Sentence Encoders for Semantic Textual Similarity - A Survey

February 9, 2018

Abstract

Finding semantic similarity between two sentences, is a basic language understanding problem that is applicable in many natural language processing(NLP) applications. Sentence representations are the basic components that greatly impact the performance of Semantic Textual Similarity(STS) models. Despite the existence of well established models for generating word representation and the consensus about its usefulness, there is a lack of profound research on learning the sentence representations. In this project, I propose to systematically compare models that achieved state of art performance in Sem-Eval STS shared task (Cer et al., 2017) proposed for learning STS and sentence representation. In addition to the comparative study, I will analyse the internal component of the architecture of each model and its impact on STS task performance and learning of sentence representations.

1 Background

1.1 Semantic Textual Similarity (STS)

Semantic Textual Similarity (STS) is the task of finding how two sentences are closely related in regard to its meaning (Agirre et al., 2012). It is used as a primary component in

many natural language processing (NLP) applications. Until 2012, there was no unified framework available to study problems related to 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 event(conference) for STS. The main objective of the event was to clearly 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, 2) Recognizing Textual Entailment (RTE). Sentence Relatedness aims to find the semantic similarity score ranging from 0 to 5 between two sentences. The reasoning for the similarity score is further discussed in Table 1. RTE 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).

Task Description

When it comes to predicting the measure of textual similarity, we are given a list of sentence pairs $X = \{(S_{a1}, S_{b1}),.., (S_{aN}, S_{bN})\}$ 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,

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

where function f maps sentence pairs to a vector representation, in which each dimension expresses a certain type of similarity between the input sentences e.g., lexical, syntactic, and semantic. The weight vector, w is a parameter of the model that is learned

during the training where h denotes the STS prediction model.

Table 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.

1.2 STS Applications

Semantic similarity between two sentences is a basic 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 are used as a primary software component in many applications such as image-captioning, automatic short question answer grading, search engines, plagiarism, newswire headlines etc (Agirre et al., 2016). For example, STS tasks can be adapted 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) heavy revision of material 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).

1.3 Word and Sentence Represention

1.3.1 Brief History

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 models have been proposed to estimate vector representations of words. These representations were based on the co-occurrence matrix of words in the documents consisting of raw text corpus. It was also known as 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 the vocabulary V of the model. Vector representations had to be created for all the words in the vocabulary. Initially, the term-document matrix which was introduced used all the unique words from V as its rows and the documents as its columns. This matrix was used to find similar documents as part of an information retrival system (Salton, 1971). Each value in the matrix denotes the number of times a word occurs in a specific document. This was based on the idea that similar documents have similar words resulting in similar vectors.

To measure similarity between words, term-context matrix using words as its columns as well as its rows was introduced. Given the size of the vocubulary $|V|$, the term-context matrix of dimension $|V| \times |V|$ with a word co-occurrance count within a specific window size, generated. 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. Cosine similarity of two vectors are commonly used as a 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, I will focus on the neural models used for creating words and sentence representations, as latter approach is computationally efficient than

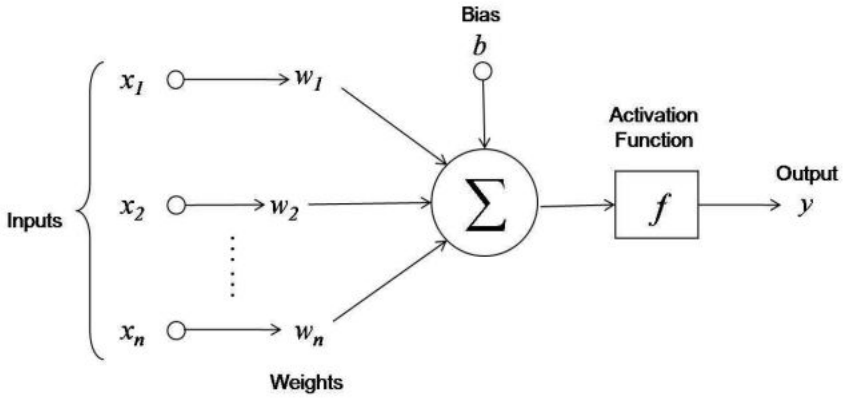


Figure 1: A simple neuron

former approach (Jurafsky and Martin, 2014).

1.3.2 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 own 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 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 1. It can be formally expressed as:

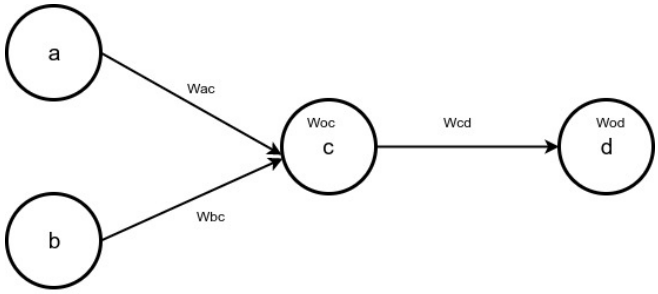
$$z = \sum_{i=1}^{n} wx_i + b$$

$$y = f(z)$$

In this section, the working of a simple neural network is explained. A simple neural network with two inputs a and b, one hidden unit c, and an output unit d can be visualized as shown in Figure 2. 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: 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.

$$in_c = W_{ac} \times a + W_{bc} \times b + W_{oc}$$
$$A_c = \frac{1}{1 + exp(-in_c)}$$
$$in_d = W_{cd} \times A_c + W_{od}$$
$$A_d = \frac{1}{1 + exp(-in_d)}$$

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 *w.r.t* all the weights.

For example, gradient of loss computed w.r.t W_{cd},

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

Weights are updated based on the gradients as shown below.

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

where δ_d is a modification error, δ_d is computed,

gradient of Loss w.r.t A_d $\times$ gradient of activation output A_d w.r.t in_d.

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 with respect to each of the output units.

1.3.3 Word Representation

Bengio et al. (2003) proposed a language model based on a neural network that predicts the next word 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 these word representations can be used as an input for various downstream 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. In the next section, we will discuss the SGNS model as it is widely adopted and used as an input for many sentence representations model (Kiros et al., 2015).

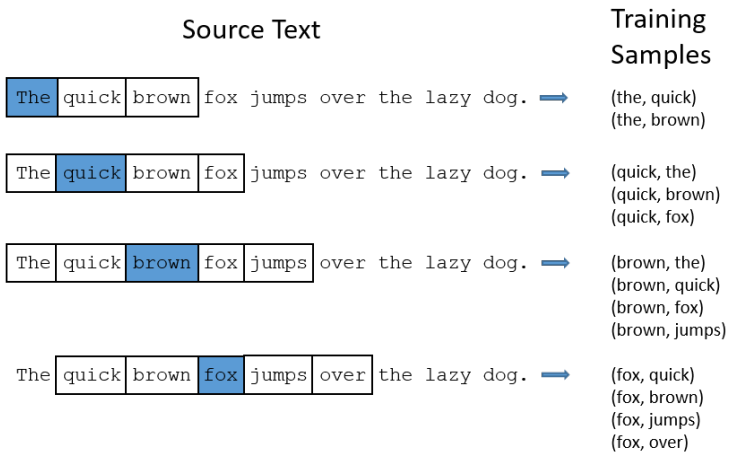


Figure 3: Training samples for skip-gram - Words pairs from raw corpus

The SGNS model is trained on a monolingual text corpus, where every center word is used to predict the surrounding words within a window size as shown in Figure 3. The model takes word input in the form of One-Hot vector of dimension $1 \times |V|$. While building a vocabulary from the training corpus, One-Hot vector $1 \times |V|$ is 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. To find the context word score, the dot product is performed between the hidden layer and the context matrix as shown in Figure 4. 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 word.

For a word w_j, the probability of any k^{th} word in V is calculated by

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

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. The weight

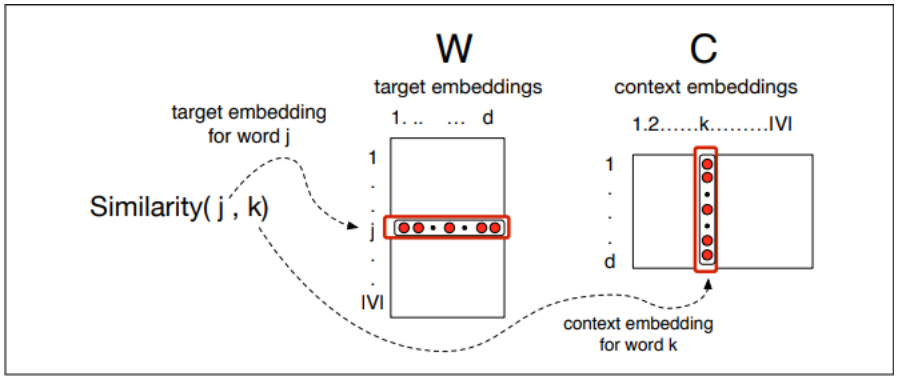


Figure 4: Context and Word matrices (Jurafsky and Martin, 2014)

parameters in this network are tuned by using cost function L

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

The prediction error is calculated by taking the derivative of L w.r.t to input units of output layer and hidden layer. 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 backpropagation 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).

1.3.4 Sentence Representation Model

As word representation models became very useful and predominantly used, a natural next step was to extend such approaches to sentences. The goal was to learn a general sentence representations 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

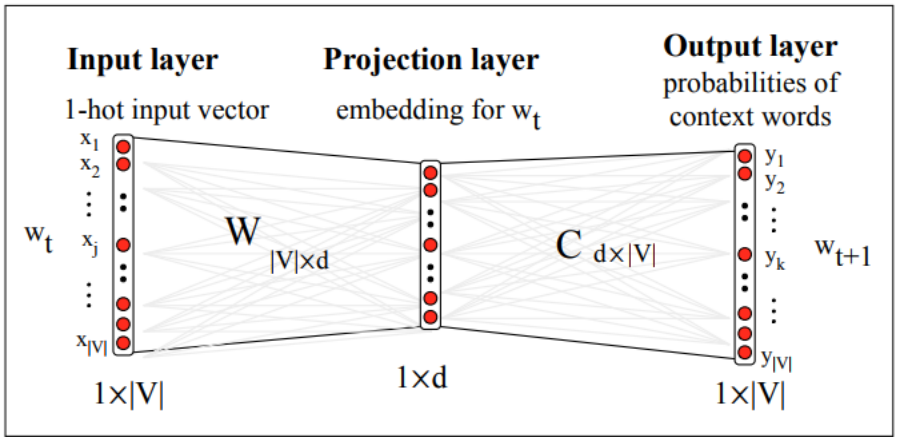


Figure 5: Skip-gram Model (Jurafsky and Martin, 2014)

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 captured 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 efficient sentence representation that consists of the whole meaning with context. This is important as it can be used for various tasks with minimal adaptation. This property results in smooth transfer of learning to the model that learns any specific task. By promoting Transfer Learning, the training complexity and training time for specific tasks decreases.

1.3.5 STS Features and Accurracy

For STS task, traditional machine learning algorithms used handcrafted features from the raw sentence such as word overlap, knowledge based similarity, length features and other similarities. Neural models use word representation to train the models. Recently neural models Conneau et al. (2017) Kiros et al., 2015 Shao (2017) have been popular as they

outperform all the traditional machine learning algorithms. Ensemble techniques which combine the prediction techniques of traditional machine learning models and neural models are also being explored.

The performance of the model is measured using Pearson Correlation of the predicted score with a human judgment score given in the dataset (Agirre et al., 2015). Generally, correlations measure the extent to which two variables change together. Pearson Correlation, meaures the direction of linear relationship between pairs of continous variables that are well suited for the sentence relatedness task.

1.3.6 STS Dataset

For training and testing of the STS model, I use data published in SemEval 2012-SemEval 2017. It includes 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 (Agirre et al., 2015).

2 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. I 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.

2.1 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.

2.2 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).

3 Proposed Work

A wide variety of supervised and unsupervised encoders for learning sentence representation have been proposed by NLP researchers in recent times. However, there is a lack of understanding about the characteristics of different encoding techniques that can capture useful, generic semantic information (Conneau et al., 2017). Supervised neural models 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, unsupervised learning models give more importance to general information, therefore, failing 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 in which the model is trained. Similarly, the encoders' architecture for both supervised and unsupervised neural models also impacts learning in different ways. A comparison study on these encoder's architecture will enable us to gain better insight on how better sentence representations are captured.

In this project, I propose to perform a systematic comparsion of different encoder techniques for generating sentence representations and their ability to capture semantics of the sentence. To do this, I am implementing and studying the following models: support vector machine (SVM), Random Forest (RF), Convolutional Neural Network Encoder (Shao, 2017) and BiLSTM RNN with max-pooling (Conneau et al., 2017). The models proposed in Shao (2017) and Conneau et al. (2017) are the best performing models in STS tasks. These models will be implemented using sci-kit learn, PyTorch and Keras library. Conneau et al. (2017) demonstrated that Recognizing Textual Entailment (RTE) captures semantics very well. Based on this inference, I will train our encoders on Standford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), and Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014), for semantic relatedness and RTE tasks.

The main objective of this comparison study is to understand the efficiency of the sentence representations models and to answer the following questions:

- What are the vital features in prediction while using machine learning models ?
- How do we measure effectiveness of the 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?

4 Evaluation

In this project, the performance of the models proposed for 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. The Pearson correlation is formally calculated by,

$$r = \frac{\sum Y\bar{Y} - \frac{(\sum Y)(\sum \bar{Y})}{N}}{\sqrt{(\sum Y^2 - \frac{\sum Y^2}{n})(\sum \bar{Y}^2 - \frac{(\sum \bar{Y})^2}{n})}} \tag{3}$$

where Y is target STS score vector, $\bar{Y}$ is predicted score vector and n is the size of Y.

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. To perform this analysis, all the models implemented in this project will be trained using a machine with 4 $\times$ CPUs, 16GB RAM and 1 $\times$ NVIDIA Tesla K80 GPU.

The evaluation metrics discussed in this section will be used for 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.

5 Implementations

I have implemented the models proposed in Shao (2017) and measured their performance using Pearson's Correlation. In this section, the implementation details of one of the model

architectures used for the comparison study are discussed. Two other models are yet to be implemented.

5.1 A Simple CNN Model For STS Task

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 distributional 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.

Sentence Model using CNN

Our 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.

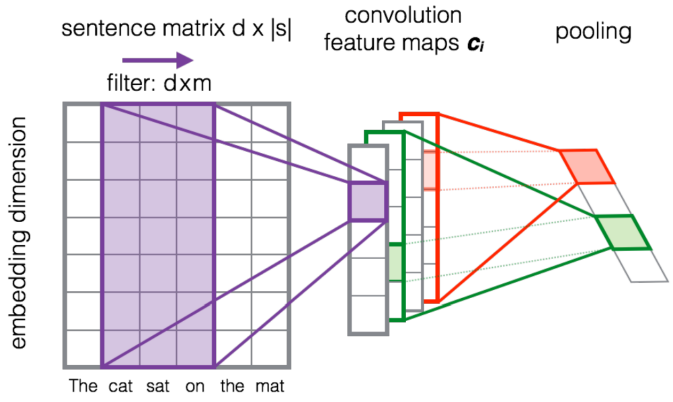


Figure 6: CNN Sentence Model Severyn and Moschitti (2015)

Figure 7: 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 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 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.

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

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 7.

Finally, the model is trained using the categorical cross-entropy loss: given a vector of probabilities p for a training pair of sentences, if the correct similarity category corresponds to index i of p, the model will evaluate the loss as $L = log(pi)$.

6 TimeLine

In this section, the timeline regarding my project is discussed. I have completed implementing the CNN model with a small dataset and reproduced the results stated in Shao (2017). For ensemble models and traditional models, I have implemented a model to hand engineer 71 features categorised under single sentence features and sentence pairs features. This model and InferSent will be further discussed in future reports.

Table 2: TimeLine

Task	Task Period	
Literature Survey	Nov - Jan	Completed
Implementation - Traditional ML models	Dec	Completed
Implementation - CNN Model	Nov	Completed
Implementation - InferSent	Jan - Feb	InProgress
Proposal	Jan 18	Under review
Project Report	Jan - Mar	In-Progress
Project Defence	-	-

References

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 M Cer, Mona T 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 *SemEval@ NAACL-HLT*, 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 prob-

abilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 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.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. Umbc_ebiquity-core: Semantic textual similarity systems. In * *SEM@ NAACL-HLT*, pages 44–52, 2013.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016.

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

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 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 *SemEval@ COLING*, pages 1–8, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

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.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 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.