Sentence Encoders for Semantic Textual Similarity

- A Survey

Aarthi Babu

March 19, 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Processing (NLP) is a field at the intersection of Linguistics and Artificial Intelligence (AI) (Jurafsky and Martin, 2014). The major goal of this field is to make computers process and understand human languages. Language processing helps to perform many useful tasks for humans such as language translation, information extraction, intelligent search engines, and question answering systems etc. (Jurafsky and Martin, 2014). These tasks involve processing text and understanding the meaning of words and phrases in it.

This project studies and evaluates the ability of different machine learning algorithms proposed for sentence understanding problems in NLP. The text meaning extracted using these algorithms, are estimated using Semantic Textual Similarity task. This study helps to advance our understanding of existing models. In specific, we study how machine learning algorithm is used to represent the meaning of the given text. Therefore, it is essential to understand what constitutes a meaning. Section 1.1 and 1.2 briefly introduces how semantics are interpreted in the field of linguistics and computer science.

1.1 Natural Language Semantics

In the field of Linguistics, Semantics studies the meaning or sense of words and their relationship with other words. Contextual information is always used to express the sense of the word. Meaning can be classified as Lexical and Grammatical. Lexical meaning denotes the meaning of the words based on its parts of speech tags such as noun, verbs, adverbs, adjectives etc. The latter signifies the meaning of the words that are related to the function of the sentence such as articles, determiners, pronouns or prepositions.

Many properties of human languages make language processing a very complex task for computers. For words, a single word can have various different meanings also known as sense. For example, a word *bank* can mean a river bank or financial institution. But when humans read a sentence consisting of *bank* and *wood*, they understand the meaning through context with their knowledge of the world. In some cases, the meaning of a word is part of another word such as animal and cat. Therefore, we can deduce that the words have a similar meaning when they share context.

Harris (1954) and Firth (1957) formulated a hypothesis that *"Words which occur in similar contexts tend to have similar meanings"*. There have been many approaches based on this hypothesis that tried to capture and represent the meaning of a word using words that occur around it. For example, consider a text *"One bottle of Tesgino makes you drunk.*

We make tesgino out of corn" (Jurafsky and Martin, 2014). The word *tesgino* appears to be an alcoholic drink based on the words such as *drunk* and *bottle*. The same words (drunk and bottle) often occur around the word beer, which has a similar meaning. There are a lot of techniques to find similar words, but fewer approaches to find similar sentences.

The compositional and ambiguous feature of the sentences makes it complex to process and understand its meaning. For example, the compositional meaning of *big apple* may not mean *large apple*, but maybe *New York city*. An ambiguous sentence can have two different meanings. For example, *"We saw her duck "* can mean either *We looked at a duck that belonged to her*, or *We saw her duck under something*. Languages are also highly variable in structure as *I ate pizza with friends* can also be expressed as *Friends and I shared some pizza* (Jurafsky and Martin, 2014).

1.2 Computational Semantics

Most commonly accepted methods to determine the word similarity are knowledge-based and corpus-based methods. The knowledge-based approach uses structure resources such as WordNet (Pedersen et al., 2004) consisting of highly relevant information like synonyms, words relation tree etc. The corpus-based method measures word similarity using sizable raw text corpora as a source data to infer information such as co-occurrence, and the frequency of the words. Techniques such as term frequency and inverse document frequency, built based on document's word distribution, proved to be very useful and was successfully used in an information retrieval system (Salton, 1971; Deerwester et al., 1989). Later, these vectors were used as features in various machine learning algorithms. Context-based meaning extraction hypothesis was successfully used in language modelling (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Mikolov et al., 2011) and word representation models (Mikolov et al., 2013; Pennington et al., 2014).

Neural models have become more effective in many complex NLP problems such as neural machine translation systems (Luong et al., 2015), sentiment analysis (Socher et al., 2011), and text generation (Wen et al., 2015) . Even though the representations created by the neural models are latent and not interpretable, they were a huge success in capturing the word meaning. These models were capable of learning and using their numeric representations for many other downstream tasks. Models were also proposed to capture sentence-level semantics using these word representations. (Kiros et al., 2015; Conneau et al., 2017; Shao, 2017). The success of these neural networks algorithms applied in semantically complex tasks makes it a potential component of this comparison study. Also, study on the accuracy of semantics captured by these algorithms helps to infer insights that would lead to significant progress in language understanding problems.

1.3 Project Overview

Despite the existence of well-established models for generating word representation and the consensus about its usefulness, the existing techniques proposed for learning the sentence representations have not fully captured the complexity of compositional semantics (Conneau et al., 2017). In this project, we compared various machine learning techniques used to determine the semantic representation of a text. Semantic Text Similarity (STS) is used as a primary task to evaluate these models (Agirre et al., 2012). STS task was proposed to stimulate research and to encourage the development of new approaches for modelling sentence-level semantics. This task can be used to evaluate and investigate the capability of the machine learning techniques proposed for learning text semantics; as it is important to capture the meaning of a sentence to perform well in this task.

In this project, we compare models that achieved the state of the art performance in Sem-Eval STS shared tasks (Cer et al., 2017). In addition to the comparative study, we will also analyse the internal component of various architectures and their impact on STS task performance.

This chapter gives an overview of meaning and the complex nature of language. Chapter 2 presents a background of the techniques used to capture the semantics of a textual data from words to sentences. Chapter 3 discusses the existing sentence representation models and its limitations on the sentence similarity task. Chapter 4 outlays the comparison study and its motivation. It also presents the architecture and the algorithm details of the encoders. Finally, Chapter 5 presents the experiments and its evaluation that establish the sentence encoder's ability to capture accurate representations.

Chapter 2

Background

This chapter introduces the technical concepts related to this project. Section 2.1 introduces Semantic Textual Similarity (STS) in more detail and discusses its application. Section and 2.3 discusses the principles of feature-based machine learning and neural networks approach. Section 2.4 and 2.5 discusses their use in sentence encoding.

2.1 Semantic Textual Similarity (STS)

STS is the task of finding how two sentences are closely related concerning its meaning (Agirre et al., 2012). It constitutes as a primary component in many natural language processing (NLP) applications. Until 2012, there was no unified framework available to study problems related to the semantic analysis of text data. Because of this, it was difficult to measure the performance and visualize the impact of different sentence representation approaches on NLP applications. In 2012, the association of Computer Linguistics (ACL) introduced a shared task conference for STS. The major focus of this conference was to define the STS research problem and standardize the dataset for it. This shared task event encouraged extensive evaluation of the approaches proposed every year. (Agirre et al., 2012). The STS task has two sub-tasks: 1) Sentence Relatedness, and 2) Recognizing Textual Entailment (RTE). Sentence Relatedness aims to find the semantic similarity score ranging from 0 to 5 between two sentences.

Table 2.1: Degree for semantic relatedness (similarity score) (Agirre et al., 2016)

Score	**Score reasoning and Sentence Pairs**	
0	**The two sentences are completely dissimilar.**	
	The black dog is running through the snow.	A race car driver is driving his car through the mud.
1	**The two sentences are not equivalent, but are on the same topic.**	
	The woman is playing the violin.	The young lady enjoys listening to the guitar.
2	**The two sentences are not equivalent, but share some details.**	
	They flew out of the nest in groups.	They flew into the nest together.
3	**The two sentences are roughly equivalent, but some important information differs/missing.**	
	John said he is considered a witness but not a suspect.	He is not a suspect anymore. John said.
4	**The two sentences are mostly equivalent, but some unimportant details differ.**	
	Two boys on a couch are playing video games.	Two boys are playing a video game.
5	**The two sentences are completely equivalent, as they mean the same thing.**	
	The bird is bathing in the sink.	Birdie is washing itself in the water basin.

Table 2.1 discusses the reasoning for the similarity score. Recognizing Textual Entailment measures the existence of the meaning overlap between two sentences, and classifies the relationship into three categories: 1) Entailment (E); 2) Contradiction (C); 3) Neutral

(N). Table 2.2 explains the reasoning behind these target labels.

In this project, the experiments are built around Semantic Textual Similarity with an objective to study and determine the usefulness of the semantics captured by the encoder algorithms. This task serves to be ideal for this comparison study because to succeed in this task; the representation should consist of proper semantics.

Table 2.2: Recognizing Textual Entailment (Classification Label) (Jurafsky and Martin, 2014)

Class Label	**Class reasoning and Sentence Pairs**	
Entailment	**Meaning overlap exists.**	
	If you help the needy, God will reward you.	Giving money to a poor man has good consequences.
Contradiction	**The meaning of two sentences contradict with each other.**	
	If you help the needy, God will reward you.	Giving money to a poor man has no consequences.
Neutral	**There is no meaning overlap**	
	If you help the needy, God will reward you.	Giving money to a poor man will make you a better person.

2.1.1 STS Applications

Semantic similarity between two sentences is a fundamental Natural Language Understanding (NLU) problem that is applicable in many NLP tasks such as information retrieval, evaluation of machine translation system, and automatic text summarization etc., (Agirre et al., 2016). STS models act as a primary software component in many applications such as image-captioning, automatic short question answer grading, search engines, plagiarism, and newswire headlines etc. (Agirre et al., 2016). For example, STS tasks are being used as a plagiarism checker by classifying the input text into following categories: 1) copying and pasting individual sentences from Wikipedia; 2) light revision of material copied from Wikipedia; 3) massive change of content from Wikipedia; 4) non-plagiarised answers produced without even looking at Wikipedia (Agirre et al., 2015). Similarly, STS can also be used to automatically evaluate the quality of machine translation systems, by comparing the machine-generated translations and its corresponding gold standard translations generated by humans Agirre et al. (2015).

2.1.2 Data for STS

The performance of machine learning models highly depends on the quality of its training data, as the machine learning models are approximated based on the interactions between input and output values. In the case of neural networks, the size of the data has a large influence on its performance as it doesn't get a chance to optimize well without training on various domains or aspects of the problem. In this project, the models are trained using dataset from Standford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), and Sentences Involving Compositional Knowledge (SICK) corpus (released as SemEval 2014 Task 1, (Marelli et al., 2014)).

The SICK Marelli et al. (2014) was constructed using the human-annotated corpus for Sentence Relatedness and RTE tasks. It has 4500 sentence pairs that include the collection of STS dataset created using corpus from various domains. The training data domains include

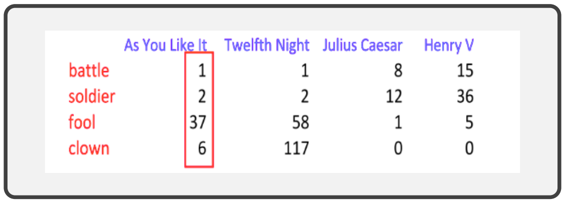
1. News headlines from the RSS feed of the European Media Monitor
2. Image captions from Flickr
3. Pairs of answers collected from Stack Exchange
4. Student answers paired with correct reference answers from the BEETLE corpus
5. Forum discussions about beliefs from the DEFT Committed Belief Annotation dataset

SNLI corpus consist of data from Amazon Mechanical Turk, an online marketplace, and Flickr30K corpus consisting 160k captions (Bowman et al., 2015). It was a hand annotated dataset comprising 550k sentence pairs.

2.2 Brief History of Meaning Representation

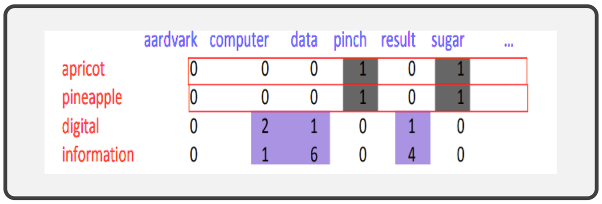
Any NLU problem starts with the challenge of describing words and sentences in the form of a machine-understandable representation, i.e. a vectorial representation that encodes its meaning. Historically, many knowledge-based and vector-based approaches have been proposed to estimate vector representations of words. Many algorithms used WordNet, a lexical knowledge-base to measure similarity, word hierarchy etc.. In a corpus-based approach, the representations are built based on the co-occurrence matrix of words in the documents consisting of the raw text corpus. It was also known as the distributional representation as it represents the meaning of the word from the distribution of words that occur around it. All the unique words in documents form a vocabulary V of the model.

Figure 2.1: Term-Document Matrix



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Figure 2.2: Term-Context Matrix



	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

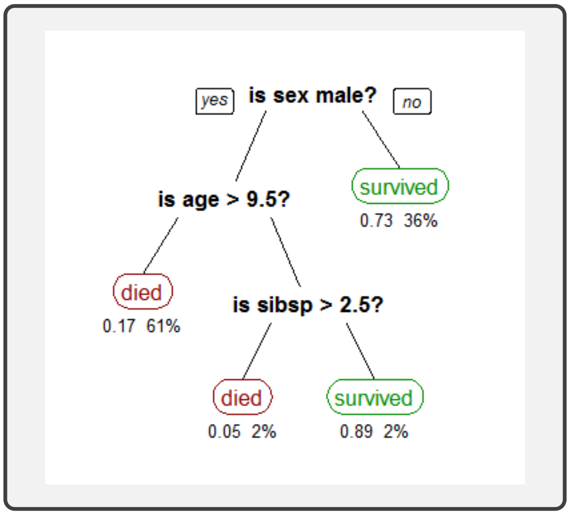
All the words in the V were used to build the Vector representation. Initially, the term-document matrix was introduced to represent a set of documents as vector, based on its content. In Figure 2.1, the term-document matrix consists of four novels as its column and all the unique words from V as its rows. With the word occurrence count in *As You Like it* column in Figure 2.1, it's clear that novel belongs to a comedy genre. This matrix was used to find similar documents as part of an information retrieval system (Salton, 1971), based on the idea that the related documents have almost the same distribution of words resulting in similar vectors. Each row in this matrix represents the word, but its not very accurate in carrying contextual information.

Term-context matrix was introduced to handle this limitation and measure the similarity between words. It used vocabulary words as its columns as well as its rows. Given the size of the vocabulary $|V|$, the term-context matrix carries the word co-occurrence count, within a specific window size, with a dimension $|V| \times |V|$ as shown in Figure 2.2. These representations are also called Sparse Vector Representations as most of the matrix cell values are zero. They mostly capture syntactic information rather than the semantics of the word as the window size gets smaller. The difference in orientation of two vectors denotes the measure of similarity between words. Usage of a sparse vector model for any semantic analysis task was computationally complex. To overcome this issue, many models were proposed to generate short and dense representations as: 1) dimensionality reduction using singular value decomposition; 2) neural network approaches like skip-gram and Continous Bag of Word (CBOW). In this project, we will focus on the models that use neural networks for creating words and sentence representations, as the latter method is more computationally efficient than the former approach (Jurafsky and Martin, 2014).

2.3 Machine Learning models

The learning theory and pattern recognition in AI gave rise to the field of machine learning (Jurafsky and Martin, 2014). The main objective of the machine learning algorithms is to learn from the previous data and predicts future data based on its previous knowledge. It learns a hypothesis consisting of weight parameters, which map the input features to output or target values. For any training set with input and output $\{(x_i, y_i)\}_{i=1}^{n}\}$, these algorithms learn a model *h(x)* = *y* from a collection of statistical feature set, extracted from the input in the training dataset. Then, it makes predictions on the unseen data. While training , the parameters are optimized based on an error rate of training time output $\bar{y_i}$ against true

Figure 2.3: A simple Decision Tree



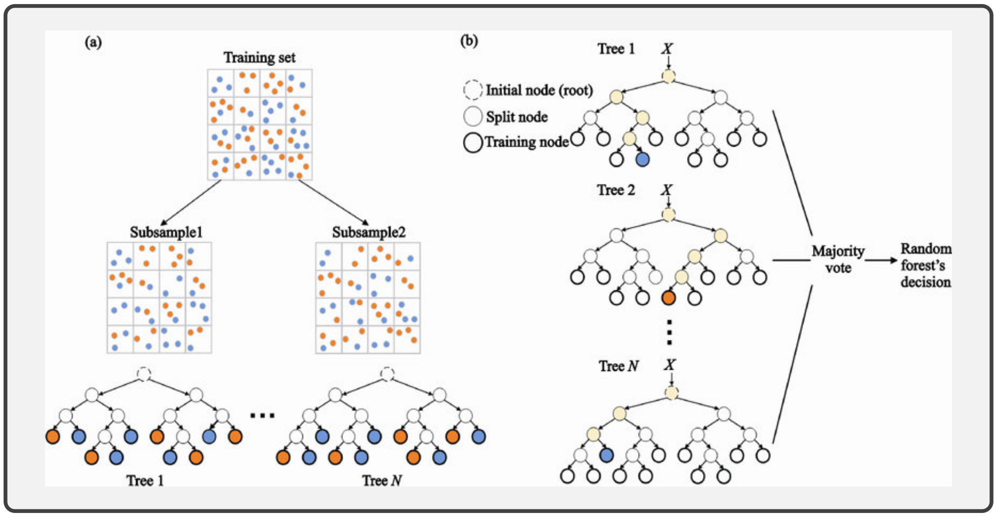
output y_i, for k features $< x_i^1, x_i^2, .., x_i^k >$ extracted from the input x_i. The data with discrete outputs such as RTE tasks are segregated as the classification problem, and the data with the continuous value output such as sentence relatedness, are classified as Regression problem.

2.3.1 Traditional ML models

In this project, the ensemble of machine learning algorithms such as Support Vector Machine, Random Forest, and Gradient Boosting, is used for learning an RTE classifier and a sentence relatedness task-based regressor.

Random Forest is an ensemble classifier that consists of a collection decision trees. Decision Tree is a tree with leaf and decision nodes that represent the function that takes a vector of feature values, and outputs a single target value. The output value can be discrete or continuous. It decides the prediction based on a specific sequence of rules. Each internal

Figure 2.4: Random Forest



node corresponds to a condition applied to the training set, using any one of the input features, that splits the data as shown in Figure 2.3. Although decision tree performs well, it is prone to over-fit as the decision node rules are built from the distribution of the training data. Therefore, it does not generalize well on the unseen data while testing. But, Random forest does not over-fit as it is an ensemble of decision trees, constructed from the sub-samples of the training data as shown in Figure 2.4.

Support Vector Machine (SVM) algorithms apply to both linear and non-linear data. It is one of the most popular robust algorithms which performs well even with the small quantity of data .i.e with less prior knowledge about the problem domain. The critical trick that contributes to its robustness is maximum margin separator, a decision boundary with the most significant possible distance between the hypothesis and the training data points. This decision boundary helps the model to generalize well on unseen data. SVM handles linearly non-separable input data points, by expanding the hypothesis space, by transforming the input data into higher dimensional space. The data points are projected to a higher dimension using a kernel function. In higher dimension, a linear separating hyper-plane is created using kernel function to classify the input data with its class label. This separating hyper-plane is a non-linear line in the original space.

Boosting algorithm is an ensemble of a set of learning algorithms that combines many base models that have limited prediction ability. Initially the boosting algorithm focused on binary classification $c(x) = sin(f(x))$ with response $\bar{y} \in \{-1, 1\}$ where

$$f(x) = \sum_{i=1}^{N} \mathit{\Theta}_i c_i(x) \tag{2.1}$$

and c_i are the base learning models,

The boosting algorithm learns the model by assigning weights $\mathit{\Theta}$ to the data and training the weak classifiers against the weighted data-set. At each iteration, the weights are optimized in a way that the mis-classified data points get higher weights.

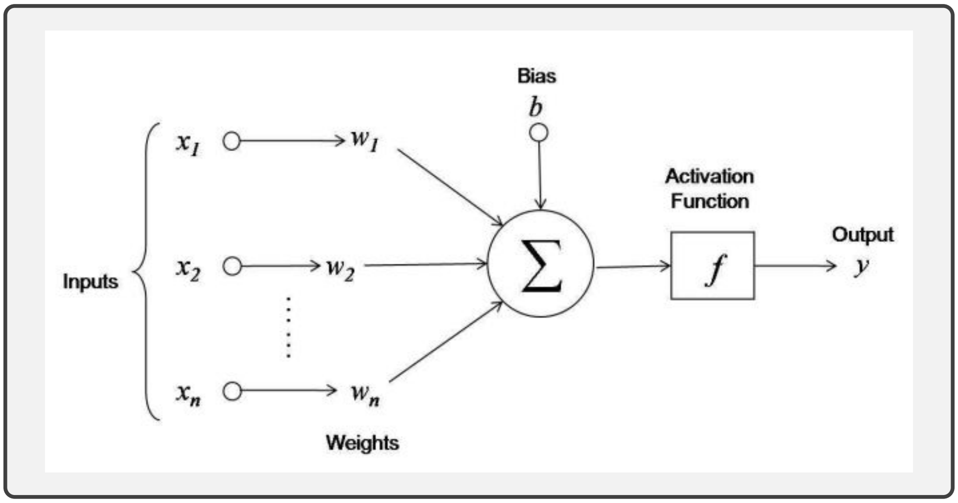
Performance of all these algorithms crucially depends on the features selected to represent the sequence. For example, choosing feature/attributes such as pos-tags proportion, syntactical structure equivalence, and word taxonomy etc. to train a model for STS task would obviously give a better performance than a model trained on only the word count feature. The extraction of each attribute takes a long time and finding the interactions between this feature, specifically for a task further slows down the training process. Most of the time, the model is provided with n incomplete or over specified feature set. If an algorithm can learn the attributes by itself, the training process can be automated more efficiently, and it helps in solving many NLP tasks. Neural networks are one such algorithm which can learn the feature on it own.

Since 1980, various neural models have been proposed. Initially, neural models did not perform well in any task as they learn the feature by itself and require an enormous amount of data to generalize for unseen data. Recent insights in optimization, advances in parallel computing, and the availability of large datasets encouraged these architectures to achieve state of the art performance.

2.3.2 Neural Networks

A neural network is a directed graph with neurons as its node. Neurons are computational units connected by directed links. Each link has a weight that determines its importance or strength. Similarly, all the computational units consist of activation functions that are applied to the input. The activation function can be any linear or non-linear function such as sigmoid (σ), hyperbolic tangent (tanh), and rectified linear units (RELU) etc. So, an output of any computational units is a function over the sum of the weighted inputs. Consider a computational units with activation function f that takes inputs $x_1, x_2, ..., x_n$ with weights $w_1, w_2, ..., w_n$ and bias b as shown in Figure 2.5. It is formally expressed as:

Figure 2.5: A simple neuron



$$z = \sum_{i=1}^{n} wx_i + b \tag{2.2}$$

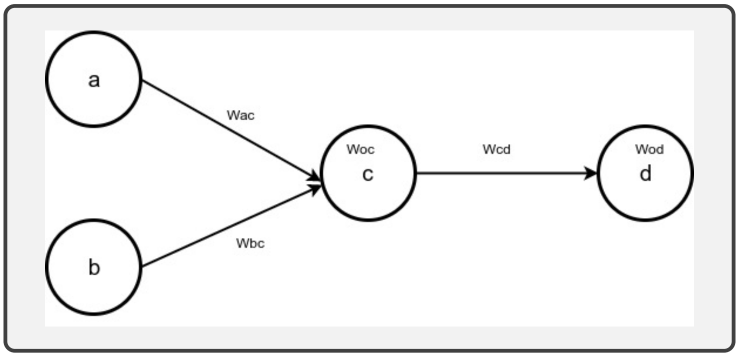
$$y = f(z) \tag{2.3}$$

A simple neural network with two inputs a and b, one hidden unit c, and an output unit d, is visualized as shown in Figure 2.6. Consider a sigmoidal function as the activation function in node c and d. This network can be trained by optimising its weights $[W_{ac}, W_{bc}, W_{cd}]$ based on an objective function. The training has two phases: forward pass and backward pass.

Forward Pass

For node c, in_c is computed from the given weights and the input. The in_c is fed into the activation function to get the output A_c. The output A_d of Node d is computed in the same

Figure 2.6: Neural Network



way.

$$in_c = W_{ac} \times a + W_{bc} \times b + W_{oc} \tag{2.4}$$

$$A_c = \frac{1}{1 + exp(-in_c)} \tag{2.5}$$

$$in_d = W_{cd} \times A_c + W_{od} \tag{2.6}$$

$$A_d = \frac{1}{1 + exp(-in_d)} \tag{2.7}$$

Backward Pass for weights adjustments

The total loss of the networks is computed using the objective function $L = \frac{1}{2}(y - A_d)^2$, where y is an actual output and A_d is the predicted output. The gradient of loss L is calculated concerning all the weights.

For example, gradient of loss computed with respect to W_{cd},

$$
\begin{aligned}
\frac{\partial L}{\partial W_{cd}} &= \frac{\partial L}{\partial A_d} \times \frac{\partial A_d}{\partial in_d} \times \frac{\partial in_d}{\partial W_{cd}} \\
&= \delta_d \times \frac{\partial in_d}{\partial W_{cd}} \\
&= \delta_d \times \frac{\partial (W_{cd} \times A_c + W_{od})}{\partial W_{cd}}
\end{aligned}
$$

$$
= \delta_d \times A_c \tag{2.8}
$$

where δ_d is a modification error. Weights are updated based on the gradients as shown below.

$$
W_{c,d} \leftarrow W_{c,d} + \alpha \times A_c \times \delta_d \tag{2.9}
$$

where δ_d is a modification error,

If there is more than one output unit in the layer, the partial derivative of the error across all of the output units, is equal to the sum of the partial derivatives of the error concerning each of the output units.

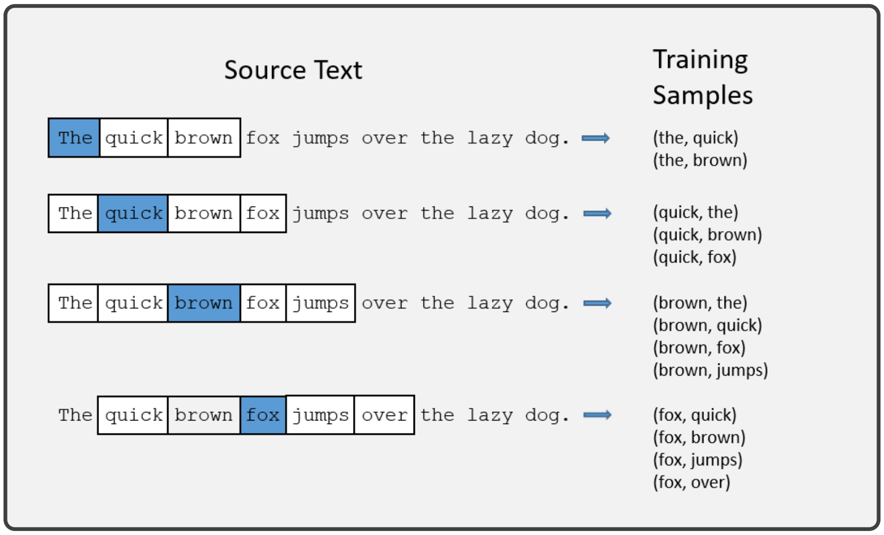
A simple feedforward neural accepts fixed length input, and the length of the sentences are varying. To feed a sequence as an input, the vector representations of each word in the sentence should be transformed using additional operations such as average or sum. By changing the sequence input, it is impossible to capture word order information. Recurrent neural networks handle this case by employing recursive computational units as per the sequence length.

2.4 Word Vector Representation

Firth's hypothesis of representing each word meaning by using its nearby words, was successfully used in many statistical NLP techniques. For instance, Brown Clustering (Brown and Huntley, 1992) , Latent Dirichlet Allocation (Blei et al., 2003) etc used word co-occurrence count as its primary component. In the field of deep learning, Bengio et al. (2003) proposed a language model based on a neural network that predicts the next word for the given previous words in a sequence. They also noticed that this prediction model helped in learning a vector representation for words called word embeddings or word representations. Later in 2008, Collobert and Weston (2008) showed the usefulness of word representations in various downstream NLP tasks. Inspired by these models, Mikolov et al. (2013) proposed two novel methods: 1) Skip-Gram with Negative Sampling (SGNS); 2) Continous Bag of Words(CBOW) models for learning word representations. The SGNS model is discussed in detail below, as it is widely adopted and used as an input for many sentence representations model (Kiros et al., 2015).

The SGNS model was trained on a monolingual text corpus, where every centre word is used to predict the surrounding words within a window size as shown in Figure 2.7. The model takes word input in the form of the One-Hot vector of dimension $1 \times |V|$. While building the vocabulary from the training corpus, One-Hot vector $1 \times |V|$ gets assigned to each word consisting of value 1 in the position that is the same as the position of the word in the vocabulary, and value 0 in all other positions. This input vector X is multiplied with the word matrix W, to get the hidden layer v (target word representation) of dimension $1 \times |D|$,

Figure 2.7: Training samples for skip-gram - Words pairs from raw corpus

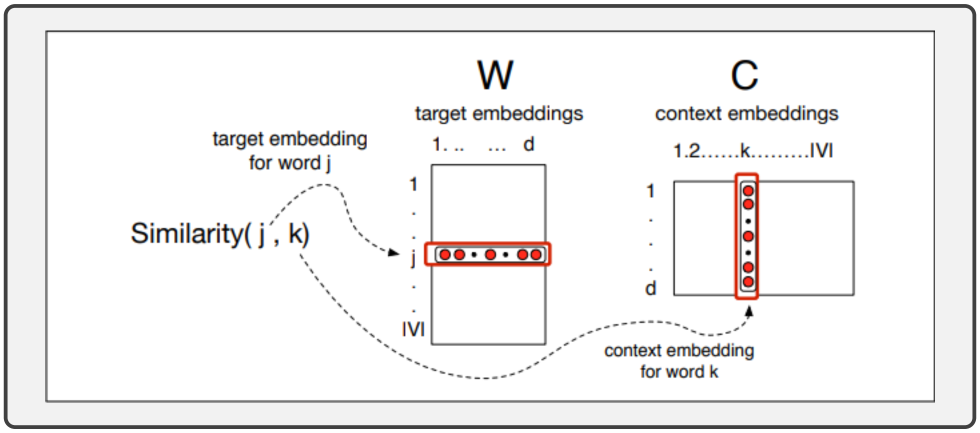


where D is the dimension of a word representation. The dot product of the hidden layer and the context matrix is to find the context word score as shown in Figure 2.8. For each word, the context word score is normalized using soft-max function, to get the probability of each word in the vocabulary occurring near the given the word. For a word w_j, the probability of any k^{th} word in V is calculated as shown in equation 2.10

$$p(w_k|w_j) = y_k = \frac{\exp(c_k \cdot v_j)}{\sum_{i=1}^{|v|} \exp(c_i \cdot v_j)} \tag{2.10}$$

The output of the network is the vector $1 \times |V|$ consisting of the probability distribution of each word in the whole vocabulary. Each probability value in the vector position denotes the likelihood of a word corresponding to the same position in the vocabulary. Cost function

Figure 2.8: Context and Word matrices (Jurafsky and Martin, 2014)

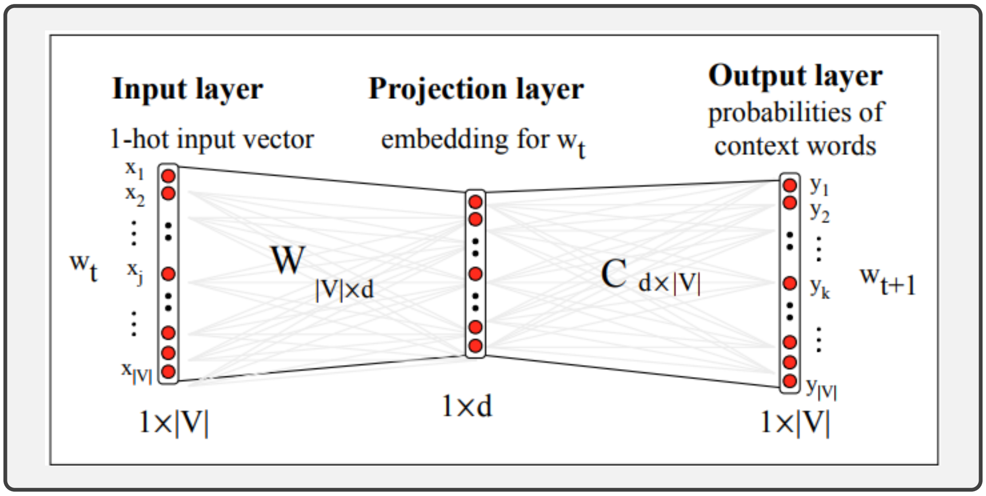


L tunes the weight parameters in this network by using equation 2.11

$$L = -\log(p(w_{O,1}, wO, 2, ..., w0, C|wI)) \tag{2.11}$$

The derivative of L with respect to input units of the output layer and hidden layer, constitute the prediction error. The learning algorithm is started with the randomized word and context matrices (weight parameters of this network). Optimizer algorithms such as Stochastic Gradient Descent are used to tune the weight parameters using error back-propagation to broadcast the gradient through the network. Later, the Skip-Gram model was improved by replacing the soft-max function in the output layer with Negative Sampling. This model provided state of the art performance while testing for semantic and syntactic word similarities. (Mikolov et al., 2013).

Figure 2.9: Skip-gram Model (Jurafsky and Martin, 2014)



2.5 Sentence Representation Model

As word representation models became very useful and predominantly used, a natural next step was to extend such approaches to build sentence encoders. The main objective of sentence encoders was to learn an accurate sentence representation that would capture its semantics using previously trained word representation. The vector representations can be learned by using two approaches 1) Supervised models (Conneau et al., 2017) 2) Unsupervised models (Kiros et al., 2015). Skip-gram model is unsupervised learning as its objective function was to optimise the word representations. In this type of learning, general information is captured. The knowledge obtained by these models can be transferred to any NLP task that needs to process a sequence.

Supervised learning of vector representations is training a model on a pre-defined task, with defined input-output samples where the model parameters are optimised based on the task. The word representations are learned in internal states as the model is trained on an STS task. In this type of learning, the model learns the information that is specific to the task and fails to capture general knowledge. All proposed models aim to generate an accurate and generic sentence representation that consists of the whole meaning of the context. These generic representations are vital as they can be used for various tasks with minimal adaptation. This property results in the smooth transfer of learning to the model that learns any specific task. By promoting Transfer Learning, the training complexity and training time for particular tasks, decreases.

2.6 Summary

This chapter present the technical background of Natural language understanding through word representations to sentence representations.This chapter also discussed about the successful word representation model in details. This model was used as a base component to develop other word and sentence representation models. Next Chapter discuss about the exiting models proposed for extracting sentence representation.

Chapter 3

Related Work

Despite developing a number of learning algorithms for representing words and sentences, generating high quality and efficient sentence representations remains an unsolved problem (Conneau et al., 2017). An efficient sentence representation that consists of the whole meaning with context is important as it can be used across various tasks with minimal adaptation. This property results in a smooth transfer of the learning to train any NLP task. We focus on the STS task, as Conneau et al. (2017) demonstrated that natural language inference tasks appear to capture more generalizable sentence representation using Transfer learning. This chapter discusses the existing models and comparison studies in order to compare and study different models used for STS tasks and to infer how they impact in learning good sentence representations.

In this chapter, we discuss about the existing sentence representation encoder models and theirs usefulness. Section 3.1 presents the classical machine learning models proposed for STS task and the features proposed in their systems to represent semantics. In section 3.2, we discuss about the neural models which gave the best performance in last five years and the comparison studies on these models.

3.1 Traditional Machine Learning Models

For decades, traditional machine learning algorithms such as support vector machine or logistic regression were to used to solve any NLP tasks. In 2012, supervised models based on the lexical and syntactic features of the sentence pair, showed promising results on measuring semantic relatedness. These systems gave 52% - 59% correlation on various data-sets by using regression models consisting of various similarity measures as its input features. The unsupervised models did well for next two years in row using the WordNet knowledge and the LSA similarity measures which assume that the words with closer meaning highly co-occur in the text corpora.

Han et al. (2013) proposed three approaches that involved 1) LSA similarity model, 2) semantic similarity model based on the alignments quality of the sentences, and 3) support vector regression model that had features from different combinations of similarity measures, and the measures from two other core models. It was observed that using the n-gram overlap feature increased LSA similarity model. Out of three models proposed by Han et al. (2013), the alignment based system gave 59% - 74.6% Pearson Correlation on four different data-sets. Using this model's alignment quality as one of the features in the Support Vector Regression model improved the correlation score to 78 %. Various supervised models using uni-gram (one word) or bi-gram (two words) overlap, vector distance, and cosine similarity of sentence embedding, were proposed (Agirre et al., 2015).

Tian et al. (2017) proposed a system that adapted ensemble learning techniques to solve the Textual Entailment and STS tasks, using the same set of features. The combination of classical NLP models like Support Vector Machine, Random Forest, Gradient Boost and a deep learning model are used in this system. For classical NLP models, single sentence and sentence pairs feature sets, are hand engineered based on properties like N-gram overlap, syntax, alignments, word sequence, word dependency, word representations etc.. In SEM-EVAL 2017, this mixed ensemble model gave 81 % Pearson Correlation, outperforming all the neural models presented in that shared-task event.

Although using hand-crafted features in the above mentioned models works, it has some drawbacks such as tuning the features extracted on addressing the corpus from new domains, high computational complexity in hand engineering the features, and effective feature selection etc.. Recent approaches in deep learning continue to prove that the problem of semantic text matching can be handled in an efficient way (Cer et al., 2017). The problem of semantic word matching can be extended to solve the problem of the semantic sentence match by using deep learning approaches. This helps when it comes to effectively learning the word meanings in the sentence individually and deriving a meaningful sentence representation from the word vectors.

3.2 Neural Models

This section discusses the top ranking neural models presented in Sem-Eval 2017 that have been proposed to build sentence representations and predict sentence relatedness.

Kiros et al. (2015) proposed the Skip-Thought model based on skip-gram objective from Mikolov et al. (2013). For any three consecutive sentences in the document S_{i-1}, S_i, S_{i+1}, the Skip-Thought model predicts the previous sentence S_{i-1}, and next sentence S_{i+1} given any sentence S_i. This work focuses on training an encoder-decoder model. A variant of recurrent networks consisting of gated recurrent units (GRU) (Cho et al., 2014) is used as an encoder to map input sentences into a generic sentence representation. RNN with conditioned GRU is used as a language model to decode the sentence representation and predict surrounding sentences S_{i-1} and S_{i+1}. In evaluating a semantic relatedness task, Skip-Thought outperformed all systems proposed in a shared task SemEval 2014 (Marelli et al., 2014).

Tai et al. (2015) proposed a recurrent neural networks(RNN) with tree based LSTM units with two variants Child-Sum Tree-LSTM and N-ary Tree LSTM. Given a sentence syntactic structure in a form dependency tree of the words, Tree-LSTM networks are capable of integrating the child node's information. The Tree-LSTM units in each node t consists of input gate i_t, output gate o_t, a cell unit c_t and a hidden output h_t. Unlike Standard LSTM, the parent node has one forget gate f_{tk} for each child node *k* in the Tree-LSTM. This property allows selective usage of child information. Previously proposed RNN models with sequential LSTM units, have limited ability to capture the meaning difference in the two sentences raised due to word order and synactical structures. Tree-LSTM addresses this issue by computing its hidden layer output as a function of the outputs from its children hidden units and input vector.

In modelling semantic relatedness, the input x_t denotes the word vectors of the sentence parse tree. The proposed model retains the information of more distant words from the current words compared to other existing models. These properties make the model effective in highlighting the semantic heads in the sentence. It also captures the relatedness of two phrases which have no word overlap. With these properties, Tree LSTM performs better than existing sequential RNN-LSTM models, and models with hand engineered features on predicting the semantic relatedness of two sentences. But one major downside is that the dependency tree-LSTM relies on parser for dependency tree input, which is computationally expensive to collect and does not exist for all languages making it inefficient in cross-lingual sentence representations.

Shao (2017) presented a simple Convolutional neural network model for STS tasks. This model constsis of CNN model and fully connected neural network (FCNN). CNN takes pre-trained word vectors from Glove Pennington et al. (2014) enhanced with hand-crafted features as its input. It enhances word vector to task specific forms in the convolutional layer, and max-pooling generates the task-dependent sentence representation. FCNN generates the similarity score ranging from 0-5. This model ranked 3rd in SemEval-2017 with a 78 % correlation on STS tasks.

Pagliardini et al. (2017) proposed a simple unsupervised objective Sent2Vec, to train a generic distributed representation for sentences. The main contribution of Sent2Vec is its low computational cost for both training and inference, relative to other existing state-of-art models. This model is an extension of CBOW training objective from Word2Vec (Mikolov et al., 2013), to sentence context.

Conneau et al. (2017) investigated the performance of various supervised encoders in learning universal sentence representations. They hypothesized that textual entailment task is a good choice for learning universal representations and demonstrated the hypothesis with various encoder models. To prove that the sentence representations learned are universal, the representations learned from unsupervised and proposed hypothesis was used in 12 different transfer tasks, such as Caption-Image retrieval, Paraphrase detection, Entailment/semantic relatedness, and sentiment analysis etc.. As the result of their experiments, Bi-LSTM with max-pooling trained on Natural Language Inference Task (Textual Entailment), generated the best sentence representations, and outperformed SkipThought Kiros et al. (2015) and FastSent Hill et al. (2016).

3.3 Summary

This chapter has reviewed the pro and cons of the best performing models proposed in last five years. The research in semantics extraction is progressing using both classical machine learning and neural networks model. The models that took advantage of both classical machine learning and neural models using ensemble techniques showed good performance. The neural models that used enhanced word representation by concatenating the actual word representation and the alignment features of two sentence has also showed promising results. Next chapter discusses about the proposed study in this project.

Chapter 4

Proposed Work

In recent times, a wide variety of encoders for learning sentence representation have been proposed by NLP researchers. However, there is a lack of understanding about the characteristics of different encoding techniques that can capture useful, accurate semantic information (Conneau et al., 2017). In feature based machine learning models, hand crafting and selecting optimal features is hard. Although neural models learn the feature by itself, they suffer from an inherent bias toward the task and data-set that they are trained on. This feature is a downside because it learns the task very well and fails to capture generic useful information during the training time, leading to poor generalization. On the contrary, neural models trained independent of any task give more importance to general information. But, it fails to specialize the model for any specific task. Many factors affect how the basic semantics of a sentence are being captured during training.

In this project, we perform a systematic comparison of different encoder techniques and assess their ability to capture semantics of the sentence, based on their performance in STS tasks. To investigate the performance, various models such as support vector machine (SVM), Random Forest (RF), Convolutional Neural Network Encoder (Shao, 2017) and Bi-LSTM RNN with max-pooling (Conneau et al., 2017) were implemented. These encoder models achieved good accuracy in STS task, and it outperformed the state-of-the-art encoders in SemEval 2017 conference (Cer et al., 2017). Conneau et al. (2017) demonstrated that Recognizing Textual Entailment (RTE) task trained using Standford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), captures semantics very well. Based on this inference, the neural models were trained using SNLI data-set, and Sentences Involving Compositional Knowledge (SICK) corpus (Marelli et al., 2014), for semantic relatedness and RTE tasks.

Similarly, the encoders' architecture for both task dependent and independent neural models also impacts learning in different ways. This comparison study on these encoder's architecture and performance on STS task primarily focus on understanding for following:

- Encoder's ability to capture semantics.
- Encoder's potential to capture accurate meaning representations that is generic.

Above two objectives of this comparison study answer the following question that helps in improving the models that already constitute the state of the art in sentence encoding.

- Which encoder techniques perform well in learning accurate sentence representations ?

- Among classical machine learning models, what are the features contribute more to prediction ?
- What are the optimal hyper-parameters of the encoder model that helps in better performance ?
- Since the dimension has direct effect on the memory requirements and processing time, where dimension has a good trade-off between accuracy and training time ?
- What is the preferable neural network architecture for learning generic sentence representations that can aid in transfer learning ?

Learning Methods

This section describes the architecture of a various learning methods that are investigated for its ability in extracting sentence representation.

4.1 Ensemble Model

This section discusses about the architecture details of an ensemble model proposed by Tian et al. (2017). This model adapts a combination method to intergrate the prediction of traditional ML algorithms such as support vector machine (SVM), Random Forest (RF), Gradiant Boosting (GB) and a neural network algorithm such as deep averaging network. This project explores the combined performance of SVM , RF and GB as mentioned in

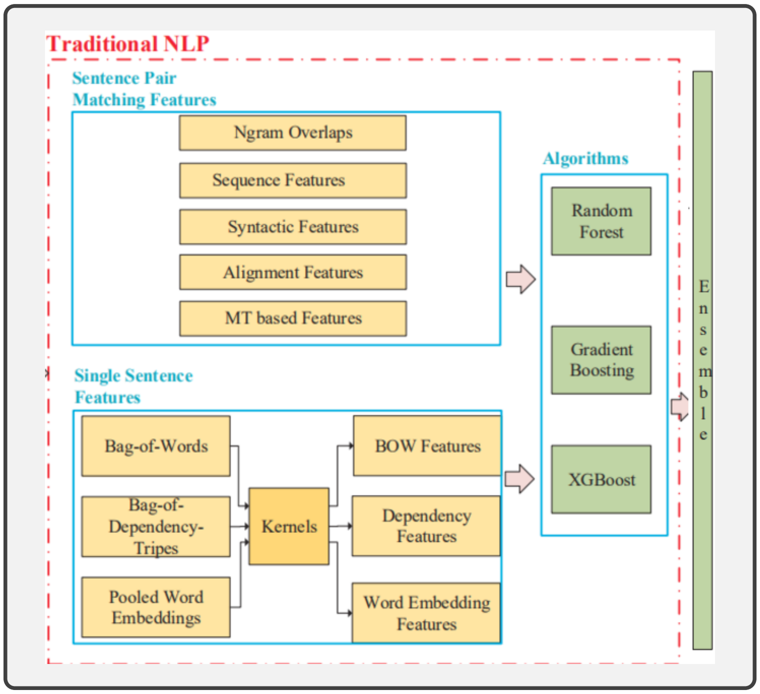
(Tian et al., 2017). Given *n* training data with input and output $\{(x_i, y_i)\}_{i=1}^{n}$, these models focus on estimating a function (hypothesis) *h(x)=y* that maps input to output. In STS task, with sentence pair as the input, a standard approach is to represent the sentence pair in the form of similarity features. Therefore, effective feature representations that capture semantic and synactic matching degree are hand engineered.

With these feature representations as input, the predictions are done based on the hypothesis *h(x) = W . X* where X is the similarity feature vector and W is the corresponding weight vector. The model learns an appropriate weight for each feature while training. In Ensemble approach, various models trained on sentence relatedness task with continous outputs, are optimizied based on the average of the score returned by them. In the case of classification tasks such as RTE, the models are optimised based on the majority voting of their classifications. The architecture diagram of this model is shown in Figure 4.1.

4.1.1 Features

For this ensemble model, the features of a sentence pair is extracted based on the length of two sentences, n-gram overlap, syntactic structure, alignment and machine translation metrics. N-Gram represents the group of *n* consecutive character/words/sequence in a corpus. For the STS task, the sequence of words or phrase overlaps are useful in expressing the common expression between two sentences. In this project, the normalised n-gram overlap is extracted in both word and character level. Similarly the longest common sub-sequence, prefix and suffix are computed to extract the sequeunce similarity. To estimate accurate

Figure 4.1: Ensemble Model (Tian et al., 2017)



similarity, the words are lemmatized where the words are replaced with its root word.

Although the sequence of words overlap information can give a good indication of similarity, it fails to capture the word dependencies in a sentence which primarily influence the sentence meaning. The syntactical structure of a sentence is captured in the form of tree and the number of common subtree, constitutes the feature set. Further, the monoligual word alignment proportion is extracted as a feature. This feature maps the words between two sentences based on their meaning, parts of speech tag, and the synactic structure. The alignment features include normal alignment proportion, pos tag based weighted proportion. Other features include various WordNet based and word vector representation based similarities measures, such as

- **Levenshtein distance** : It signifies the number of minimum edits required to convert one sequence to the other. This is also known as Path distance. This feature implies the sequence is more similar when the distance is shorter.

- **Leacock-Chodorow distance** : This feature extends the path distance by scaling it using depth of hierarchy structure based on *is-a* relationship

- **Resnik distance** : This feature measures the similarity based on taxonomy information of two words/sequence. The similarity is computed based on the distance of first common predecesor of the two words/sequence in WordNet's taxonomy tree which conveys how much two words are related.

- **Jiang-Conrath distance** : This feature is the measure of increase difference in relation of two words/sequence.

- **Cosine distance** : Given the vector representation of words, normalised dot product of two vectors indicates the similarity between them.

Finally, a sentence pair is represented using 47 features. These features are standardised to [0,1] using max-min normalization to reduce the standard deviation and handle the outliers in the features. Then, the traditional machine learning models such as Support vector machine (SVM), Random Forest(RF), and Gradient Boosting (GB) are trained using these features.

4.1.2 Objective Function

For semantic relatedness task, a list of sentence pairs $X = \{(S_{a1}, S_{b1})$,.., $(S_{aN}, S_{bN})\}$ is given as input. The sentence pairs come with similarity scores $Y = \{Y_{ab1}, Y_{ab1}, ..., Y_{abN}\}$ that consist of value ranging from 0 indicating no similarity to 5 indicating the high similarity between the sentences. The goal is to build a model that is able to produce the correct similarity score Y_{ab} for each sentence pair (S_{ai}, S_{bi}).

Formally, the task to learn is represented as,

$$h(w, f(S_a, S_b)) \rightarrow Y_{ab} \tag{4.1}$$

where function f maps sentence pairs to a vector representation, in which each dimension expresses a certain type of similarity between the input sentence pair such as lexical, syntactic, semantic etc. The weight vector, w is a parameter of the model that is learned during the training and h denotes the STS prediction model.

The RTE task is a classification problem that consists of three target clasess *C* =\{*entailment, contradiction, neutral*\}. A classifier function γ is learned to map the sentence pair to its corresponding RTE class labels *C*.

In ensemble of SVM, RF and GB, their predictions are combined into one final prediction using the Stacking algorithm. In stacking, the training set is split to several subsets and each model is trained and tested on one of those subsets. Finally the predictions are fed into an outer model with their actual target value for its training. This outer classifier combines the prediction of SVM, RF and GB.

4.2 Convoutional Neural Network

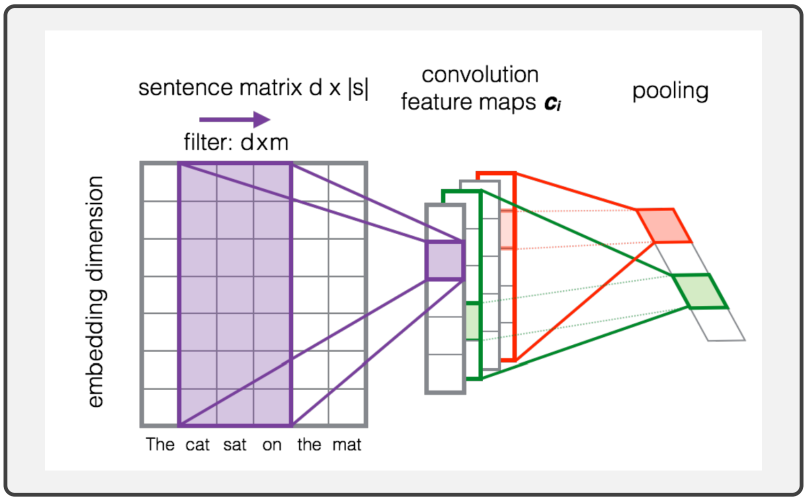
This section explains the convolution neural networks (CNN) based learning model used for semantic sentence similarity. The two main components of this model are (CNN) based sentence representation model, and fully connected neural networks (FCNN) used for STS task. The CNN architecture consists of two convolution networks that work parallel to mapping the two sentences to a vector space. The vectors of the sentence pairs are used by FCNN to classify their sentence similarity score. In the following, we first describe our sentence model for mapping sentence pairs to their intermediate representations and then explain how these representations are used to classify the relatedness score.

4.2.1 Sentence Model using CNN

CNN architecture for mapping sentences to vector representations inspired from Shao (2017) is shown in Figure 4.2. This architecture consists of two 1-dimensional convolution layers and a max pooling layer. The objective of this network is to convert the raw sentence into vector representations from Pennington et al. (2014), using pre-trained 300 dimension word embeddings of all the words $\{w_1, w_2, ..., w_{|s|}\}$ present in the sentence.

The input sentence to the convolution layers is treated as a sequence of a real valued number where the real valued integers are retrieved from the integer-word mapping present in the vocabulary V. The vector representation of all the words $w \in \mathbb{R}^d$ are drawn from embedding matrix $W \in \mathbb{R}^{d \times |V|}$ in the embedding layer. To enhance the word representation with respect to this task, a true flag for word overlap is added as an additional dimension into the word vector representation for each word in the sentence. Then the CNN network applies the convolution and max pooling operation to find the optimal feature vectors for the sentence that capture its semantics.

Figure 4.2: CNN Sentence Model Severyn and Moschitti (2015)



The idea behind the convolution layer is to learn the features which identified the relationship between n-gram of the sentence using weight vectors *m* $\in \mathbb{R}^{|m|}$. The 1×1 weight vector *m* also known as filters of the convolution is used. This convolution operation is followed by applying the Relu activation function to learn non-linear decision boundaries. This filters out the insignificant features learned in previous operation. The output from the convolution layer is passed to the max pooling layer with pool size (1, $|S|$) where the semantic information learned is aggregated, and reduces representation dimension from $1 \times |S| \times 300$ (word vec dimension) to 1×300 (word vec dimension). The convolution layers along with RELU activation function and max pooling acts as a non linear feature detector for the given sentence. The output sentence representation from CNN is used to find the Semantic Difference Matrix by performing a series of operations on the two sentence vector.

Semantic Difference Matrix

The semantic difference matrix is generated by concatenating the vector difference and vector product of a two sentence representation. This matrix is used to classify the similarity measure using fully connected neural network (FCNN) with 2 dense layers.

$$SDV = (|SV_1 - SV_2|.(SV_1 \circ SV_2))$$

Figure 4.3: Hyperparameters for FCNN Shao (2017)

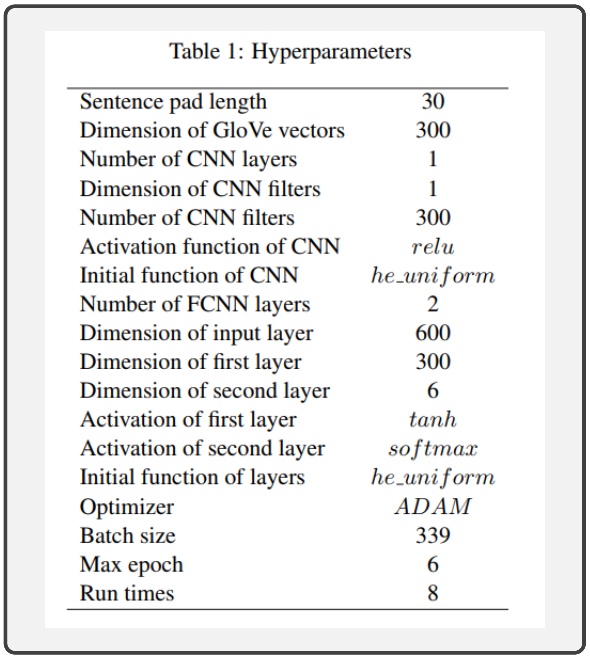


Table 1: Hyperparameters

Sentence pad length	30
Dimension of GloVe vectors	300
Number of CNN layers	1
Dimension of CNN filters	1
Number of CNN filters	300
Activation function of CNN	$relu$
Initial function of CNN	$he\_uniform$
Number of FCNN layers	2
Dimension of input layer	600
Dimension of first layer	300
Dimension of second layer	6
Activation of first layer	$tanh$
Activation of second layer	$softmax$
Initial function of layers	$he\_uniform$
Optimizer	$ADAM$
Batch size	339
Max epoch	6
Run times	8

4.2.2 Similarity Measure using FCNN

This network consists of one hidden layer consisting of 300 computational nodes, and an output layer with 6 nodes. The hidden layer applies a *tanh* activation function and the output layer applies softmax layer. The softmax layer calculates the probability over the six score labels. The hyper parameters of this network is shown in Figure 4.3.

The maximal point is calculated from the probability distribution over six score labels. Finally, the model is trained and optimised using the root mean squared loss of target and predicted continous value.

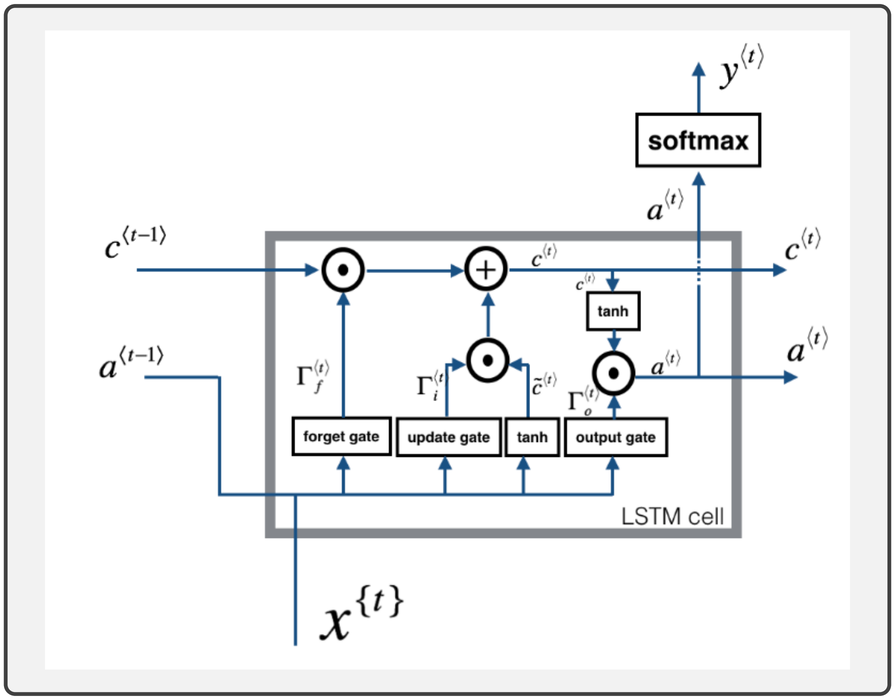
4.3 Recurrent Neural Network

This section desecribes the architecture of InferSent, a variant of recurrent neural network (RNN) proposed for RTE task (Conneau et al., 2017). This model outperformed all the existing state-of-the-art models such as SkipThought, FastSent etc, in extracting accurate sentence embedding. InferSent consists of a bi-directional Long-Short Time Memory (LSTM) based RNN sentence encoder and a fully connected neural network based decoder as shown in Figure 4.5. The architecture of encoder is discussed in the following section.

4.3.1 Bi-LSTM with max pooling

Generally, RNN has the ability to accept input of arbitary length and return a fixed dimensional vector. Also, it is capable of propogating the previously processed input's informa-

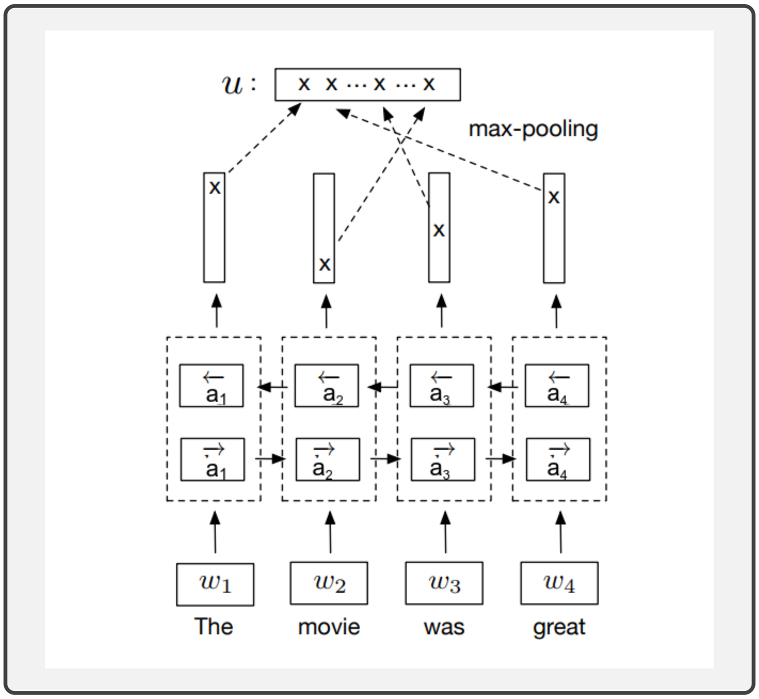
Figure 4.4: A Single LSTM cell (Ng, 2018)



tion while processing a word in the given sequence at timestep *t*. This encoder architecture consists of bi-directional RNN which reads the sentence in two opposite directions. It is capable of remembering the past and future context information of a sequence at any time step. The objective of this network is to extract generic sentence representation for a given word representations of the words present in the sentence.

The RNN consisting of a sequence of LSTM computational cells in this architecture accpets a raw sentence $< x^1, x^2, ..., x^t >\in S$ input in the form word representations. It reads the input S and process one word x^t at a time t sequentially. Each LSTM cells consist of a memory cell C to remember information about previous words and forget, update,output gates to manage memory while processing the words in the sentence. This

Figure 4.5: A Bi-LSTM Network (Conneau et al., 2017)



helps LSTM to keep track of the synactical structure in the case where it stores a gender of the subject and relates it to the pronoun in the later part of the sentence. Each cell accepts current word input x^t , memory cell $\overrightarrow{C^{t-1}}$ from the previous cell state and hidden state $\overrightarrow{a^{t-1}}$. It outputs a fixed dimension hidden state vector $\overrightarrow{a^t}$ and its prediction y^t for current time step t. For this bidirectional RNN, the hidden state output a^t at each time t is computed by concatenating the hidden state output $[\overrightarrow{a^t}, \overleftarrow{a^t}]$ of forward and backward LSTM cells . Max pooling is applied over the varying length hidden state outputs $< a^1, a^2, a^3, ..., a^t >$, to obtain a fixed length sentence representation. In Max pooling, maximum value over each dimension of the hidden states is selected to represent the input sentence.

Semantic Difference Matrix

After extracting the sentence representation $\{u, v\}$ for the sentence pair $\{S_1, S_2\}$, the semantic difference matrix is computed by concatenation, element-wise product and absolute element-wise difference of u and v as shown in equation 5.2 .

$$SDV = ([u, v], |u - v|, (u * v)) \tag{4.2}$$

Textual Entailment Classifier

This semantic difference matrix is fed into a fully connected neural network decoder to classify the textual entailment in the sentence pair. This network consist of a single hidden layer with 512 hidden units and a output layer with 3 output nodes.

The whole encoder decoder architecture is trained and optimised using the categorical cross-entropy as its loss function. With the probabilties from the decoder's softmax layer, one hot vector is calculated by assigning 1 to the class with maximum probability and 0 to others. It is compared to the target one hot vector as its a multi-class classification while calculating loss.

4.4 Summary

In this chapter, the motivation and objective of the comparison study was discussed. Also, this chapter presented the architectural details of the models involved in the comparison study. Chapter 5 outlays the experiments to answer the questions stated in this chapter.

Chapter 5

Experiments and Results

In this chapter, we discuss the results of our comparison study. The goal of our study is to compare the ability of traditional Machine Learning (ML) to neural model, in capturing semantic representation of sentences. All the models are trained for Sentence Relatedness and Recognising Textual Entailment (RTE) tasks. In traditional ML models, handcrafted features are extracted from given a sentence pair, to represent semantics of sentence pairs, used to learn and predict their similarity. In the case of neural models, an encoder learns the sentence representations that allows a decoder to learn and predict their semantic similarity. To perform better in these tasks, the encoder model should effectively capture the compositional meaning of the given sentence.

This chapter is organised as follows. Section 5.1 and 5.2 discusses about evaluation metric and experiment setup. In section 5.3, we evaluate the performance of various sentence encoders in sentence relatedness and RTE tasks. Section 5.4 present the experiments that are performed to tune the hyper-parameter of the encoder architecture and presents the optimal parameter for best performance. Section 5.5 and 5.6 presents the experiments related to feature selection and Transfer Learning.

5.1 Evaluation metric

The performance of the models is evaluated on STS tasks using **accuracy of its prediction**. The accuracy is measured using **Pearson Correlation** between the predicted similarity scores and gold scores created, using humans judgement. It is a measure of the linear relationship between the predicted and target scores. The correlation value ranges from -1 to 1 where, 1 indicates perfect positive correlation, 0 indicates no correlation and -1 indicates negative correlation.

Our second metric is **training time**. It is used to infer the trade-offs between the model's performance and the time taken for it to converge to the global optimum. Although complex models give better accuracy, they take a long time to train. The objective aim of any ML algorithm is to minimise the loss in the predicted value when compared to target value. Analysing the time taken for loss of the encoder models to converge to its global minima, can help in finding a trade-off between the training time and the model performance.

The models are trained until the loss converges to its global minima. At that point the model becomes optimal. Further optimisation leads to no improvement. Various hyper-parameters impact the time taken by a model to converge. Sometimes, the learning algorithm consistently learns unnecessary features, because of the randomness in the training data points. In this case, the loss is always high and the model is said to have a high bias. This model doesn't converge to the global minima resulting in underfitting. In contrast, it starts to memorise the training data which is known as overfitting. At this point, the model's prediction on unseen data is comparatively poor.

In addition, the sentence representations encoders are evaluated for their performance on generating a generic sentence representation that aids in **transfer learning**. This evaluation is carried out by using representations learned from training RTE as input in sentence relatedness tasks. Generic representations are expected to perform well in both tasks in terms of accuracy. This helps to train a model for the task that does not have sufficient resources.

5.2 Experiment Setup

In this chapter, we evaluate a variety of sentence encoding models, including an Ensemble model (based on combination of SVM, RF, GB), a simple Convolutional Neural Network model , and a Bi-Directional LSTM based Recurrent Neural Network model. To make these evaluations more descriptive, the experimental setup of each models architecture, including its default hyper-parameters, are outlined in this section. Any modification in hyper-parameters are specified in each experiment.

Table 5.1 shows the parameter settings of the classical ML models. All these models are trained until its loss converges to its global minimum. All the neural models were trained using Stochastic Gradient Descent (SGD) optimiser with 0.1 learning rate. CNN encoder was trained with a batch size of 336, and Bi-LSTM RNN was trained with a batch size of 64. For RTE classifier, we used a fully connected neural network with one hidden layer of 512 hidden units, and a final softmax layer of 3 outputs with a **Categorical Cross Entropy** loss function. For Sentence Relatedness decoder, we used a fully connected neural network with one hidden layer of 512 hidden unit and an output layer with a softmax function consisting of 6 outputs. The probability distribution was transformed to a single score in final layer. This network used a **Mean Squared Error** loss function for optimizing the model weights.

Table 5.1: Ensemble Model - Parameter Setup

Algorithms	**Sentence Relatedness** **(Regression Problem)**	**RTE** **(Classification Problem)**
	Bounded output : 0 - 5	3 Class Labels : [0,1,2]
SVM	Support Vector Classifier (Kernel: Polynomial)	Support Vector Regressor (Kernel : Polynomial)
Random Forest	RFClassifier (estimators = 200)	RFRegressor (estimators = 200)
Gradient Boosting	GBClassifier (estimators = 200)	GBRegressor (estimators = 200)
Ensemble performed	Voting based classifier	Stacking Model (meta-model : Linear Regression)

We trained and evaluated the sentence encoder models using Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) for RTE task and Sentences Involving Compositional Knowledge (SICK) (Marelli et al., 2014) data-set for sentence relatedness task.

The ensemble model is implemented using SciKit-Learn library. The data pre-processing are performed utilizing NLTK-toolkit; a python-based library (Bird and Loper, 2004). The pre-processing steps includes stop-words removal, replacing words with its root words, and extracting similarity measures, mentioned in section 4.1. The features based on alignment and syntactical tree structure properties are created using Standford-coreNLP (Manning et al., 2014).

The neural models are implemented using Keras[1] and PyTorch[2]. Keras is a python-based high-level neural networks API that runs on top of TensorFlow[3]. PyTorch is a python package that faciliates to building dynamic neural networks architecture. They were trained using a machine with 8 × CPUs, 32GB RAM and 1 × NVIDIA Tesla P100 GPU from Compute Canada [4] to perform the comparison study. Tesla P100 GPU has a computing speed of ~ 4.7 teraflops [5] whereas the Core i7 CPU-7700 has a computing speed of $\sim$ 100 gigaflops.

[1] https://keras.io/

[2] http://pytorch.org

[3] https://github.com/tensorflow/tensorflow

[4] https://www.computecanada.ca/

[5] http://www.nvidia.ca/object/tesla-p100.html

5.3 Encoder Architectures - Performance

In this section, the performance of the encoder models on Sentence Relatedness and RTE tasks are explored. The encoder is trained on Sentence Relatedness task using 20k sentence pair (Cer et al., 2017) and RTE task using 100k sentence pair (Bowman et al., 2015). As mentioned in Chapter 2, the data-set used for this experiment is a collection of data from various domains. The trained models are tested with sentence pairs that were not encountered by them while training. For this experiment, each the models were run with its optimal hyper-parameter, to explore its best performance in both the tasks.

5.3.1 Analysis

Table 5.3 and 5.2 shows the test performance of various models trained on RTE and Sentence Relatedness tasks. We found that the Bi-LSTM RNN with the max pooling model, has the best performance in both the tasks. This model is capable of parsing sentences in both directions and maintains a dynamic knowledge base using LSTM units. This enables it to capture the compositional semantics of a sentence. For RTE tasks, there is no significant accuracy difference between the voting based ensemble and GB. For Sentence Relatedness task, the GB does slightly better than the stacking ensemble model.

Among classical ML algorithms, Gradient Boosting (GB) performs better than random forest. This could be because the Random Forest(RF) reduces the error by lowering its variance to handle over-fitting, and the bias in prediction caused due to under-fitting stays fixed. In GB, weights are optimised in such a way that both bias and variance are minimised. This

Table 5.2: Peformance of Classical ML models

Algorithms		**RTE (Test)**	**Sentence Relatedness (Test)**
Single ML Model	Support Vector Machine (SVM)	75.2	76.2
	Random Forest (RF)	75.4	77.5
	Gradient Boosting (GB)	76.7	77.5
Ensemble	SVM + RF + GB	76.9	75.4

Table 5.3: Performance of Neural Model in RTE and Sentence Relatednes Task

Algorithms	**RTE**		**Sentence Relatedness**	
	Train	Test	Train	Test
CNN	71.6	61.3	69.6	61.4
Hierarchical CNN	73	79.84	-	-
Bi-LSTM RNN	83.98	84.35	79.9	78.5

reduces the possibility of a model to under-fit or over-fit to the training data.

The convolutional network did not perform well in either of the tasks. It was observed that the testing accuracy of the CNN model was very low compared to the training accuracy, meaning that the model is over fitting the data. The CNN model architecture was updated such that three convolution layers followed by a max pooling layer was added, to investigate if there is any improvement in the performance. Inspired from self-hierarchical sentence model (Zhao et al., 2015), the final sentence representation was derived from concatenating the output from all the max-pooling layers. The hierarchical CNN showed better performance than single layer CNN model and all classical ML algorithms.

5.4 Hyper-parameter Tuning

This experiment aims to find the optimal hyper-parameters of the neural model's architecture that can contribute to better performance. The hyper-parameters are the external configuration of a model that helps in estimating the weight and bias variables of the model. For this experiment, we studied the following hyper-parameters: optimizers, dropout rates, sentence representation dimensions, and learning rates. The updates to the weights and the biases of the networks are heavily dependant on the hyper-parameters.

There are various optimizer functions that help in optimizing the network's learnable parameter. In this experiment, we compare a non-adaptive optimizer such as Stochastic Gradient Descent (SGD) with an adaptive one such as Adam. Both these optimizers focus on minimizing the loss of the model. The learning rate determines the step size that is taken by the optimizer to reach the global minima, and dropouts are used to overcome the over-fitting issue by ignoring the randomly chosen hidden units.

5.4.1 Analysis

Table 5.4, and Table 5.5 report the results of our experiments on optimal hyper parameters. These experiments were performed on our best performing model- BiLSTM RNN with the

Table 5.4: Learning rate - optimizers

Optimizer	Learning Rate	Epochs	RTE Task	
			dev	test
Adam	0.1	4	57.2	57.15
SGD	0.1	8	83.98	**84.35**
	0.01	13	83.49	83.28
	0.001	21	78.17	77.86

max pooling layer. For this model, the network did not converge at all when the Adam optimizer was used. The training was stopped in 4 epochs as the loss of the network did not decrease. But, the same network converged to optimal solution when we used SGD as reported by Conneau et al. (2017).

For SGD, the network was trained with three different learning rates *[0.1,0.01,0.001]*, two dropout rates *[0.2,0.5]* for 21 epochs each. When using 0.1 learning rate, the network achieved its maximum accuracy within 8 epochs. Lower learning rates took longer time to converge and did not achieve global optima as shown in Figure 5.4. For dropouts, no difference was found in output performance under both the settings as the model was not over-fitting.

Table 5.5 presents the effects of using different dimensions for sentence representations. For all the models in this table, SGD optimizer with 0.1 learning rate was used. We find interesting trade-offs between accuracy and training time. When a large dimension is used, the training time rapidly increases as the number of parameters of the network increase and the model complexity increases. Hence, the model is able to capture the linguistics features of the sentences better and produces 84.35% accuracy on test data. As the dimension of the sentence representation decreases, the accuracy and time taken to train the model also decreases.

Table 5.5: Sentence Representation Dimension

Algorithms		**RTE**		
	Dimension	**Converge Epoch**	**Total training time**	**Test**
Bi-LSTM RNN	500	20	$\sim$ 400 mins	83.12
	1000	8	$\sim$ 360 mins	83.97
	1500	6	$\sim$ 360 mins	84.17
	2048	6	$\sim$ 540 mins	**84.35**

5.5 Input features for Traditional ML

In this section, the importance of handcrafted features and their impact on the performance of ML techniques are studied. Table 5.6 reports the performance of the Ensemble model with different input features. We chose the ensemble model as it was the best performing traditional ML algorithm. Unlike neural network based models, the performance of these algorithms heavily depends on hand-crafted features.

Table 5.6: Performance of the Ensemble Model with different Input features on RTE task

Input Features	**RTE** **Test accuracy**
Ngram Overlap	63.8
Sequence Features	47.4
Similarity scores	70.6
Alignment Features	66.7
All Features	76.0

As discussed in Section 4.1.1, the input features used in this model are categorised as : n-gram overlap, sequence features, word representation features and word alignment features. We observe highest accuracy when the model is given all categories of features, indicating that all features contribute to the prediction.

Among the different categories of features, we find that similarity scores based on the word representations of input sentences performs the best with 70.6% test accuracy. This is because we use neural-network based pre-trained word representations, that extracts semantics at word level. Sequence features, such as longest common sub-sequence, prefix and suffix do not contribute much to the overall performance. This can be due to fact that semantically similar sentences may have no sequence overlap.

5.6 Transfer Learning

This section reports on the ability of Bi-LSTM RNN and CNN models to aid in transfer learning. Conneau et al. (2017) showed that models trained on SNLI dataset captures universal representations of sentences that can be transferred to other tasks. Convolutional neural networks have achieved outstanding success in tranfer learning on ImageNet classification. We train Hierarchical CNN model and Bi-LSTM RNN with max pooling on SNLI dataset for RTE task and evaluate the encoder performance on sentence relatedness task. Here RTE is the source task and the sentence relatedness is the target task. In this experiment, the dataset of source task does not overlap with the dataset of the target task to measure the generalizability of the trained model. We use STS benchmark dataset (Cer et al., 2017) for target task. This target dataset consists of 8628 sentence pairs, from different domains such as image captions, news headlines, and user forums and their similarity scores.

Table 5.7: Transfer Learning

Algorithms		**Sentence Relatedness**	
		Train	**Test**
Neural Model	CNN	71.2	68.4
	Bi-LSTM RNN	78	74

Despite this success of CNN model in transfer learning, Bi-LSTM RNN with the max pooling layer outperforms the hierarchical CNN model by a considerable margin as shown in Table 5.7. This result shows that the representations captured by Bi-LSTM RNN with max pooling has higher quality features for similarity and classification task.

5.7 Summary

In this chapter, we performed various experiments on three models on RTE and sentence relatedness task. Out of all the models, Bi-LSTM RNN with the max-pooling layer, achieved best performance in extracting the semantic representation. The optimal hyper-parameters and the impact of various dimensions of sentence representation on the performance, were studied for the best performing model. In classical ML models, Alignment and similarity features contributed more to the prediction. Next chapter presents the conclusion of this comparative study.

Chapter 6

Conclusion

Recently, applications based on language understanding domain have been commercially a huge success, particularly dialogue assistants. Language understanding serves as a primary component in many NLP applications such as dialogue assistant, search engine, plagiarism checker, information extraction etc. There is a growing need for more accurate language understanding models to fulfil the requirements of these applications.

In this project, we performed a comparison study of different existing models on their ability to learn the semantics of a text. The experiments and the results of our comparison study is presented in Chapter 5. Here, we summarise our findings:

- For both STS task and RTE task, BiLSTM RNN with max pooling proposed in Conneau et al. (2017) outperforms other models. Gradient boosting and ensemble model (SVM + RF + GB) gave the best performance among the traditional ML algorithms.

- For traditional ML models, the selection of features plays vital role in the performance of the model. It is important to exclude the feature that makes a negative contribution to the prediction. From the experiment results, all the features contributed to the learning and there was no need for feature selection. Similarity features based on pre-trained word representation had more impact on performance followed alignment based features.

- For neural model, we note that using 2048-dimensional embedding yields the best performance but only by a small margin. Network with larger dimensions take longer time to converge as the network complexity increases.
- Transfer learning allows us to exploit the knowledge gained while solving a task (source) and to apply it in different, but related task (target). This learning is very useful in cases where training data is scarce. In the experiment 5.6, Hierarchical CNN and Bi-LSTM RNN was compared for their performance in transfer learning by training it in RTE task and applying the gained knowledge to sentence relatedness task. Although CNNs are good choice for transfer learning in image domain, they are out performed by RNN networks in language domain.

Bibliography

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., et al. (2015). Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Agirre, E., Banea, C., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., Mihalcea, R., Rigau, G., and Wiebe, J. (2016). Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@ NAACL-HLT*, pages 497–511.

Agirre, E., Diab, M., Cer, D., and Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Bird, S. and Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Brown, D. E. and Huntley, C. L. (1992). A practical application of simulated annealing to clustering. *Pattern recognition*, 25(4):401–412.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing:

Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Deerwester, S. C., Dumais, S. T., Furnas, G. W., Harshman, R. A., Landauer, T. K., Lochbaum, K. E., and Streeter, L. A. (1989). Computer information retrieval using latent semantic structure. US Patent 4,839,853.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Han, L., Kashyap, A. L., Finin, T., Mayfield, J., and Weese, J. (2013). Umbc_ebiquity-core: semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 44–52.

Harris, Z. S. (1954). Distributional structure. In *Papers in structural and transformational linguistics*, pages 775–794. Springer.

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3294–3302. MIT Press.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 3111–3119. Curran Associates Inc.

Ng, A. (2018). Sequence models course, deeplearning.ai.

Pagliardini, M., Gupta, P., and Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Salton, G. (1971). The smart retrieval systemexperiments in automatic document processing.

Severyn, A. and Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Shao, Y. (2017). Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 130–133.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. 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)*, volume 1, pages 1556–1566.

Tian, J., Zhou, Z., Lan, M., and Wu, Y. (2017). Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 191–197.

Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems.

In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Zhao, H., Lu, Z., and Poupart, P. (2015). Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.