Sentence Encoders for Semantic Textual Similarity - A Survey

Aarthi Babu

March 15, 2018

Contents

1 Introduction **3**

1.1 Natural Language Semantics 3

1.2 Computational Semantics 5

1.3 Project Overview 6

2 Background **7**

2.1 Semantic Textual Similarity (STS) 7

2.1.1 STS Applications 8

2.1.2 Data for STS 9

2.2 Brief History of Meaning Reprsentations 10

2.3 Machine Learning models 12

2.3.1 Traditional ML models 12

2.3.2 Neural Networks 15

2.4 Word Vector Representaions 17

2.5 Sentence Representation Model 20

3 Related Work **22**

3.1 Traditional Machine Learning Models 22

3.2 Neural Models 24

4 Modeling Sentence Encoders **27**

4.1 Ensemble Model 29

4.1.1 Features . . . 30
4.1.2 Learning Models . . . 31
4.2 Convoutional Neural Network . . . 32
4.2.1 Sentence Model using CNN . . . 33
4.2.2 Similarity Measure using FCNN . . . 34
4.3 Recurrent Neural Network . . . 35
4.3.1 Bi-LSTM with max pooling . . . 35
4.4 Summary . . . 38

5 Experiments **39**
5.1 Encoder Architectures - Performance . . . 40
5.1.1 Experiment Setup . . . 41
5.1.2 Ensemble Model . . . 41
5.1.3 CNN Model . . . 41
5.1.4 Bi-LSTM Model . . . 41
5.1.5 Experiment . . . 41
5.1.6 Analysis . . . 41
5.1.7 Ensemble Model . . . 41
5.2 Encoder Architectures - Hyperparameter Tuning . . . 41
5.3 Features Impact . . . 41
5.4 Representation Dimension Size . . . 41
5.5 Transfer Learning . . . 41

Chapter 1

Introduction

This project studies and evaluates the ability of different encoders in sentence understanding problem in Natural Language Processing (NLP). Semantic Textual Similarity task is used to evaluate the semantics of a sentence captured utilizing these encoder algorithms. This study helps in advancing our understanding of existing models.

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 system etc. (Jurafsky and Martin, 2014). These tasks involve processing text and understanding the meaning of words and phrases in it.

1.1 Natural Language Semantics

As this project studies the ability of machine learning algorithm to extract representation of meanings of the text, it is essential to know what constitutes a meaning. This section briefly introduces the semantics and its similarity based on two approaches.

In the field of Linguistics, Semantics studies the meaning or sense of words and their relationship with other words. Contextual information is always to used 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 meaning of the words that is related to the function of the sentence such as articles, determiners, pronoun or prepositions.

Many properties of human languages make this a very complex task for computers. For words, a single words can have various different meaning also known as sense. For example, a word bank can mean a river bank or financial institution. But when humans read a sentence consisting bank and wood, they understand the meaning through context with their knowledge of the world. In some case, the meaning of a word is part of another word such animal and cat. Therefore, we can deduce that the words have 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 exists a lot of techniques to find word similarity, but fewer approaches to find sentence similarity.

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 WordNet (Pedersen et al., 2004) consisting of highly relevant information like synonyms, words relation tree etc. The corpus-based method measures word similarity using sizeable raw text corpora as a source data to infer information such as co-occurrence, 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 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), 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 section 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 sentence similarity task. Chapter 4 lays out the proposed study and its motivation. It also presents the architecture and the algorithm details of the encoders. Finally, chapter 5 presents the experiment set and its evaluation that establish 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 2.2, 2.3 discusses the principles of feature-based machine learning, and neural networks approach. Section 2.4, 2.5 discusses their use in sentence encoding.

2.1 Semantic Textual Similarity (STS)

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. STS is the task of finding how two sentences are closely related concerning its meaning (Agirre et al., 2012). It constitutes in many natural language processing (NLP) applications as a primary component. This task serves to be ideal for this study because to succeed in this task; the representation should consist of proper semantics.

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 impact of different sentence representation approaches on NLP applications. In 2012, 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 proposed approaches every year. (Agirre et al., 2012). 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 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.

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,

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.

newswire headlines etc. (Agirre et al., 2016). For example, STS tasks are being used 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 case of neural networks, the size of the data has a high influence on its performance as it doesn't get chance to optimize well without training on various domains or aspects of the problem. In this project,the model 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 task. 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, Flickr30K corpus consisting 160k captions (Bowman et al., 2015). It was hand annotated dataset comprising 550k sentence pairs.

2.2 Brief History of Meaning Reprsentations

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 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. 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

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

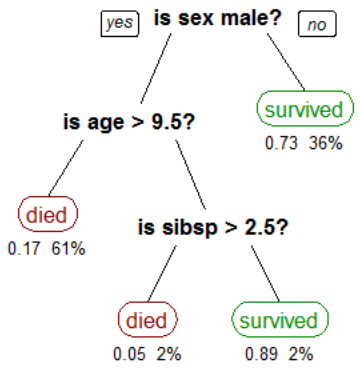
Figure 2.2: Term-Term Matrix

	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	

from V as its rows. With 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 an almost 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: 1) dimensionality reduction using singular value decomposition; 2) neural network approaches like skip-gram and CBOW. In this project, we will focus on the neural models used for creating words and sentence representations, as the latter method is computationally efficient than former approach (Jurafsky and Martin, 2014).

Figure 2.3: A simple Decision Tree



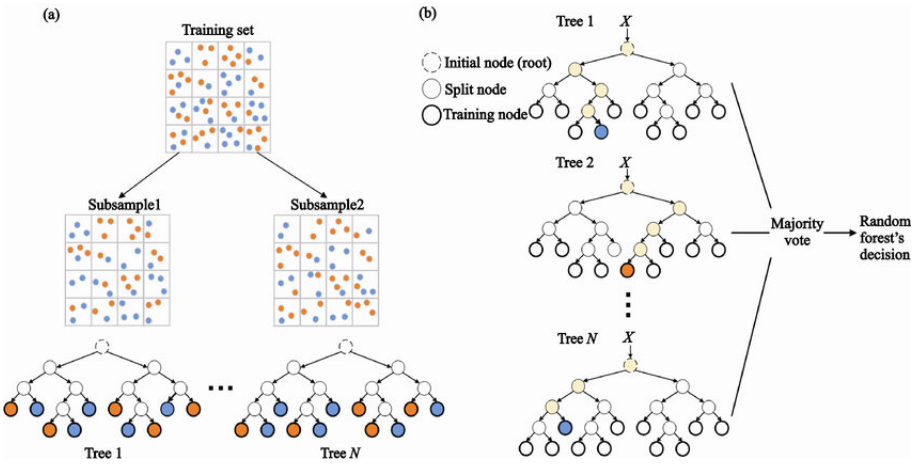
2.3 Machine Learning models

The learning theory and pattern recognition in AI gave rise to the field of machine learning. 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 learns a model *h(x)* = *y* from a collection of statistical feature set, extracted from the input in training dataset. Then, it makes predictions on the unseen data. While training , the parameters are optimized based on error rate of training time output $\bar{y_i}$ against true 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 output such as RTE task are known as the classification problem, and the data with the continuous value output such as sentence relatedness are called Regression problem.

2.3.1 Traditional ML models

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

Figure 2.4: Random Forest



Random Forest, an ensemble classifier that consists of a collection decision trees. Decision Tree is a tree with leaf and decision nodes that represents 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 for the prediction based on a specific sequence of rules. Each internal nodes 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 overfit as the decision node rules are built from the distribution of the training data. Therefore, it doesn't generalize well on the unseen data while testing. But, Random forest doesn't overfit as it is a 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 higher

dimension using a kernel function. In higher dimension, it creates a linear separating hyperplane using kernel function to classify the input data with its class label. This separating hyperplane is 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

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

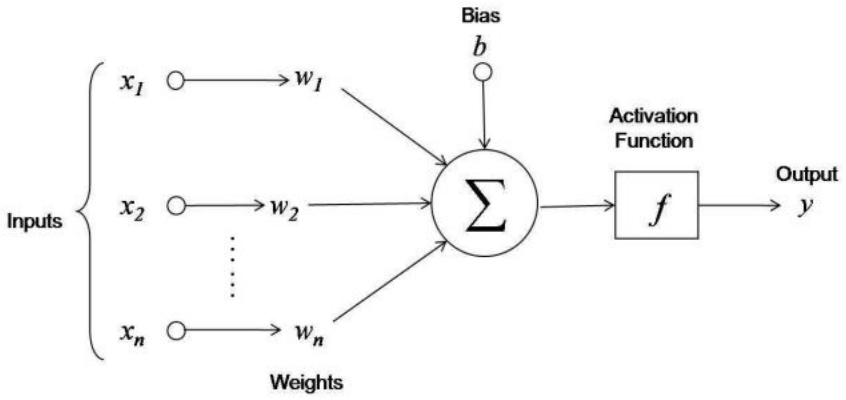
and c_i are the base learning models,

Boosting algorithm learns the model by assigning weights Θ to the data and training the weak classifiers against the weighted dataset. At each iteration, the weights are optimised in a way that the misclassified data points get higher weights.

Performance of all these algorithms crucially depended on the features selected to represent the sequence. For example, choosing a feature/attributes such as pos-tags proportion, syntactical structure equivalence, word taxonomy etc. to train a model for STS task would obviously give better performance than a model trained on only 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 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 task. Neural networks are one such algorithm which can learn feature on it own.

Since 1980, various neural network algorithms are proposed. Initially, neural networks didn't perform well in any task as they learn feature by itself and requires 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.

Figure 2.5: A simple neuron



2.3.2 Neural Networks

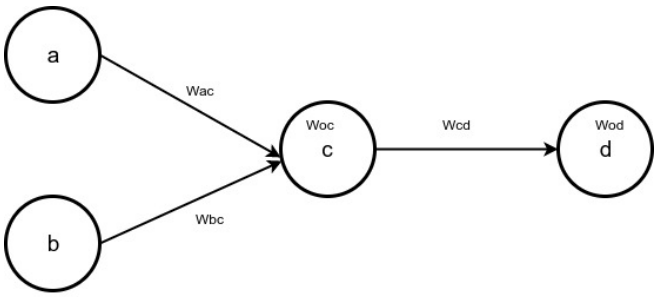
A neural network is a directed graph with neurons as its node. Neurons are computational units connected by directed links. Each link has its weight that determines its importance or strength. Similarly, all the computational units consist of activation functions that are applied to the input. The activation function can be any linear or non-linear function such as sigmoid (σ), hyperbolic tangent (tanh), rectified linear units (RELU) etc. So, an output of any computational unit 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:

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

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

This section explains the working of a simple neural network. 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.

Figure 2.6: Netural Network



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 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 Representaions

Firth (1957) 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 that the usefulness of word representations in various downstream NLP tasks. Inspired by these models, Mikolov et al.

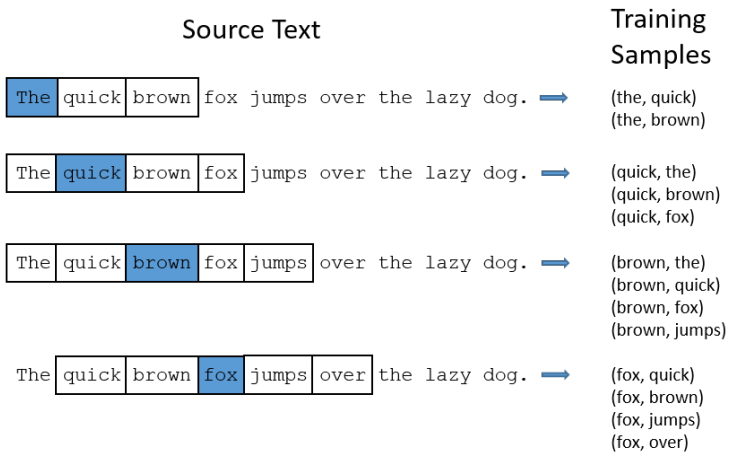


Figure 2.7: Training samples for skip-gram - Words pairs from raw corpus

(2013) proposed two novel methods: 1) Skip-Gram with Negative Sampling (SGNS); 2) Continous Bag of Words(CBOW) models for learning word representations. Later, the SGNS model is discussed in detail, 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 $1 \times |D|$, where D is the dimension of 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 normalised using softmax 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 by

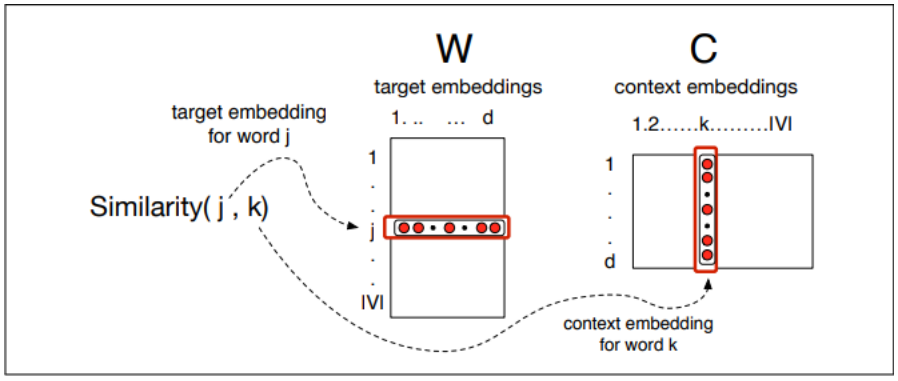


Figure 2.8: Context and Word matrices (Jurafsky and Martin, 2014)

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

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 L tunes the weight parameters in this network by using

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

The derivative of L with respect to input units of output layer and hidden layer denotes the prediction error is calculated. The learning algorithm is started with the randomised word and context matrices (weight parameters of this network). Optimiser 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 Softmax 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).

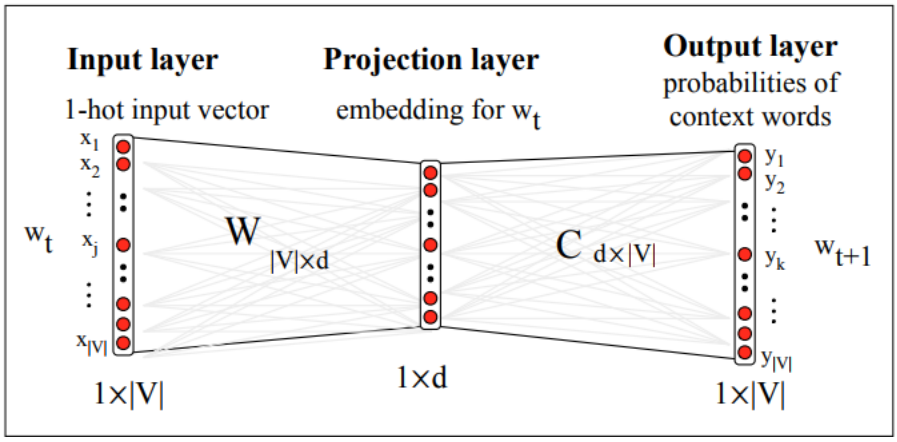


Figure 2.9: Skip-gram Model (Jurafsky and Martin, 2014)

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 sentences 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 context. These generic representations are vital as it 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.

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 generalisable sentence representaion using Transfer learning. This project proposes to compare and study different models used for STS tasks to infer how they impact in learning good sentence representations.

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 datasets 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 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 datasets. 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 unigram (one word) or bigram (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 techiques 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 SEMEVAL 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, 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 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 consist 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 exisiting 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 parsers for dependecy 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 handcrafted 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 78 % correlation on STS task.

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 model. 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 enoder 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 retrievel, Paraphrase detection, Entailment/semantic relatedness, 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).

Chapter 4

Modeling Sentence Encoders

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 feature is a hard. Although neural models learns the feature by itself, they suffer from an inherent bias toward the task and dataset 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. An important factor to note is the task for which the model is trained. 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

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

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

- What are the vital features in prediction while using traditional machine learning models ?
- What are the trade-offs incurred by neural networks as opposed to the traditional machine leanrning models?
- What is the impact of various activation functions used in the encoders hidden layers?
- What is the preferable neural network architecture for learning better sentence representations?
- What are the impact of various optimisers in training a model?
- Since the dimensionality has direct effect on the memory requirements and processing time, what dimensionality size has good trade-off between accuracy and training time?

In this project, a systematic comparsion of different encoder techniques have been carried out to assess their ability to capture semantics of the entence, 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 BiLSTM RNN with max-pooling (Conneau et al., 2017) were implemented. These encoder models achieved good accuracy in predicting sentence relatedness, and it outperformed the state-of-the-art encoders such as SkipThough, FastSent while being much faster to train than others. The neural models were implemented using pytorch and keras library. The ensemble model was implemented using scikit learn library. Conneau et al. (2017) demonstrated that Recognizing Textual Entailment (RTE) captures semantics very well. Based on this inference, the neural models were trained using dataset from Standford

Natural Language Inference (SNLI) corpus (Bowman et al., 2015), and Sentences Involving Compositional Knowledge (SICK) corpus (Marelli et al., 2014), for semantic relatedness and RTE tasks.

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 form of similarity features. Therefore, effective feature representations that captures 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 corresponding weight vector. The models learns a 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 return by them. In case of classification task 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.

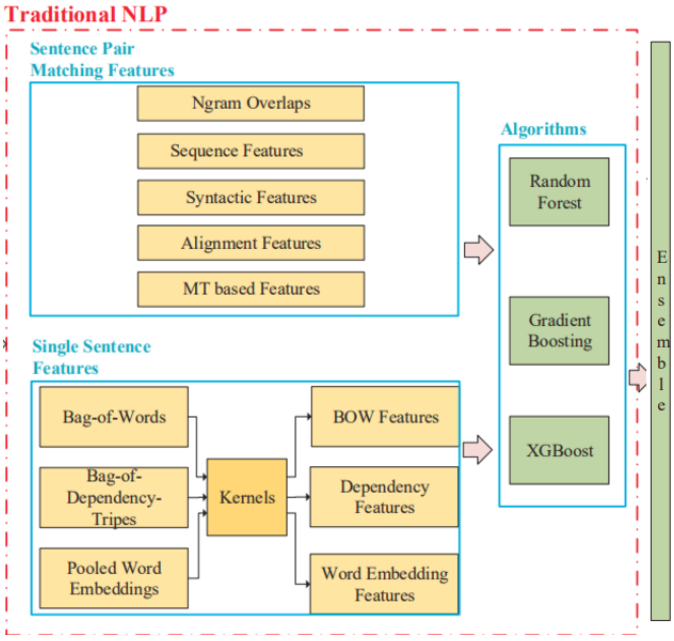


Figure 4.1: Ensemble Model (Tian et al., 2017)

4.1.1 Features

For this ensemble model, the features of a sentence pair is extracted based on length of two sentence, 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 overlap are useful in expressing the common expression between two sentence. 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 similarity, the words are lemmatized where the words are replaced with its root word.

Although 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 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 sentence 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 feature includes 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 word/sequence to the other. This is also known as Path distance. This feature implies the word/sequence more similar as shorter the distance.
- **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 measure 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 Learning Models

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 takes a 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 consist 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 Stacking algorithm. In stacking, the training set is split to several subsets and each models is trained and tested on one of those subsets. Finally the predictions are fed into a 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 as the multi-class classifier. 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 feature vectors inspired from Shao (2017) is shown in Figure 1. 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 representaion 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 vectos for the sentence that capture its semantics.

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 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 reduuces 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

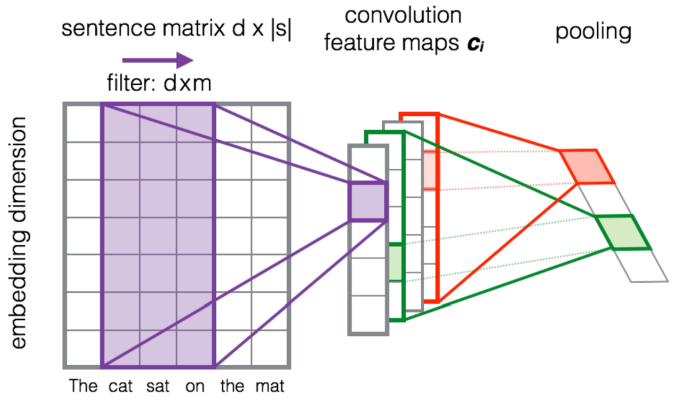


Figure 4.2: CNN Sentence Model Severyn and Moschitti (2015)

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

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.

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

4.2.2 Similarity Measure using FCNN

This network consists of one hidden layer a 300 node size, and an output layer of a size 6. 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 desecribe 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 consist 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 ability to accept input of arbitary length and return fixed dimensional vector. They are capable of propogating the previously processed input's information while processing a word in the given sequence at timestep *t*. This encoder architecture consist 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 cell 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 helps LSTM to keep track of synactical structure in case where it stores a gender of the

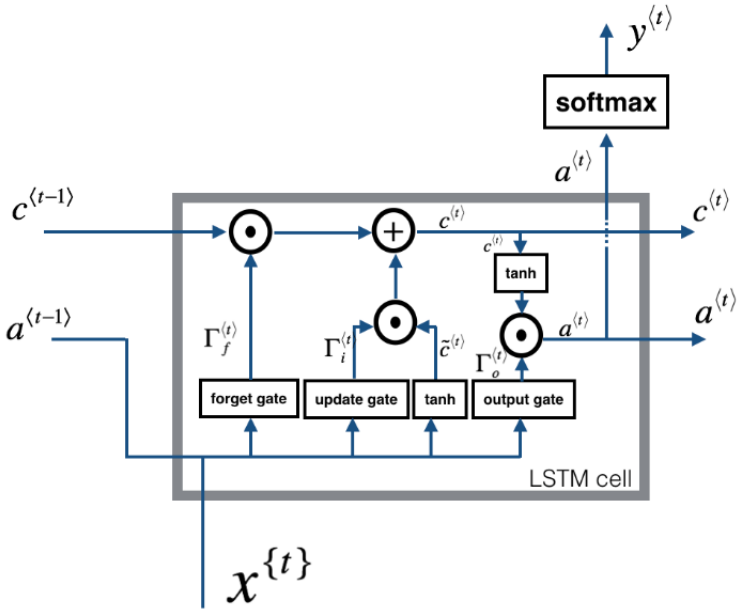


Figure 4.4: A Single LSTM cell (Ng, 2018)

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 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 cell . 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 .

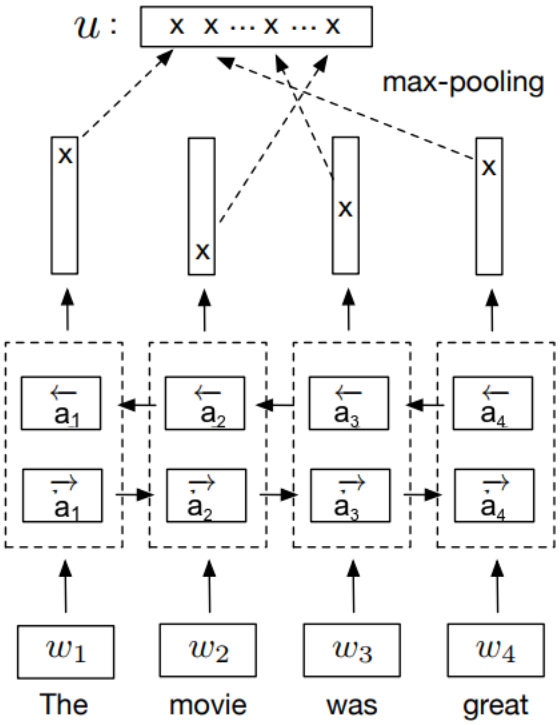


Figure 4.5: A Bi-LSTM Network (Conneau et al., 2017)

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

Textual Entailment Classifier

This semantic difference matrix is fed into a full connected neural network decoder to classify the textual entailment in the sentence pair. This network consist of 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 decoder's softmax layer, one hot vector are calculated by assigning 1 to the class with maximum probability and 0 to others. It is compared againt 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. Following Chapter lays out the experiments and evaluations of these models.

Chapter 5

Experiments

This chapter lays out the comparison study on the ability of traditional machine learning and neural models in extracting the meaning representation of sentence. In this project, the models are trained on Sentence Relatedness and Recognizing Textual Entailment (RTE) task. While training, the encoder model learns sentence representations that allows a decoder to judge how much two sentences are semantically similar or whether two sentences have meaning overlap. To perform better in these tasks, the encoder model should capture the compositional meaning of the given sentence. This constraint makes it the ideal tasks for evaluation.

The performance of the models on STS task is evaluated using two metrics, 1) Accuracy of its prediction; 2) Training Time. The accuracy of the STS models is measured using Pearson correlation between machine generated semantic score and gold standards created using human judgement. It helps in capturing the linear relationship between the predicted and target semantic score. This correlation value ranges from -1 to 1 where, 1 indicates perfect positive correlation and -1,0 indicates no or negative correlation.

Although complex models give better accuracy, they take a long time to train. Analysing the training time of the encoder models help in finding a tradeoff between the training time and the model performance. The neural models implemented in this project, are trained using a machine with 4 $\times$ CPUs, 16GB RAM and 1 $\times$ NVIDIA Tesla K80 GPU to perform

the comparison study.

In addition, the sentence representations encoders are evaluated for its performance on generating a generic sentence representation that aids in transfer learning. This evaluation is carried out by using representations learned from training RTE task as input in sentence relatedness task and vice versa. Generic representations are expected to perform well in both tasks in terms of accuracy.

In this chapter, we evaluate a variety of sentence encoding models, including a 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 informative, the experimental setup of each models architecture including its hyperparamters are outlined along with their performance. We use Stanford Natural Language Inference (SNLI) corpus and Sentences Involving Compositional Knowledge (SICK) dataset for training and evaluation of the sentence encoder models.

5.1 Encoder Architectures - Performance

In this section, the performance of the encoder models on Sentence Relatedness and RTE task 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 dataset used for this experiment is a collection of data from various domain. The trained models are tested with sentence pairs that was not encountered by them while training.

Table 5.1: My caption

Algorithms		Sentence Relatedness		RTE	
		Train	**Test**	**Train**	**Test**
Single ML Model	Support Vector Machine (SVM)		76.2	74.9	75.2
	Random Forest (RF)		77.5	74.7	75.4
	Gradient Boosting (GB)	81.1	77.7	74.9	76.7
	XGradient Boosting (XGB)		77.2	75.6	76.5
Ensemble	SVM + RF + GB		75.4	76	76.9
Neural Model	CNN	69.6	61.4	71.6	61.3
	Bi-LSTM RNN	79.1 fake	75.2	83.98	84.35

5.1.1 Experiment Setup

Ensemble Model

CNN Model

Bi-LSTM Model

5.1.2 Experiment

5.1.3 Analysis

5.1.4 Ensemble Model

5.2 Encoder Architectures - Hyperparameter Tuning

5.3 Features Impact

5.4 Representation Dimension Size

5.5 Transfer Learning

Bibliography

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 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, 2012.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 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, 2015.

Eneko Agirre, Carmen Banea, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@ NAACL-HLT*, pages 497–511, 2016.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

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

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 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, 2015.

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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 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, 2017.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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, 2014.

Ronan Collobert and Jason Weston. 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, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 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, 2017.

Scott C Deerwester, Susan T Dumais, George W Furnas, Richard A Harshman, Thomas K Landauer, Karen E Lochbaum, and Lynn A Streeter. Computer information retrieval using latent semantic structure, June 13 1989. US Patent 4,839,853.

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

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 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, 2013.

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

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 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, 2016.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3294–3302. MIT Press, 2015.

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

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 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, 2014.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 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., 2013.

Andrew Ng. Sequence models course, deeplearning.ai. 2018. URL `https://www.coursera.org/learn/nlp-sequence-models/home/welcome`.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

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

Gerard Salton. The smart retrieval systemexperiments in automatic document processing. 1971.

Aliaksei Severyn and Alessandro Moschitti. 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, 2015.

Yang Shao. 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, 2017.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 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, 2011.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 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, 2015.

Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 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, 2017.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 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, 2015.