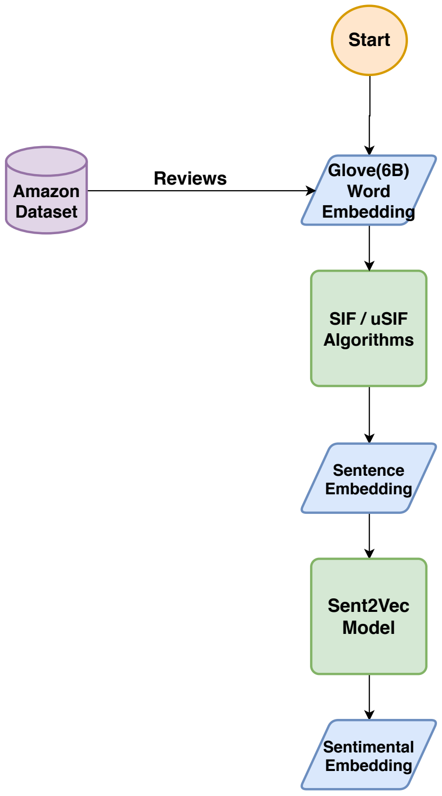
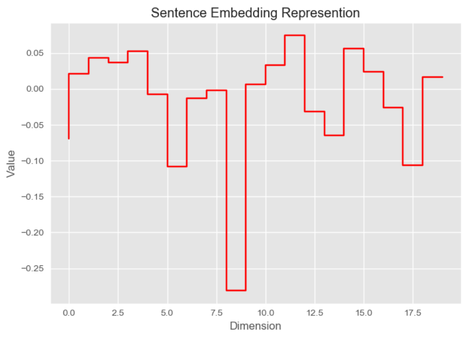
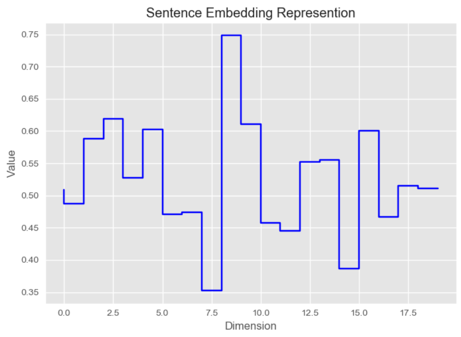


Figure 9.1. Training process of Sent2Vec neural network

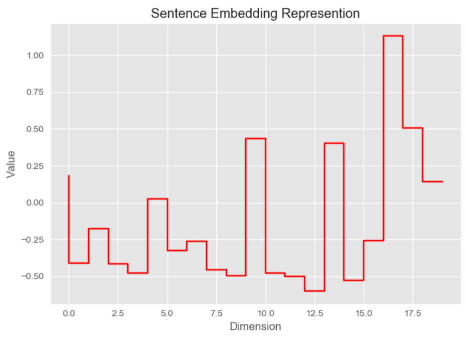
neural network. In this experiment, all the word embedding representations mentioned in the Table 9.2 have 300 dimensions. Each sentence embedding vector $\mathcal{X}$ represents an embedding for a review from the training dataset. In this research, a review is treated as a sentence that is fed to the Sent2Vec neural network. Eventually, to build the corresponding vector $\mathcal{Y}$, we multiplied vector $\mathcal{X}$ with an score from look up Table 9.1 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. 9.2 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.



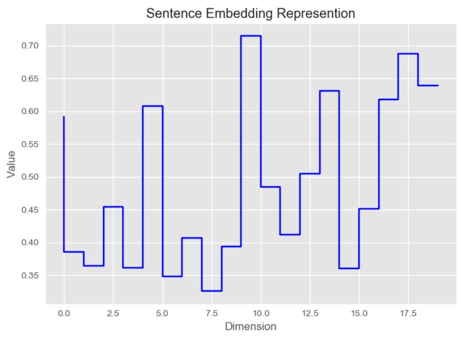
$\Longrightarrow$



Original Negative Embedding
Sentimental Negative Embedding



$\Longrightarrow$



Original Positive Embedding
Sentimental Positive Embedding

Figure 9.2. Embedding Transformation to Sentimental Embedding

LDM

We implemented the RNN with original and LSTM decomposed topologies using Keras[1] API in Python programming language. As it was outlined in the Fig. 8.2, in the original form of the RNN the input sequence of words adjusted to 500 characters to build a vector embedding representation with 32 dimensions per word. In our implementation we chose $ADAM$ optimizer for neural networks along with batch size of 256 and early stopping mechanism with $patience$ value of 3. As part of the requirement of the neural network we had to specify the vocabulary list built upon top frequent words. We observed that there is a slight difference between top 5000 and top 15000 most relevant words according to their TF-IDF score [91] as shown in Fig. 9.3 and Fig. 9.4. The larger vocabulary size entails to a higher number of weights to be trained in the neural network and more training time. Hence, We selected top 5000 frequent words in our model architecture as it had the very similar impact of a model with 15000 vocabulary. In other words, the important words are common in both vocabulary lists and a larger vocabulary may add unnecessarily noises. Furthermore, to align with the previous study on the dataset in [120], 5000 is the reasonable number.

[1]https://keras.io/

book
like
good
just
great
time
movie
did
read
don
really
little
love
use
better
story
buy
work
does
think
best
got
cd
new
way
want
product
bought
album
music
know
make
ve

Figure 9.3. Word cloud of top 5000 frequent words with highest TF-IDF

book
like
good
just
great
time
movie
read
really
album
love
little
use
new
buy
product
best
think
better
did
story
work
bought
way

Figure 9.4. Word cloud of top 15000 frequent words with highest TF-IDF

Evaluation

Our objective goal in this section is to evaluate the LDM method and Sent2Vec model over different metrics separately. The comparison between the original architecture and decomposed LSTM networks was to explore the impact of the new topology over the training time and network accuracy. The experimental study of the implemented decomposed LSTM method on a large document classification problem will be evaluated on various LSTM unit sizes. For the Sent2Vec model, we will apply different word embedding representations as shown in the Table 9.2 to build a sentence embedding through uSIF algorithm. As Fig. 9.5 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 the uSIF algorithms. We train a logistic regression classifier over those sentence embeddings and measure the performance of the classification. Additionally, we create sentence embedding using uSIF for

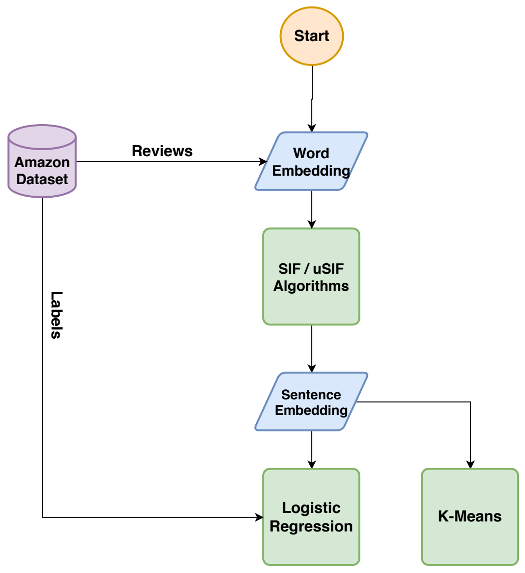


Figure 9.5. Testing process of different word embeddings

each candidate in Table 9.2 of a subset of samples and pass the embeddings into the K-Means clustering algorithm with K=5 and measure their cohesion per each cluster.

Sent2Vec

We employed 3 million Amazon reviews for the training and 650,000 reviews for the testing in the logistic regression with different embedding representations shown in Table 9.2. The reader is referred to (Zhang et al [120]) 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 architecture but trained over 1,000,000 and 300,000 samples respectively. Table 9.3 shows the logistic regression classification results of Sent2Vec against different embedding representations. We measured the performance of the logistic regression through the accuracy presented by the F1 score [96].

In Table 9.3, 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 9.3 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.
For the second experiment, we created sentence embedding of a subset of samples using uSIF for each candidate in Table 9.2 and passed the embeddings into the K-Means clustering algorithm with K=5 as illustrated in Fig. 9.5. Finally we measured the silhouette scores [94] 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 well matched to its own cluster and inadequately matched to neighboring clusters. For detail on the silhouette scores, readers may refer to [94].
Table 9.4 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. 9.6. The figures demonstrate how each cluster has been formed from its embedding representation of 10000 samples. Furthermore, our analysis according to the Table 9.4 and Fig. 9.6 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.
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 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 9.5. 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 9.6 and trained a logistic regression classifier with the different embedding repre-

sentations. The selected dataset have different label classes and different sizes. We trained the Sent2Vec model (Figure 8.1) over the training data per each dataset. Table 9.7 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 9.6.

LDM

Our objective goal in the comparison between the original architecture and decomposed LSTM networks is to explore the impact of the proposed neural network topology on the training time and network accuracy. In the first experiment, as it is implied in Table 9.8 we selected two original networks with LSTM units number of (N=20) and (N=40) respectively and evaluated with their decomposed versions. Each notation of X-X corresponds to a decomposed LSTM network with 10 or 20 units per each LSTM respectively. For example the network with notation of 10-10, has a sub-layer with two LSTMs where each LSTM layer has units of 10 (N = 10). Our first analyze indicates that between the original and decomposed LSTM, the decomposed ones are always trained in lesser time with a slight reduce of accuracy in this experiment. In the second part of the comparison, we applied the pre-trained LSTM layers on decomposed networks and included the accuracy of the new pre-trained decomposed model (denoted with X-X*) in the Table 9.8. To build a pre-trained LSTM, a neural network with single LSTM trained over the full dataset. For example in (10-10 *) a neural network with a single LSTM of 10 trained over the full dataset first and then the LSTM layer weight used in a decomposed (10-10) neural network to build a (10-10 *) decomposed pre-trained network. The results of a pre-trained decomposed network clarify even a further training time drops in a neural network. For instance, an original neural network of LSTM with 40 units that transformed into (20-20*) - a pre trained network decomposed- achieved almost 17 hours faster in training time (54 hours vs 37 hours) compared to its original architecture shown in Fig 8.2.
In another comprehensive analysis to verify the performance of the model using transfer learning, we trained the LSTM module over Yelp dataset to build the pre-trained network. We noticed although the network trained over a different dataset, yet the training time on the Amazon dataset is lesser than original and decomposed architectures without any

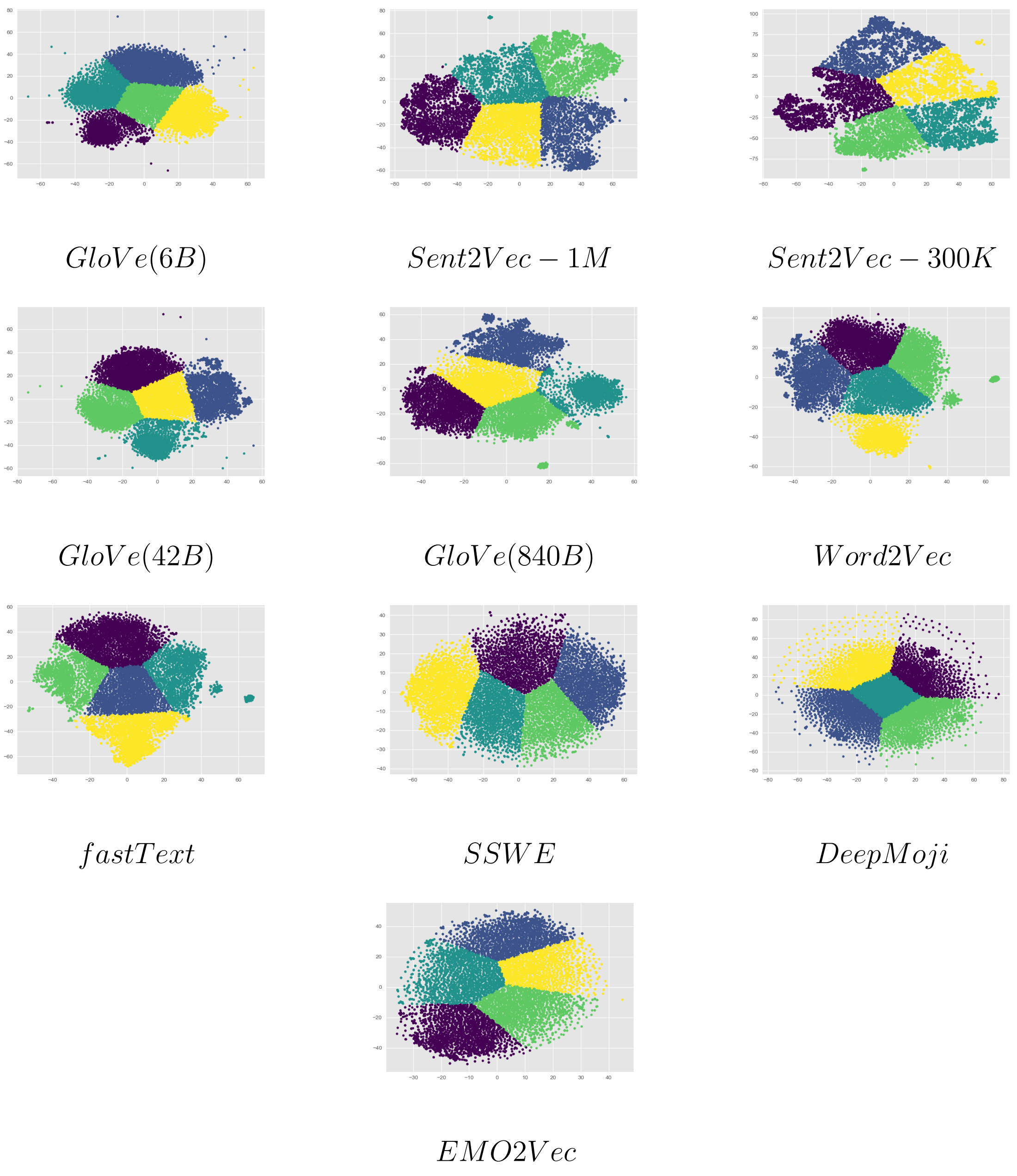


Figure 9.6. Cluster Representation for each Sentence Embedding

pre-training. However, it is not as fast as pre-trained decomposed models where the pre-trained LSTMs trained on Amazon dataset itself. In the last experiment, we used only 1 million records out of 3 million records to pre-train LSTM models and observed the performance follows similar behaviours as ones that we trained over the Yelp dataset for single layer LSTM architecture. However, in the pre-trained decomposed category, the network performed poorly in training time comparing to other decomposed structures. A possible reason that the training time is not surpassing the decomposed architectures can be the noises that exist in the dataset. This noise causes the pre-trained model to set the weights wrongly in the pre-trained phase and it took more time to converge once training over full 3 million records.

To better analysis of the training time between the original and decomposed topologies, we plotted the training epochs for these networks. Fig. 9.7 displays the accuracy of the neural network during training over EPOCHs. We noticed that the networks with decomposing topologies always converge with in lesser epochs than the corresponding original RNN networks. Pre-trained models can trained once and being employed in complex hierarchies to save training time further. It is clear that a network with a pre-trained LSTM over the dataset can be expected to start the training from a higher accuracy in early epochs and even converges quicker. Additionally, as the units increase on the single LSTM layer from 20 to 40 the accuracy of the network increases during the training. A possible strategy can be to use multiple decomposed LSTM layers where each layer has sufficient unit size.

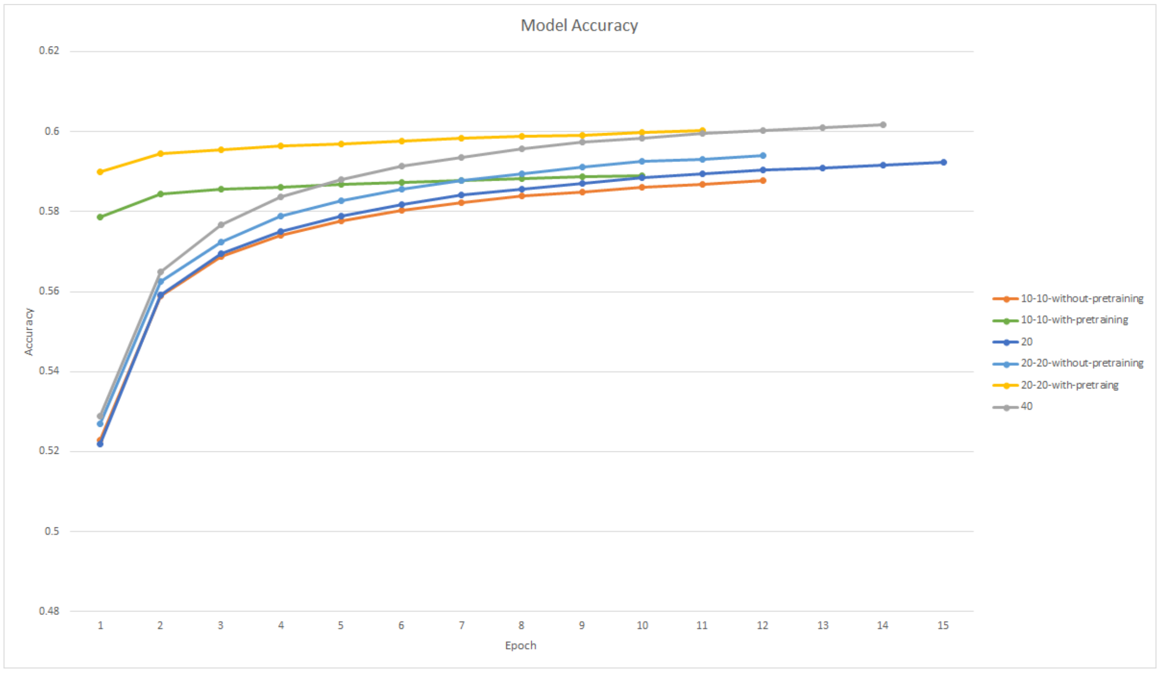


Figure 9.7. Model accuracy over training dataset

Table 9.2. 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 [107]
DeepMoji	Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm [37]
Emo2Vec	Emo2Vec: Learning Generalized Emotion Representation by Multi-task Training [117]

Table 9.3. 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

Table 9.4. 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

Table 9.5. 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 9.6. Dataset Description

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

Table 9.7. 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

Table 9.8. Accuracy

LSTM	**Training**	**Test**	**Training Time**	**Total Weights**
Original RNN				
20	59.23	57.63	17:01:50	164,345
40	60.16	58.41	2 days, 5:54:41	171,885
Decomposed LSTM				
10-10	58.76	57.30	13:22:43	163,545
20-20	59.39	57.76	1 day, 17:41:57	168,685
Pre-trained Decomposed LSTM				
10-10*	58.91	57.24	11:33:12	163,545
20-20*	60.02	57.95	1 day, 13:02:40	168,685
Pre-trained with Yelp dataset				
10-10*	58.67	57.16	12:26:50	163,545
20*	58.92	57.52	12:42:43	164,345
20-20*	59.54	57.94	1 day, 12:38:44	168,685
Pre-trained with Amazon subset (1 Million sample)				
10-10*	58.85	57.28	14:58:09	163,545
20*	58.96	57.50	11:57:36	164,345
20-20*	59.53	57.90	1 day, 20:28:58	168,685

CHAPTER X
DISCUSSION

In this paper we introduced two methods to address two major challenges in neural networks for sentimental analysis. First LSTM Decomposing Method (LDM) presented to decompose the processing unit of LSTM neural network to reduce the training time. The second innovation is Sent2Vec, a new data representation for sentences that boosts the performance of sentimental analysis by generating a new sentence representation.

Sent2Vec

For Sen2Vec, 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 0s and 1s where position of the numbers 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.

LDM

Experimental study of the implemented decomposed LSTM method on a large document classification problem showed reductions of training times for various LSTM unit sizes while attaining similar predictive power. The decomposed LSTM method has fewer trainable parameters than the original method, and we believe this might be one of the reasons for the reduced training time. Furthermore, the neural network is decomposed in such a way that subunits of the decomposed LSTM allow the use of pre-trained LSTM unit to start as the subunits. Leading to further reduction of training time for the decomposed LSTM network. The flexibility to use pre-trained subunits enable the re-use of models from earlier studies that employed LSTM network of smaller sizes. When results of earlier studies turned out to be inadequate in prediction accuracy, the efforts are not all lost and the LSTM units can be used as pre-trained units for later large scale LSTM network.

CHAPTER XI
CONCLUSION

Beyond understanding what is being discussed, human communication requires an awareness of what someone is feeling. Many conversational systems also known as chatbot relies on recognizing feelings in the conversation partner and replying accordingly which is a key communicative skill that is trivial for humans. An important module in developing such systems is a sentimental analysis component to recognize the feeling of received text. To improve the accuracy of the sentimental model most of the previous works have focused on improving the classification tasks using new algorithms. However, there is another trend which focuses on text representation that includes the semantic embedding. The success of the new embedding has shown an improvement in the accuracy of the sentimental classifier. In this work, we introduce two methods, one to reduce the training time and the other to improve accuracy of the sentimental classifier.

We proposed LSTM Decomposing Method (LDM), a method based on disintegrating the internal unit of a Recurrent Neural Network (RNN) into sub-units. Unlike previous similar researches, our approach does not need an active re-training during the parameter reduction phase. Additionally, we utilized transfer learning of an LSTM layer that trained over different dataset in Amazon dataset to reduce the training time further. Experimental tests reported in this paper has shown that the learning power of the new neural network over a large real dataset, indicated by prediction accuracy, has changed very slightly while the training time has been reduced signicantly.

Furthermore, we presented an alternative embedding representation that includes the sentimental semantic of a sentence in its embedding vector. 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. This can impact on the accuracy of a classification model to predict the correct sentimental score for a text. 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 word embeddings and an intermediate method is needed to convert word embedding to sentence embedding. 2) We applied 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.
Our future plan is to use Sent2Vec as the sentiment classifier for the chatbot app discussed in this dissertation. We plan to extend our research in another comprehensive study toward another set of experiments to observe the behaviour of the network while the vocabulary of texts are different but sentimental contexts remain fixed. Another experiment is to use LDM as the classifier when the input to the network is the Sent2Vec representation for an classification task, we intend to evaluate both accuracy and training time compared once the LDM and Sent2Vec merged together.

BIBLIOGRAPHY

[1] Rebecca M Achen, K Ryerson, and G Clavio. “What customers want: Defining engagement on social media in sport”. In: *Global Sport Business Journal* 5.3 (2017), pp. 1–21.

[2] Areej Alhothali and Jesse Hoey. “Good news or bad news: Using affect control theory to analyze readers’ reaction towards news articles”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1548–1558.

[3] Tim Althoff, Kevin Clark, and Jure Leskovec. “Large-scale Analysis of Counseling Conversations: An Application of Natural Language Processing to Mental Health”. en. In: *Trans Assoc Comput Linguist* 4 (2016), pp. 463–476.

[4] Jose M Alvarez and Mathieu Salzmann. “Compression-aware training of deep networks”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 856–867.

[5] Dario Amodei et al. “Deep speech 2: End-to-end speech recognition in english and mandarin”. In: *International conference on machine learning*. 2016, pp. 173–182.

[6] Chidanand Apt, Fred Damerau, and Sholom M Weiss. “Automated Learning of Decision Rules for Text Categorization”. In: *ACM Trans. Inf. Syst* 12 (1994), p. 233.

[7] Chidanand Apt, Fred Damerau, and Sholom M Weiss. “Towards Language Independent Automated Learning of Text Categorisation Models”. In: *SIGIR*. 1994, p. 23.

[8] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. “A simple but tough-to-beat baseline for sentence embeddings”. In: (2016).

[9] Sanjeev Arora et al. “A latent variable model approach to pmi-based word embeddings”. In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 385–399.

[10] Justin Bayer et al. “Evolving memory cell structures for sequence learning”. In: *International Conference on Artificial Neural Networks*. Springer. 2009, pp. 755–764.

[11] Jiang Bian, Umit Topaloglu, and Fan Yu. “Towards large-scale twitter mining for drug-related adverse events”. In: *Proceedings of the 2012 international workshop on Smart health and wellbeing*. ACM. 2012, pp. 25–32.

[12] Oliver Borchers. *Fast sentence embeddings*. https://github.com/oborchers/Fast_Sentence_Embeddings. 2019.

[13] Jacqueline Brixey et al. “Shihbot: A facebook chatbot for sexual health information on hiv/aids”. In: *Proceedings of the 18th annual SIGdial meeting on discourse and dialogue*. 2017, pp. 370–373.

[14] Andrew Brock et al. “Smash: one-shot model architecture search through hypernetworks”. In: *arXiv preprint arXiv:1708.05344* (2017).

[15] Lea Canales et al. “Exploiting a bootstrapping approach for automatic annotation of emotions in texts”. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2016, pp. 726–734.

[16] Daniel Cer et al. “Universal Sentence Encoder”. In: *CoRR* abs/1803.11175 (2018). arXiv: 1803.11175. URL: http://arxiv.org/abs/1803.11175.

[17] Daniel Cer et al. “Universal sentence encoder”. In: *arXiv preprint arXiv:1803.11175* (2018).

[18] Yee Seng Chan and Dan Roth. “Exploiting Syntactico-semantic Structures for Relation Extraction”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT ’11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 551–560. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002542.

[19] Minmin Chen. “Efficient Vector Representation for Documents through Corruption”. In: *CoRR* abs/1707.02377 (2017). arXiv: 1707.02377. URL: http://arxiv.org/abs/1707.02377.

[20] Minmin Chen et al. *Marginalized Denoising Autoencoders for Domain Adaptation*. 2012. arXiv: 1206.4683.

[21] Welin Chen, David Grangier, and Michael Auli. “Strategies for training large vocabulary neural language models”. In: *arXiv preprint arXiv:1512.04906* (2015).

[22] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[23] Junyoung Chung et al. “Gated feedback recurrent neural networks”. In: *International conference on machine learning*. 2015, pp. 2067–2075.

[24] Fabio Ciravegna. “Adaptive Information Extraction from Text by Rule Induction and Generalisation”. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI’01. Seattle, WA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 1251–1256. ISBN: 1-55860-812-5, 978-1-558-60812-2. URL: http://dl.acm.org/citation.cfm?id=1642194.1642261.

[25] Alexis Conneau et al. “Very deep convolutional networks for text classification”. In: *arXiv preprint arXiv:1606.01781* (2016).

[26] James R. Curran and Stephen Clark. “Language Independent NER Using a Maximum Entropy Tagger”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 164–167. DOI: 10.3115/1119176.1119200. URL: https://doi.org/10.3115/1119176.1119200.

[27] Andrew M. Dai and Quoc V. Le. *Semi-supervised Sequence Learning*. 2015. arXiv: 1511.01432.

[28] Andrew M. Dai, Christopher Olah, and Quoc V. Le. “Document Embedding with Paragraph Vectors”. In: *CoRR* abs/1507.07998 (2015). arXiv: 1507.07998. URL: http://arxiv.org/abs/1507.07998.

[29] Abhishek Das et al. “Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions?” In: *Comput. Vis. Image Underst.* 163 (Oct. 2017), pp. 90–100.

[30] Munmun De Choudhury and Sushovan De. “Mental health discourse on reddit: Self-disclosure, social support, and anonymity”. In: *Eighth International AAAI Conference on Weblogs and Social Media*. 2014.

[31] Scott Deerwester et al. “Indexing by latent semantic analysis”. In: *Journal of the American Society for Information Science* 41.6 (Sept. 1990), pp. 391–407. ISSN: 1097-4571. DOI: 10.1002/(sici)1097-4571(199009)41:6⟨391::aid-asi1⟩3.0.co;2-9. URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9.

[32] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[33] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[34] National Institute on Drug Abuse. *Drug Overdose Deaths in the United States*. https://www.drugabuse.gov/publications/drugfacts/prescription-opioids. Online, accessed 25-March-2018. January-2018.

[35] Richard O. Duda, Peter E. Hart, and David G. Stork. “Pattern Classification (2nd Edition)”. In: 2000.

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

[37] Bjarke Felbo et al. “Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm”. In: *arXiv preprint arXiv:1708.00524* (2017).

[38] Jiashi Feng and Trevor Darrell. “Learning the structure of deep convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2749–2757.

[39] Jonathan Frankle and Michael Carbin. “The lottery ticket hypothesis: Finding sparse, trainable neural networks”. In: *arXiv preprint arXiv:1803.03635* (2018).

[40] Haoyuan Gao et al. “Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C Cortes et al. Curran Associates, Inc., 2015, pp. 2296–2304.

[41] CHE Gilbert and Erric Hutto. “Vader: A parsimonious rule-based model for sentiment analysis of social media text”. In: *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*. Vol. 81. 2014, p. 82.

[42] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv preprint arXiv:1402.3722* (2014).

[43] Klaus Greff et al. “LSTM: A search space odyssey”. In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[44] Song Han, Huizi Mao, and William J Dally. “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding”. In: *arXiv preprint arXiv:1510.00149* (2015).

[45] Song Han et al. “Learning both weights and connections for efficient neural network”. In: *Advances in neural information processing systems*. 2015, pp. 1135–1143.

[46] Awni Hannun et al. “Deep speech: Scaling up end-to-end speech recognition”. In: *arXiv preprint arXiv:1412.5567* (2014).

[47] P J Hayes, P Nirenburg, and L M Schmandt. “Tcs: a shell for content-based text categorization”. In: *Proceedings of the Sixth IEEE CAIA*. 1990, pp. 320–326.

[48] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[49] Yihui He, Xiangyu Zhang, and Jian Sun. “Channel pruning for accelerating very deep neural networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1389–1397.

[50] Yihui He et al. “Amc: Automl for model compression and acceleration on mobile devices”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 784–800.

[51] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[52] Jacob Hornik et al. “Information dissemination via electronic word-of-mouth: Good news travels fast, bad news travels faster!” In: *Computers in Human Behavior* 45 (2015), pp. 273–280.

[53] Andreas Hotho, Andreas Nürnberger, and Gerhard Paass. “A Brief Survey of Text Mining”. In: *LDV Forum* 20 (2005), pp. 19–62.

[54] Mehtab Iqbal and Sushmita Khan. “Mining Facebook Page for Bi-Partisan Analysis”. In: *SAIS Proceedings 2018* (2018).

[55] Mohit Iyyer et al. “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)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1681–1691. DOI: 10.3115/v1/P15-1162. URL: https://www.aclweb.org/anthology/P15-1162.

[56] Mohit Iyyer et al. “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)*. 2015, pp. 1681–1691.

[57] Thorsten Joachims. “Text categorization with Support Vector Machines: Learning with many relevant features”. In: *Lecture Notes in Computer Science* (1998), pp. 137–142. ISSN: 1611-3349. DOI: 10.1007/bfb0026683. URL: http://dx.doi.org/10.1007/BFb0026683.

[58] Armand Joulin et al. “Bag of tricks for efficient text classification”. In: *arXiv preprint arXiv:1607.01759* (2016).

[59] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. “An empirical exploration of recurrent network architectures”. In: *International conference on machine learning*. 2015, pp. 2342–2350.

[60] Rafal Jozefowicz et al. “Exploring the limits of language modeling”. In: *arXiv preprint arXiv:1602.02410* (2016).

[61] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. “Grid long short-term memory”. In: *arXiv preprint arXiv:1507.01526* (2015).

[62] Yoon Kim et al. *Character-Aware Neural Language Models*. 2015. arXiv: 1508.06615.

[63] Ryan Kiros et al. *Skip-Thought Vectors*. 2015. arXiv: 1506.06726.

[64] 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)*. 2013, pp. 489–493.

[65] “Language Modeling for Information Retrieval”. In: (2003). DOI: 10.1007/978-94-017-0171-6. URL: http://dx.doi.org/10.1007/978-94-017-0171-6.

[66] Quoc V. Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 2014, pp. 1188–1196. URL: http://jmlr.org/proceedings/papers/v32/le14.html.

[67] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. “Snip: Single-shot network pruning based on connection sensitivity”. In: *arXiv preprint arXiv:1810.02340* (2018).

[68] Jiwei Li et al. “Deep Reinforcement Learning for Dialogue Generation”. In: (June 2016). arXiv: 1606.01541.

[69] Jiasen Lu et al. “Deeper lstm and normalized cnn visual question answering model”. In: *GitHub repository* (2015).

[70] Andrew McCallum and Kamal Nigam. “A Comparison of Event Models for Naive Bayes Text Classification”. In: 1998.

[71] Daniel Medina Sada et al. “A Preliminary Investigation with Twitter to Augment CVD Exposome Research”. In: *Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*. 2017, pp. 169–178.

[72] Grégoire Mesnil et al. *Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews*. 2014. arXiv: 1412.5335.

[73] Grégoire Mesnil et al. “Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews”. In: *CoRR* abs/1412.5335 (2014). arXiv: 1412.5335. URL: http://arxiv.org/abs/1412.5335.

[74] Risto Miikkulainen et al. “Evolving deep neural networks”. In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[75] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. “Linguistic Regularities in Continuous Space Word Representations”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*. 2013, pp. 746–751. URL: http://aclweb.org/anthology/N/N13/N13-1090.pdf.

[76] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546.

[77] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* abs/1301.3781 (2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[78] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[79] Tomáš Mikolov et al. “Recurrent neural network based language model”. In: *Eleventh Annual Conference of the International Speech Communication Association*. 2010.

[80] 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. URL: http://icml.cc/2012/papers/784.pdf.

[81] Joao Luis Zeni Montenegro, Cristiano André da Costa, and Rodrigo da Rosa Righi. “Survey of conversational agents in health”. In: *Expert Syst. Appl.* 129 (Sept. 2019), pp. 56–67.

[82] Robert R Morris et al. “Towards an Artificially Empathic Conversational Agent for Mental Health Applications: System Design and User Perceptions”. en. In: *J. Med. Internet Res.* 20.6 (June 2018), e10148.

[83] Gabriel Murray, Steve Renals, and Jean Carletta. “Extractive summarization of meeting recordings”. In: *INTERSPEECH*. 2005.

[84] Renato Negrinho and Geoff Gordon. “Deeparchitect: Automatically designing and training deep architectures”. In: *arXiv preprint arXiv:1704.08792* (2017).

[85] Ani Nenkova and Amit Bagga. “Facilitating email thread access by extractive summary generation”. In: Jan. 2003, pp. 287–296. DOI: 10.1075/cilt.260.32nen.

[86] Alvaro Ortigosa, José M Martın, and Rosa M Carro. “Sentiment analysis in Facebook and its application to e-learning”. In: *Computers in human behavior* 31 (2014), pp. 527–541.

[87] 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)*. 2014, pp. 1532–1543.

[88] J.R. Quinlan. In: *Machine Learning* 1.1 (1986), pp. 81–106. ISSN: 0885-6125. DOI: 10 . 1023 / a : 1022643204877. URL: http : / / dx . doi . org / 10 . 1023 / A : 1022643204877.

[89] Lee Rainie et al. “Social media and political engagement”. In: *Pew Internet & American Life Project* 19 (2012), pp. 2–13.

[90] Praboda Rajapaksha et al. “Inspecting interactions: Online news media synergies in social media”. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2018, pp. 535–539.

[91] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. Piscataway, NJ. 2003, pp. 133–142.

[92] Kumar Ravi and Vadlamani Ravi. “A survey on opinion mining and sentiment analysis: tasks, approaches and applications”. In: *Knowledge-Based Systems* 89 (2015), pp. 14–46.

[93] Esteban Real et al. “Large-scale evolution of image classifiers”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2902–2911.

[94] Peter J Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[95] Gerard Salton et al. “Automatic Text Structuring and Summarization”. In: *Inf. Process. Manage.* 33.2 (Mar. 1997), pp. 193–207. ISSN: 0306-4573. DOI: 10.1016/S0306-4573(96)00062-3. URL: http://dx.doi.org/10.1016/S0306-4573(96)00062-3.

[96] Yutaka Sasaki et al. *The truth of the f-measure. 2007*. 2007.

[97] *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011. DOI: 10.1017/CBO9781139042918.

[98] Robert E. Schapire and Yoram Singer. In: *Machine Learning* 39.2/3 (2000), pp. 135–168. ISSN: 0885-6125. DOI: 10.1023/a:1007649029923. URL: http://dx.doi.org/10.1023/A:1007649029923.

[99] *Seeking Drug Abuse Treatment: Know what to Ask*. en. National Institute on Drug Abuse, U.S. Department of Health and Human Services, National Institutes of Health, 2013.

[100] Rui Shi, Paul Messaris, and Joseph N Cappella. “Effects of online comments on smokers’ perception of antismoking public service announcements”. In: *Journal of Computer-Mediated Communication* 19.4 (2014), pp. 975–990.

[101] Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. “Syntactic Simplification for Improving Content Selection in Multi-document Summarization”. In: *Proceedings of the 20th International Conference on Computational Linguistics*. COLING ’04. Geneva, Switzerland: Association for Computational Linguistics, 2004. DOI: 10.3115/1220355.1220484. URL: https://doi.org/10.3115/1220355.1220484.

[102] Stefan Siersdorfer et al. “Analyzing and mining comments and comment ratings on the social web”. In: *ACM Transactions on the Web (TWEB)* 8.3 (2014), pp. 1–39.

[103] Richard Socher et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, WA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: http://www.aclweb.org/anthology/D13-1170.

[104] Karen Spärck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of Documentation* 60.5 (Oct. 2004), pp. 493–502. ISSN: 0022-0418. DOI: 10.1108/00220410410560573. URL: http://dx.doi.org/10.1108/00220410410560573.

[105] Kenneth O Stanley and Risto Miikkulainen. “Evolving neural networks through augmenting topologies”. In: *Evolutionary computation* 10.2 (2002), pp. 99–127.

[106] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*. 2015. arXiv: 1503.00075.

[107] Duyu Tang et al. “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)*. 2014, pp. 1555–1565.

[108] Vladimir N. Vapnik. “The Nature of Statistical Learning Theory”. In: (2000). DOI: 10.1007/978-1-4757-3264-1. URL: http://dx.doi.org/10.1007/978-1-4757-3264-1.

[109] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[110] Subhashini Venugopalan et al. “Sequence to sequence-video to text”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4534–4542.

[111] Subhashini Venugopalan et al. “Translating Videos to Natural Language Using Deep Recurrent Neural Networks”. In: (Dec. 2014). arXiv: 1412.4729.

[112] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning - ICML ’08* (2008). DOI: 10.1145/1390156.1390294. URL: http://dx.doi.org/10.1145/1390156.1390294.

[113] Hao Wang et al. “A Dataset for Research on Short-Text Conversations”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 935–945. URL: https://www.aclweb.org/anthology/D13-1096.

[114] Yashen Wang et al. “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)*. 2016, pp. 505–515.

[115] Wei Wen et al. “Learning structured sparsity in deep neural networks”. In: *Advances in neural information processing systems*. 2016, pp. 2074–2082.

[116] Lingxi Xie and Alan Yuille. “Genetic cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1379–1388.

[117] Peng Xu et al. “Emo2vec: Learning generalized emotion representation by multi-task training”. In: *arXiv preprint arXiv:1809.04505* (2018).

[118] Zichao Yang et al. “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*. 2016, pp. 1480–1489.

[119] Wen-tau Yih et al. “Multi-document Summarization by Maximizing Informative Content-words”. In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI’07. Hyderabad, India: Morgan Kaufmann Publishers

Inc., 2007, pp. 1776–1782. URL: http://dl.acm.org/citation.cfm?id=1625275.1625563.

[120] Xiang Zhang and Yann LeCun. *Text Understanding from Scratch*. 2015. arXiv: 1502.01710.

[121] Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. "Practical network blocks design with q-learning". In: *arXiv preprint arXiv:1708.05552* 6 (2017).

[122] Julian Georg Zilly et al. "Recurrent highway networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 4189–4198.

[123] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (2016).

[124] Barret Zoph et al. "Learning transferable architectures for scalable image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.