A Survey on Word Embedding Techniques and Semantic Similarity for Paraphrase Identification

Abstract In Natural Language Processing (NLP), Paraphrase Identification (PI) determines the relatedness between the pair of sentences having fewer or negligible lexical overlap but still pointing towards the same meaning. The major challenge faced while attempting to solve this problem is the many possible linguistic variations conveying the same purpose. This paper aims to provide a detailed survey of traditional similarity measures, Statistical Machine Translation metrics, Machine Learning and Deep Learning techniques and a well-defined flow between them. This article encompasses various word embedding methods and step-wise derivation of its learning module. This survey paper also provides a definite flow pointing towards the evolution of Deep Learning in an unambiguous manner. A comparative analysis of various techniques to solve PI is presented and it will provide research directions to work in the similar domain.

Keywords: Paraphrase Identification; Word embedding; Deep Learning, Convolutional Neural Network; Semantic Similarity.

1 Introduction

Paraphrasing means the restatement of the original sentence in some other words which is probably done for better understanding of the meaning of original sentence. For example, “The signal is green” can be restated as “The vehicles can now pass as the signal turned green” to give a clearer picture of the scenario. Also, the same original statement can also be restated in many different ways like, “The traffic signal can now turn from red to green”, “After some time, the traffic signal will change its color from red to green and then our vehicle can pass” to convey the same meaning. Paraphrase Identification can be viewed as a task to find a probability value whether the given pair of sentences is a paraphrase of each other. It is obviously not feasible to manually write all the possible ways which convey the same meaning and then deploy the system in practical use as it will take an infinitely huge amount of time, still, it will fail as there still lies a small chance of not covering each and every possible linguistic variation.

In a majority of NLP applications like Information Retrieval (De Marneffe et al. 2008, Duclaye et al. 2003), Question-Answering (Fader et al. 2013, Voorhees 2001, Harabagiu et al. 2003, Mollá & Vicedo 2007), Recommendation systems, Plagiarism Detection (Barrón-Cedeño et al. 2013) etc, finding semantic relatedness or semantic similarity between a pair of sentences plays an important role. The semantic similarity also has its application in Named Entity Recognition (NER) (Sekine & Ranchhod 2009, Shinyama et al. 2002). NER can be considered as a sub-task in information extraction system where various named entities are classified such as the names of persons, organizations, locations, quantities, currencies, etc. In conversational bots/engines, it is pretty much important to understand what the user actually wants to say and then map the query to get the accurate answer from knowledge-base, making the Paraphrase Identification task vital in such areas. A significant amount of research is being performed in this challenging task of finding the semantic similarity using Paraphrase Identification due to a lot of linguistic variations conveying the same meaning. The traditional systems were based on word-wise lexical overlap similarity and magnitude of words which failed to consider the meaning (Shinyama et al. 2002, Gomaa & Fahmy 2013), for example, the words ‘fired’ and ‘tired’ have high similarity value even though their meaning is totally different. This problem was solved by using semantic word-wise similarity using different external resources (knowledge-based semantic networks) like WordNet. As this approach was heavily dependent on word order and also didn’t consider n-grams (the context in which the word occurs) or phrases, this approach too failed. Research has shown that the bi-grams have about 70% dependency on each other and can considerably contribute in picking up the context in most of the times (Collins 1996). Consider the following pair of sentences taken as a sample from Microsoft Research Paraphrase Corpus (MSRP)

S1: “County Judge Tim Harley declared a mistrial because the jury could not reach a verdict.”
S2: “Judge Tim Harley declared a mistrial in the Adrian McPherson gambling trial today after the jury was unable to reach a verdict.”

Both the sentences S1 and S2 have the same meaning. In this case, the word-wise lexical and semantic word-wise based approaches will fail as there is less lexical overlap, word-order is not preserved and also no consideration of the n-grams. In order to solve such a problem, the paraphrases have to be identified and then they have to be semantically compared. In S1 and S2, ‘could not reach’ and ‘was unable to reach’ are paraphrases to each other conveyed in the context of ‘verdict’.

The first step to approach the task of PI is to represent the text in a way which is understandable to a computer system. This is done because the machine learning as well as Deep Learning algorithms accept the inputs in numerical form and cannot understand natural language text. Hence, the first major challenge is how well the words are represented in a format that the system can clearly understand with its associated semantic and syntactic meaning without losing the context. The words are converted to real-valued numbers or vectors. There are various ways to get vector representation of words. A detailed description about various methods is done in section V. But the major problem is how well these words are represented in the vector forms and the parameters in consideration like *window size* and the *dimension of word embeddings* which can greatly influence the accuracy in final results. The next major challenge the researchers face in the task of Paraphrase Identification is the lack of training data available. Some researchers tried to solve this problem by using annotated datasets, corpus construction and also by pre-training techniques.

This paper kick starts with a section by giving a brief introduction of various String-based, Corpus-based and Knowledge-based measures in section 2. Section 3 introduces all the techniques to convert words in their respective vector form. Also, the learning process is described with a clear explanation of equations. Section 4 deals with all the techniques used to attempt the task of Paraphrase Identification. The techniques include Statistical Machine Learning (SMT) metrics, Machine Learning algorithms, Deep Learning algorithms. Section 5 deals a thorough comparison in various forms. The comparative analysis is done in between word embedding techniques, various approaches to solve Paraphrase Identification, and finally by taking into consideration the important parameters used in Machine Learning and Deep Learning techniques. The paper concludes with section 6 by concluding the findings and future scope in the related domain.

2 Traditional Similarity Methods

The traditional methods to find similarity between the pair of sentences include word-wise lexical overlap and semantic similarity (word-wise) methods (Gomaa & Fahmy 2013, Agirre et al. 2016) which depends on magnitude and meaning of words respectively. These methods can further be classified in String Based and Corpus & Knowledge-based. It includes lexical similarity which is highly dependent on the number of overlapping words and semantic similarity which includes using resources like WordNet (Perkins 2010, Loper & Bird 2002, Pedersen et al. 2004, Agirre et al. 2009) to find synonymous words between sentence pair. Lexical similarity is explained by String based measures and semantic similarity is expressed by explaining Knowledge and Corpus-based similarity measures.

2.1 String based Similarity Measures

String-Based Similarity measures (Gomaa & Fahmy 2013, Agirre et al. 2016) focus on sequences of strings between by measuring the similarity and dissimilarity between sentence pair. String-based measures can further be classified as ‘character based’ and ‘term-based’. The aim of semantic similarity measure is to get the word-wise meaning between sentence pair. The semantic similarity methods can be further classified as Corpus-based and Knowledge-based similarity measures.

Measures	Description
Longest Common Subsequence (LCS)	Length of continuous sequence of characters between sentence pair.
Levenshtein	Distance between sentence pair by counting minimum number of operations needed to transform one string into the other.
Jaro	Number of common characters between sentence pair by considering spelling mistakes upto certain limit.
Jaro-Winkler	Extension of Jaro. Uses a prefix scale which gives more priority to strings that match from the beginning.
Needleman-Wunsch	Dynamic programming approach. Performs global alignment. Suitable for same length sentences.
Smith-Waterman	Dynamic programming approach. performs. Performs local alignment.
N-gram	Divide similar number of n-grams by maximum number of n-grams.

Table 1 Character-based Lexical Similarity Measures

2.2 Corpus based Similarity Measures

Corpus based measures uses corpora (large collection of text or spoken texts typically used in language research) to determining the semantic similarity between pair of sentences. The various corpus based semantic similarity measures are Hyperspace Analogue to Language (HAL), Latent Semantic Analysis (LSA) (Landauer et al. 1998), Generalized Latent Semantic Analysis (GLSA) (Royer 2005), Explicit Semantic Analysis (ESA) (Gabrilovich & Markovitch 2007), The cross language explicit semantic analysis (CLESA) (Sorg & Cimiano 2008), Pointwise Mutual Information Information Retrieval (PMI-IR) (Bouma 2009), Second order co-occurrence pointwise mutual

Measures	Description
Block distance	Distance from one point to other if grid-like path is followed. Also known as Manhattan distance, absolute value distance, boxcar distance, L1 distance, city block distance, Manhattan distance and boxcar Manhattan distance.
Cosine similarity	Similarity between two vectors of an inner-product space.
Dice's coefficient	Ratio of common terms to total number of terms between sentence pairs.
Euclidean distance	Square root of the sum of squared differences between corresponding elements of the two vectors.
Jaccard similarity	Number of shared terms over unique terms between sentence pair.
Matching coefficient	Number of similar terms (both vectors are non-zero).
Overlap coefficient	Similar to Dice's coefficient. Considers full match if one string is sub-string of the other.

Table 2 Term-based Lexical Similarity Measures

information (SCO-PMI) (Islam & Inkpen 2006), Normalized Google Distance (NGD), Extracting DIStributionally similar words using CO-occurrences (DISCO) (Matsuo & Ishizuka 2004).

2.3 Knowledge based Similarity Measures

The knowledge based semantic similarity measures aims to get the semantic similarity between sentence pair by using information which is derived from some semantic network. A classic and famous example of a semantic network is WordNet (Perkins 2010, Loper & Bird 2002, Pedersen et al. 2004, Agirre et al. 2009) which is a large knowledge base consisting of all Parts Of Speech (POS) grouped in set of cognitive synonyms (Gonzalo et al. 1998) (synsets). The knowledge-based measures can be classified in two broad categories namely, similarity-based and relatedness-based. Relatedness-based measure consists of *hso* (Pedersen et al. 2004), *leck* and *vector*. The similarity-based measures can be further divided in information content (Pedersen et al. 2004) (*res*, *lin* and *jcn*) and path length-based (*lch* (Sinha & Mihalcea 2007), *wup* (Guadarrama et al. 2013) and *path* (Pedersen et al. 2004)).

3 Word Embedding Techniques

The word embeddings are numbers that represent natural language text. A word embedding tries to map a word to a vector using a dictionary. The word embeddings can have discrete as well as distributed representation. The major benefit of using word embedding is that they are easy to work with as the word similarities are efficiently computed via low dimensional matrix operations. Almost all the machine learning and deep learning algorithms take input data in the form of integers or numbers rather than natural language text. Before applying any machine learning algorithm, the words in natural language have to be converted in a representation in such a format that a machine learning algorithm can understand. Converting a word to its vector form is an important step and in some cases, this step can affect the overall performance not only in terms of accuracy but also in terms of computation time (time complexity).

A hierarchy of word embeddings is been depicted in figure 1. There are two broad categories of word embedding namely, frequency based and prediction based. The words can be represented in a discrete or in a distributed fashion. Various frequency based representations are count vector TF-IDF vector (Ramos et al. 2003, Robertson 2004) and co-occurrence matrix (Matsuo & Ishizuka 2004). Discrete representations are seldom used in the task of paraphrase identification because although they might be a great resource, they do lack in nuances (synonyms), cannot handle newer words thus making it impossible to keep up to date, they are quite subjective and they require human effort to generate a vector. All these limitations make these discrete representations of words hard to compute word similarity. Hence, this survey literature mainly focuses on distributed word representations.

The state-of-the-art distributional representation of words (Mikolov et al. 2013) was proposed by Mikolov et al with the algorithm's name as 'word2vec' . Word2vec is a combination of two techniques namely, Skip-Gram model (predicting context words given target word) and Continuous Bag of Words (CBOW) model (predicting target word given context words) (Goldberg & Levy 2014, Levy & Goldberg 2014). The common methods used to train these models are hierarchical softmax and negative sampling. Skip-gram model aims to predict the surrounding words in a window of certain radius m around each word $t = 1...T$. The objective function is

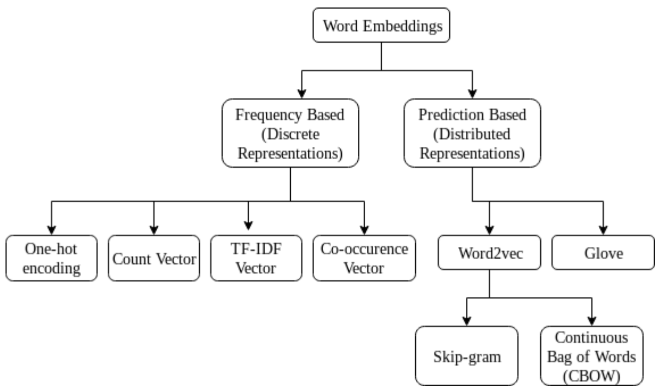
Word Embeddings
Frequency Based (Discrete Representations)
Prediction Based (Distributed Representations)
One-hot encoding
Count Vector
TF-IDF Vector
Co-occurence Vector
Word2vec
Glove
Skip-gram
Continuous Bag of Words (CBOW)

Fig. 1: Various types of Word Embeddings

to maximize the probability of the surrounding context word(s) given the current context word,

$$J^{'}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq m, j \neq 0} P(w_{t+j}|w_t;\theta) \tag{1}$$

(1) tells the model to look for each context word within a distance 2m (left and right side of context word with context window size of m) and maximize the probability of the correct context words around a given target. The model has to adjust many hyper parameters and word vectors of words so as to maximize the probabilities of actual context words surrounding a target words. θ is the set of parameters to be optimized. The equation can be alternately written in order to make sense during mathematical calculation as,

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \prod_{-m \leq m, j \neq 0} log(P(w_{t+j}|w_t)) \tag{2}$$

(2) is also called as negative log-likelihood. This objective function has to be minimized.
$P(w_{t+j}|w_t)$ can be formulated as,

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{v} exp(u_w^T v_c)} \tag{3}$$

where o is context word (output), c is the given target word, u_o, v_c are 'outside' and 'center' vectors of o and c respectively. The dot product (loose similarity) is calculated between two vectors and the exponential is applied in order to make the numbers positive, then normalization is done at denominator part of (3) to get the probability. (3) is called as 'softmax function'.

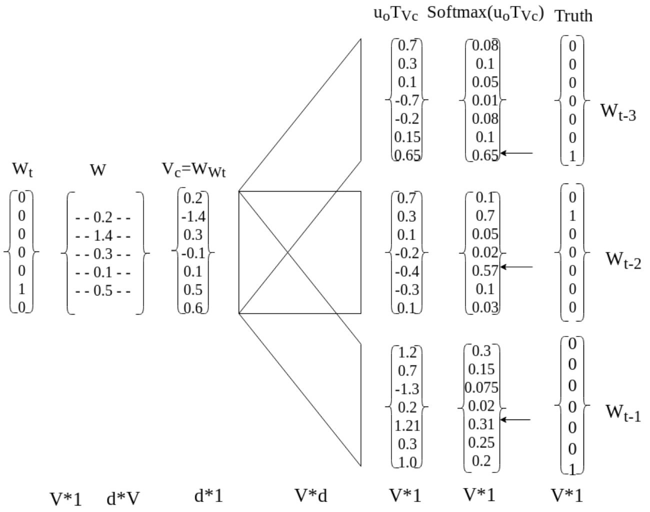

Fig. 2: Illustration of Skip-Gram Model

Fig. 2 gives an example to illustrate a skip gram model illustration. The input is in the form of one-hot encoding of the target or center word and output is the context words surrounding the target word. In order to train the model, all the vector gradients have to be computed. To accomplish this, all the parameter set is being put in a one long vector θ. The dimension of θ will be of the order $2dV$ as each word has two vectors u and v.

Gradient descent is used to optimize the parameters. From (2) and (3),

$$Error = log(\frac{exp(u_o^T v_c)}{\sum_{w=1}^{v} exp(u_w^T v_c)})$$

$$= \frac{\partial}{\partial v_c}(logexp(u_o^T) - log\sum_{w=1}^{v} exp(u_w^T v_c))$$

$$= u_o - [\frac{\partial}{\partial v_c} log \sum_{w=1}^{v} exp(u_w^T v_c)]$$

Now chain rule will be applied and it turns out that the backpropagation used in neural networks is nothing but the chain rule with some efficient storage of partial quantities so no need to calculate the same quantity again and again.

$$= u_o - [\frac{1}{\sum_{w=1}^{v} exp(u_w^T v_c)}(\frac{\partial}{\partial v_c}\sum_{x=1}^{v} exp(u_x^T v_c))]$$

$$= o - [\frac{1}{\sum_{w=1}^{v} exp(u_w^T v_c)}(\sum_{x=1}^{v}\frac{\partial}{\partial v_c} exp(u_x^T v_c))]$$

$$u_o - [\frac{1}{\sum_{w=1}^{v} exp(u_w^T v_c)}(\sum_{x=1}^{v} exp(u_x^T v_c)\frac{\partial}{\partial v_c} exp(u_x^T v_c))]$$

$$= u_o - [\frac{1}{\sum_{w=1}^{v} exp(u_w^T v_c)}(\sum_{x=1}^{v} exp(u_x^T v_c) u_x)]$$

$$= u_o - [\sum_{x=1}^{v}\frac{exp(u_x^T v_c)}{\sum_{w=1}^{v} exp(u_w^T v_c)}]$$

$$Error = u_o - (\sum_{x=1}^{v} P(X|C)u_x) \tag{4}$$

(4) suggests that it is working on probability on every possible word which is appearing in the context and based on the same probability, the term u_x will be consumed. In (4), the term u_o is the 'observed value' and the term

$$\sum_{x=1}^{v}\frac{exp(u_x^T v_c)}{\sum_{w=1}^{v} exp(u_w^T v_c)}$$

is the ‘expected value’. The aim is to optimize the parameters in such a way that the ‘observed value’ and the ‘expected value’ both are approximately same so as to minimize the loss function. Hence, the model optimizes its parameters.

The Continuous Bag of Words (CBOW) model can be considered as a mirror image of Skip-Gram model. CBOW model tries to predict the target word given the context word(s).

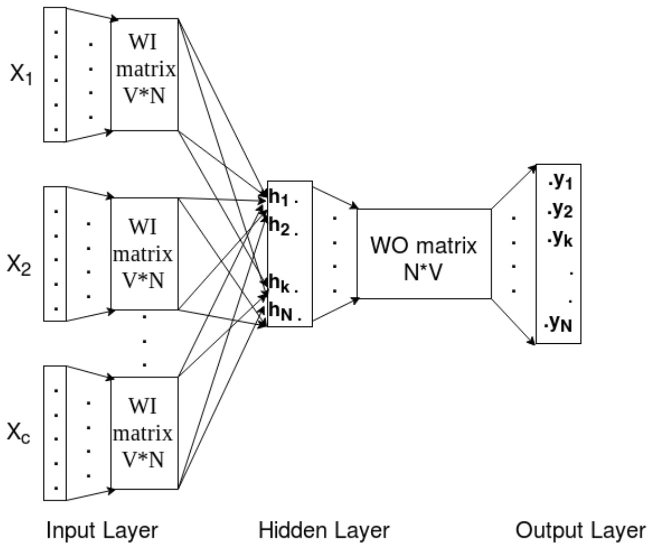

Fig. 3: Illustration of CBOW Model

The only way the CBOW model differs from skip gram model is the output is evaluated of the hidden layer which is computed by,

$$h = \frac{1}{C}W(\sum_{i=1}^{C} x_i) \tag{5}$$

(5) depicts the average of the input context vectors $[x_1, x_2, x_3, ..., x_n]$. The input to each node in the output layer is calculated by,

$$u_j = v_{w_j}^{'\ T}.h \tag{6}$$

In (6), $v_{w_j}^{'}$ is the j_{th} column extracted from the output matrix $W\prime$ Finally, the output y_i is obtained by passing u_j to the softmax function/layer given by,

$$y_i = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_j^{'})} \tag{7}$$

Henceforth, the parameter optimization can be carried on via back propagation algorithm as discussed in the derivation of skip-gram model.

Apart from Word2vec, Pennington et al. (2014) proposed another distributed word embedding technique and named it as ‘GloVe’ which stands for Global Vectors. This technique can be seen as an extension to Word2vec but GloVe significantly reduces the computational complexity by operating only on the number of non-zero elements in the input matrix. Unlike Word2vec which takes one-hot encoding vector as input, GloVe takes word-word co-occurrence matrix because the co-occurrence matrix is directly encoded by the co-occurrence counts. Instead of taking entire corpus as input, the GloVe takes the co-occurrence matrix. The GloVe claims that it outperforms other techniques but the techniques used to validate the results of GloVe is not as satisfactory as Word2vec. The performance in terms of accuracy of GloVe and Word2vec can be considered approximately similar but GloVe is computationally faster than Word2vec. Almost all the Machine Learning and Deep Learning techniques use Word2vec to generate vectors of words and then use them as input.

4 Paraphrase Identification Based Methods

The traditional similarity measures can find similarity between sentence pairs by taking individual words in consideration but not by taking the sentence and phrasal meaning as a whole. The task of Paraphrase Identification can be attempted by various techniques like Statistical Machine Translation (SMT) metrics, Machine Learning and Deep Learning techniques. A comprehensive survey is performed on the works proposed by different authors in each of the techniques. A flow is maintained between SMT, Machine Learning and Deep Learning approaches which point towards the evolution of Deep Learning algorithms to solve NLP problems.

4.1 Statistical Machine Translation Metrics

Statistical Machine Translation (SMT) metrics are used to evaluate the quality of machine translation from the source language to destination language in an automated fashion (Madnani et al. 2012, Marton et al. 2009, Madnani et al. 2007, Callison-Burch et al. 2006). Madnani et al. (2012) used these SMT metrics for the task of Paraphrase Identification. The main goal is to identify paraphrases only by training the meta-classifier using just machine translation metrics. In all, 8 different SMT metrics were used. The system makes use of only SMT metrics and no other external resources like WordNet for semantic equivalence or parsers are used to generate a syntactic parse tree. The technique uses a combination of classifiers by taking the average of all the probabilities of each classifier output and then making the final conclusion. Logistic regression, Support Vector Machines (SVM) and K-NN (K-Nearest Neighbours) are the three classifiers used. The different machine translation metrics employed are BLEU (Papineni et al. 2002), NIST (Madnani et al. 2012), TER (Abdul-Rauf & Schwenk 2009), TERp (TER Plus) (Abdul-Rauf & Schwenk 2009), METEOR (Agarwal & Lavie 2008), SEPIA (Habash & Elkholy 2008), BADGER (Parker 2008), MAXSIM (Chan & Ng 2008).

SMT Metric	Description
BLEU (Papineni et al. 2002)	Most common, n-gram overlap, exact matching
NIST (Madnani et al. 2012)	Arithmetic mean of n-gram overlaps, weights each n-gram according to frequency
TER (Abdul-Rauf & Schwenk 2009)	A heuristic algorithm to deal with shifts in addition to insertions, deletions and substitutions
TERp (Abdul-Rauf & Schwenk 2009)	Built on top of core TER and has stemming, synonymy and paraphrase.
METEOR (Agarwal & Lavie 2008)	A combination of both precision and recall. Incorporates stemming, synonymy (viaWordNet) and paraphrase
SEPIA (Habash & Elkholy 2008)	Focus on structural n-grams with long surface
BADGER (Parker 2008)	Language independent, uses a compression distance between sentence pair
MAXSIM (Chan & Ng 2008)	Treats the problem as one of bipartite graph matching

Table 3 Statistical Machine Translation Metrics

The model was first implemented by taking an individual SMT metric in which TERp performed well than others. TERp takes into consideration the stemming process with paraphrase recognition to accurately classify whether the two sentences are paraphrases of each other or not. Then, the combination of all the metrics was employed that resulted the accuracy to boost up by around 3% to 4% by considering Base Metrics (BLEU) (Papineni et al. 2002), NIST (Madnani et al. 2012), TER (Abdul-Rauf & Schwenk 2009)) in combination with TERp (Abdul-Rauf & Schwenk 2009), METEOR (Agarwal & Lavie 2008) and BADGER (Parker 2008). The task of Paraphrase Identification was utilized to improve the performance of machine translation problem. Paraphrases helped to tackle the 'problem of coverage' in SMT which fails to capture the unknown words which are not in model's vocabulary. The strategy deals with the 'problem of coverage' by first replacing the unknown source language words with their corresponding paraphrases and then translate these replaced paraphrases to the destination language.

TERp (Callison-Burch et al. 2010, Snover et al. 2009) surpasses the other individual SMT metrics more focus is on TERp so as to enlighten readers about the brief features and working of TERp. TERp uses stem matching, synonym matching and also paraphrase substitutions along with basic edit operations that are incorporated in TERp predecessors. TERp uses dynamic edit cost for different edit operations as opposed to editing cost as 1 for all edit operations in other metrics. The lower edit cost is assigned if two words are synonymous (uses WordNet), two words share same stem (Porter Stemming algorithm), two phrases are paraphrases of each other (using a paraphrase lookup table). The edit cost of phrase substitution is defined as a probability function and number of edits which are needed to align the paraphrase pair without the use of phrase substitution.

4.2 Supervised Machine Learning

Although Statistical Machine Translation (SMT) metrics are easy and straightforward to implement, they are less robust in handling data which contains misleading lexical overlap. Hence there arises the need to handle noisy data. Also, the task of Paraphrase Identification suffers from the problem of data scarcity due to lack of readily available data. Hence it is a major obstacle to develop a statistical machine translation based paraphrase models.

The Machine Learning techniques aim to be more robust against with the noisy data and to deal with data scarcity. Brockett & Dolan (2005) used annotated datasets and Support Vector Machine (SVM) so that larger monolingual corpora can be induced by using a corpus available from World Wide Web (WWW). The goal is to overcome the data sparsity problem by corpus construction. The alignment error rate can be drastically reduced by the use of the annotated dataset. The reason behind the usage of Support Vector Machine (SVM) is that, SVM is most robust against noisy data, get trained in a short amount of time, the robustness against the noisy data and the ability to deal with data sparsity.

The reason to use Support Vector Machine (SVM) is the SVM is most robust against noisy data, short training times for a comparable huge corpus and handling the sparse features which are normally encountered in natural language classification tasks. The main assumption made is that the early sentences or initial sentences provide a majority of the information of the entire article as compared with rest of the sentences part. If one sentence was a sub-string of another part, then it is interpreted as a paraphrase. The goal is to construct corpus which is non-application specific. To validate the immediate results, word-level alignment technique is used. Word-level alignment technique is an automated off-the-shelf implementation of SMT system GIZA++ (Casacuberta & Vidal 2007) which was created using human annotators. These human annotators categorized the alignments as either 'sure' or 'possible'. 'Sure' means necessary and 'allowed' means it is allowed but not required Och & Ney (2000). AER (Alignment Error Rate) can be stated by the equation as follows,

$$\frac{|A \cap P| + |A \cap S|}{|A + S|} \tag{8}$$

where A is the set of comparisons, P is the union of 'sure' and 'possible' alignments in the gold standard and S is a set of 'sure' alignments in gold standard.

Anther Machine Learning approach proposed by Kozareva & Montoyo (2006) attempts to accomplish the task of Paraphrase Identification (PI) using just lexical and semantic similarity. To improve the performance is terms of accuracy, a thorough examination is done on the combination of lexical and semantic similarity properties. The approach is somewhat similar to the previous approach (Brockett & Dolan 2005) but instead of corpus construction, the already compiled paraphrase corpus obtained from the work proposed by Quirk et al. (2004) is used. Hence, everything now boils down to the identification of paraphrases in which three classifiers namely, Support Vector Machine (SVM), K-Nearest Neighbours (K-NN) and Maximum Entropy (MaxEnt) were employed. The flow of the approach can be summarized as follows:

(i) Preprocessing or extracting features from text: Parts Of Speech (POS) tagger, WordNet

(ii) Inputting to Supervised Machine classifiers like SVM, K-NN, and MaxEnt

(iii) Evaluation of results.

In order to get the insights in the proximity of sentences, word-order is considered to be important. Also, a noun-verb similarity measure is employed. Initially, each algorithm out of SVM, KNN, and MaxEnt was implemented individually which did not contribute to much performance in terms of accuracy. SVM performed well in terms of achieving generalization. But when the voting scheme was used, the accuracy boosted by around 10%.

Lintean & Rus (2011) proposed a novel supervised machine learning approach on kernel dissimilarity kernels. The approach largely depends on lexical dissimilarity. SVM, when used in combination with lexical kernels achieve considerable performance in terms of training time and accuracy as SVM can handle a high number of features quite efficiently thereby taking less time to learn as compared to other approaches. Dual learning is used so as to handle high-dimensional data by computing dot product in an efficient way. The words are represented as n-dimensional vectors. The experiments were performed by considering only unigrams and bigrams. Bigrams (adjacent words) have almost 70% of inter-dependencies Collins (1996). In this approach, one-hot encoding technique is used to convert words to vectors. A kernel is then applied to these instances of vectors and the two similar dissimilarities are projected close to each other in the feature space.

$$K(A,B) = \sum_{\substack{w_1 \epsilon S_{A1} \Delta S_{A2} \\ w_1 \epsilon S_{B1} \Delta S_{B2} \\ w_1 \equiv w_2}} wt(w_1) * wt(w_2) \tag{9}$$

The kernel K between the instances of A and B of paired sentences (S_{A1}, S_{A2}) and (S_{B1}, S_{B2}) is defined by (9) where wt represents the weights. The sentences are represented by corresponding set of n-grams in which $S_{A1} \Delta S_{A2}$ is the symmetric difference between these two sets. $w_1 \equiv w_2$ represents identical n-grams. The worst case complexity for computing the kernel with the equation can go in the 'quadratic' order. This complexity can be lowered off the order 'linear' by performing the following steps:

1. Detecting all the n-grams which are different between two paired sentences in each instance by using quick-sort algorithm $O(nlog(n))$

2. After the comparison from the previous step, making sure that the detected differences are sorted in lexical order.

This linear time complexity is a major advantage in the kernel method and also the model can be easily interpretable by humans as the dimensions are related to words or sequence of words.

The supervised machine learning techniques performed well in some cases but in the task of finding semantic similarity using paraphrase identification, the aim is to find the complex relationships in a sentence in which the supervised Machine Learning algorithms failed to capture. In Paraphrase Identification, the complex interactions have to be captured at the word-level, phrasal-level, and sentence-level and thus deep learning techniques came into use. The Deep Learning techniques, when invented, were used in computer vision (LeCun et al. 2015, Krizhevsky et al. 2012, Koch et al. 2015) to extract meaningful features from the image pixels represented in the form of a matrix. But, due to the availability of high processing power, Deep Learning is gaining popularity in Natural Language Processing (NLP) applications due to its efficiency in capturing the features (feature matrix) at various levels of granularity.

4.3 Neural Networks and Deep Learning Architectures

The state-of-the-art shallow Neural Network architectures were first used from 2003 (Bengio et al. 2003) to model natural language. There are two significant works based on Neural Networks proposed by Bengio et al. (2003) and Collobert & Weston (2007) which builds a neural network based probabilistic model and deals with semantic labeling problem respectively.

Neural Networks are used to generate the distributed representation of words (Bengio et al. 2003). The goal is to reduce the high dimensionality of when a model is tried to learn the joint probability function of sequences of words in a particular language. The distributed representation of words allows the training sentence to tell the model about an exponential number of neighboring sentences which are semantically related, thereby helping to reduce the dimensionality. The model deals with *out of vocabulary words* and can also assign them some probability as these new combinations cannot be assigned zero probability because there's a high chance that they are likely to occur more frequently for larger context sizes. Due to the huge size of training corpus and a large number of training parameters, the computations take huge time to finish the training of the model. The obtained distributed representation of words (word vectors) are as it feed to linear classification layer and then the results were obtained.

A neural network model proposed by Collobert & Weston (2007) deals with the problem of semantic labeling with significant improvement in computation times. The previous method (Bengio et al. 2003) took huge computation time mainly due to huge amount training parameters which further increased as per increase in the amount of training data. To overcome this problem an approach was proposed which surpasses the need of building the parse tree and by using explicitly engineered features. The result of this combination is a significant improvement in computation cost to just 0.02 seconds for each instance to train. The work can be considered as an extension to improve the accuracy obtained by the previous work proposed by Bengio et al. (2003). Instead of feeding the word vectors directly to a linear classifier, the semantic role label of a word at a particular position pas_w is predicted by considering the relative position of the verb at pas_v. As soon as the accurate semantic labels are identified with the help of semantic labeling, the extracted labels like the actor performing an action, target on which the action is performed, location, the reason for performing an action can be used to find the semantic similarity between the sentences.

Deep Learning can also be called as deep structured learning or hierarchical learning. Deep Learning methods started to gain popularity from the year 2006 onwards. But the concept of deep learning was first applied to Natural Language in the year 2011 by 2011 wherein Recursive AutoEncoder (RAE) is used to model the language (Socher et al. 2011). First, a syntactic tree (binary tree) is built of which the leaf nodes represent the vector representation of individual words. The parent nodes represent the phrases of n-grams. At each parent node, the concatenation of individual word vectors takes place which is then multiplied by an encoding matrix, a bias is added and finally a non-linear activation function (tanh, sigmoid, etc) is applied giving rise to **p** which will now represent the phrases in respective tree level. The same process can be depicted by using the equation,

$$p = f(W_e\,[x_2; x_3]) + b_e \tag{10}$$

The reconstruction of original word vector can be thought of as multiplying]p with decoding matrix W_d, adding decoding bias to it and finally applying the same non-linear activation function.

$$[x_1; y_1] = f(W_d y_2 + b_d) \tag{11}$$

To deal with variable length sentences, dynamic max-pooling techniques are used to create a pooled matrix (Spooled) of fixed size in order to be inputted to a classifier. Spooled even though loses some information, it successfully captures the majority of the important information of its global structure. Hence, RAE captures word-level as well as phrase-level information which greatly affects the final performance in finding semantic similarity using paraphrase identification as RAE helps to maintain the context information. RAE is advantageous and has more parameters to learn the complex encodings but during training, RAE gets stuck in local optima.

The vector representations of words were directly inputted to Convolutional Neural Networks by using dynamic k-max pooling (Kalchbrenner et al. 2014). Unlike RAE Socher et al. (2011) which uses parse tree, the approach does not make of any parse tree which makes it language independent. The approach consists of the following steps:

(i) Compute word embeddings.

(ii) Apply wide convolution .

(iii) Apply k-max dynamic pooling on the result of previous step.

(iv) Apply non-linear feature function in combination with pooling to induce position invariance and to make range of higher order features variable.

(v) Output of previous three steps gives one feature map.

(vi) Repeat these steps to obtain multiple feature maps(increasing order) and a network of increasing depth.

The highlight of this work is the usage of dynamic k-max pooling by using the equation,

$$k_l = max(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil) \tag{12}$$

where, s is the length of input sentence, l is the current level of convolutional layer k_{top} value is fixed which is the top most convolution layer, L is the total number of convolutional layers.

l	k_l
2	10
3	5
4	4

Table 4 k-max pooling values for sentence length of 20 and total convolution layers of 6

The above table depicts the dynamically changing values for k-max pooling with respect to the level of the current convolutional layer. The above equation describes a model of a number of values which are needed in order to describe which relevant parts to extract as the order l progresses over a variable sentence length s. The final model was applied to in the tasks namely, binary and multi-class sentiment analysis of movie results, twitter sentiment analysis, and six-way question classification. The resulting conclusion is dynamic k-max pooling can improve accuracy even though the external resources like WordNet (Perkins 2010, Loper & Bird 2002, Pedersen et al. 2004, Agirre et al. 2009), etc is not applied and also with no pre-training. Also, the sequence of steps as described makes the model hard to learn as the input of many *tanh* units were close to 1 after dynamic k-max pooling. In short, the *tanh* units became saturated.

Hu et al. (2014) proposed couple of Convolutional Neural Network architectures which can be dubbed as ARC-I and ARC-II. This architecture also represents the natural language sentences at various granularity levels. The architecture differs from the work of RAE Socher et al. (2011) in two distinct ways. RAE needs an external parser to construct binary syntactic tree which decides the path of a word or phrase composition whereas the model gives the max pooling layer the authority to pick up the most important segments from a large feature map. Also, in RAE, the depth of the tree is not fixed and keeps on changing as per input sentence structure. But the model's depth is kept fixed which keeps a bound on what level till it can perform decomposition. ARC-I is a simple architecture where the words of sentences are converted to vectors and are directly fed as input to Multi-Layer Perceptron (MLP). The problem with this approach is there's a very high risk of losing the details which is not desirable when the task is to get semantic similarity. ARC-II is quite familiar. A similarity matrix is calculated between the word embeddings of the pair of sentences, then a series of convolution and pooling is applied. The ARC-II was then implemented to perform tasks like matching, Sentence Completion, tweet response matching.

The features are extracted at different granularity levels in the work proposed by He et al. (2015). The model differentiates itself by making use of different types of pooling namely *max, min* and *avg.* The model is divided into two parts namely sentence model to convert a sentence to an appropriate representation using CNN and similarity measurement layer to calculate similarity using different similarity measures. Two networks with shared weights are incorporated. The highlight of this architecture is the holistic filter and a per-dimension filter. A sentence can be represented as $length * dimension$ matrix.The holistic filter concept can be depicted in the following equation:

$$out_F\left[i\right] = h_F(w_F * sent_{i:+ws-1} + b_F) \tag{13}$$

The filter size F is applied to the input sequence $sent$, then the inner product is calculated between w_F and each possible window size w_s. The per-dimension filter is a new concept where individual the match is calculated against each dimension of word embedding independently. Combining these two filters yield better representations which extract more refined features. The holistic filter can be viewed as performing 'temporal convolution' while, per-dimension filter can be viewed as 'spatial convolution'. Also, there's no need for any external resources like WordNet or parsers.

The previous work (He et al. 2015) considered the granularity only at the word level. The words were converted to vectors and concatenation of words is done to represent sentences which is then fed to the sequence of steps as discussed. In order to get insights in Paraphrase Identification task, a sentence has to be represented by multiple levels of granularity. The concept of parsing was introduced by Socher et al. (2011) to construct binary parse trees. But parsing doesn't perform well when the data is noisy and hence it is not suitable for the applications where there is a requirement of highly accurate parsers. The work proposed by Yin & Schütze (2015) can be considered as an extension of RAE. Instead of generating phrasal vector representation at parent nodes by considering the leaf nodes in RAE, Yin et al used sliding context windows to split sentences into phrases. The process when repeated, captures longer phrases by combining small phrases. Also, the process followed is closely related to the work proposed by Hu et al. (2014) The aim is to use Bi-CNN-MI which stands for Binary (two)- Convolutional Neural Network-Multigranular Interaction features and Logistic Regression final classification layer. Idea is to use pair of Neural Networks with shared weights. To cope up with the problem of saturation of tanh units (Socher et al. 2011), the sequence was slightly modified from convolution, folding, tanh to convolution, averaging, tanh, k-max pooling which doesn't saturate tanh units. It uses the same wide convolution and dynamic k-max pooling which was used by previous work. The additional part is the averaging layer sandwiched between the wide convolutional layer and dynamic k-max pooling layer. The averaging layer is introduced to consider the simple relations across rows. For this, the average takes place between each odd rows and the immediate rows behind it. The resulting matrix is of the size $\frac{d}{2} \times (|S_i| + m - 1)$ where, S_i is the number of tokens in input sentence, d is the word embedding dimension and m is the filter width. The architecture of each convolutional neural network is a follows:

- First Block
  - (i) Convolutional Layer
  - (ii) Averaging
  - (iii) Dynamic k-max pooling
- Top Block

(i) Convolutional Layer
(ii) Averaging
(iii) Dynamic k-max pooling

- Concatenation of four matrices namely,

(i) F_s : Sentence Level Feature Matrix
(ii) F_{ln} : Long n-gram Feature Matrix
(iii) F_{sn} : Short n-gram Feature Matrix
(iv) F_u : Unigram Feature Matrix

The Bi-CNN-MI can be summarized into an architecture having three parts namely, CNN-SM (Sentence Model) consisting of pair of shared weights CNN, CNN-IM (Interaction Model) where the four interaction matrices or feature matrices $(F_s, F_{ln}, F_{sn}, F_u)$ are computed at various levels of granularity and Logistic Regression on top of the network for prediction. The feature matrices are computed by using Euclidean distance and not by Cosine Similarity measure which is normally used (between vectors) because in this case, the magnitude of hidden units is more important than just the direction. The aim of generating various matrices at different levels is to compute the semantic relatedness between the pair of paraphrases on multiple levels of granularity. Another important feature is the unsupervised pretraining of word embeddings. The task of PI is challenging due to less available data which causes the problem of data sparsity and overfitting. In order to overcome the problem of data sparsity and overfitting, a large corpus (English Gigaword) is chosen and the CNN model is trained on MSRP data plus the English Gigaword corpora consisting of approximately 1,00,000 sentences which significantly improved the accuracy of the model. There are two phases of learning: supervised (language modeling approach) and unsupervised (Bi-CNN-MI) as opposed to other work which has to only train in a supervised manner. Due to the variable length of sentences, the computed feature matrices are also of variable length which makes it impossible to use as the input in the final layer. To solve this problem, again dynamic k-max pooling is used to transform all the four feature matrices to equal size depending on sentence length of both the input sentences.

The interactive representation of sentences on the word level, phrase level and also on sentence level was attempted by Zhang et al. (2017). It aims to find the interactive representation of sentences on the word level, phrase level and also on the sentence level. The word embeddings are generated by using Word2vec tool or GloVe or any other suitable word to vector tool. The modules in this architecture include Recursive Auto Encoder (RAE), Convolutional Neural Networks (CNN) and final classification layer. The important aspect is the combination of RAE (Socher et al. 2011) to get the vector representation at the word, phrases and sentence level and then employing CNN to conduct Paraphrase Identification (PI) task. The model makes use of semantic and syntactic features of the sentence. The flow can be summarized as follows:

(i) Convert word to vectors (Represent words to low dimensional space)

(ii) Obtain phrase and sentence representation by using RAE

(iii) Construct 3 types of similarity matrices to get complex relationships between sentences.

  (a) Word Embedding (WE) + tile + Number Features (NF)
  (b) WE + RAE + tile + NF
  (c) WE + RAE + pooling + NF

RAE can be considered having two major parts namely, folding and unfolding (reconstruction). In folding part, the vector representation at parent node is computed using the leaf nodes from the binary syntactic tree constructed using external parser. The folding part can be represented by the equation,

$$y_1 = f(W_e\,[x_2; x_3] + b_e \tag{14}$$

where W_e is encoding matrix whose values initialised to small random values, $x_2 and x_3$ are the concatenation of two vector representation of words given as input and b_e is the encoding bias. Sigmoid or tanh activation function can be used. The unfolding or reconstruction part can be represented by the equation,

$$[x_1; y_1] = f(W_d y_2 + b_d) \tag{15}$$

During folding or encoding phase, the inner product is computed between the encoding matrix W_e and the concatenated representation of word vectors. An encoding bias b_e is added to the result thus obtained and finally an activation function is applied to get the phrasal or sentence vector representation at the parent level. The decoding or reconstruction process can be related with the encoding process where the inner product is computed between the decoding matrix W_d and the encoding vector y_2, decoding bias b_d is added to the result and then an activation function is applied to get the reconstruction of original word or phrase.
It is observed that using the matrix with high granularity level (WE (Word Embeding or encoding matrix) + tile (fixed sized matrix) + pooling + NF (Number of Features in CNN) yields better result as it captures more complex interaction features. This differs from previous work by using efficient usage of step by step modules to get more complicated relationships. All the modules (Word to vectors, RAE, CNN) contribute in vector representation except final prediction layer. The figure depicts the encoding and decoding process for three word embeddings namely x_1, x_2, x_3 Three

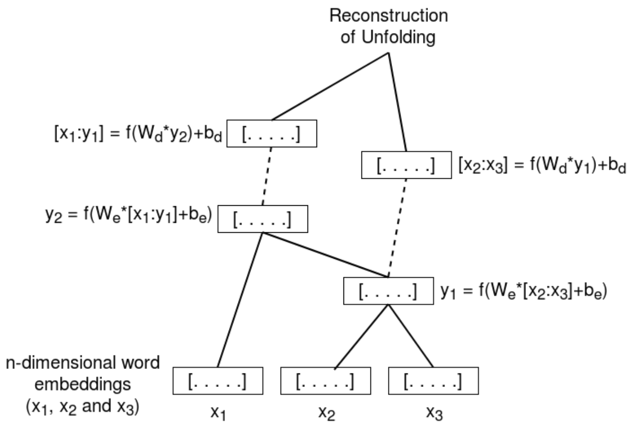
Reconstruction of Unfolding
$[x_1:y_1] = f(W_d*y_2)+b_d$
$[x_2:x_3] = f(W_d*y_1)+b_d$
$y_2 = f(W_e*[x_1:y_1]+b_e)$
$y_1 = f(W_e*[x_2:x_3]+b_e)$
[. . . . .]
n-dimensional word embeddings (x_1, x_2 and x_3)
x_1
x_2
x_3

Fig. 4: RAE Encoding and Decoding

machine learning prediction algorithms namely, Logistic regression, Multi Layer Perceptron (MLP) and CNN were used whose inputs are one of the three computed final matrices.

5 Comparative Analysis and Discussion

The comparative analysis is conducted between,

(i) Word embedding Approaches.

(ii) Traditional, SMT metrics, Machine Learning and Deep Learning techniques.

(iii) Parameters used in Machine Learning and Deep Learning approaches.

The comparative analysis starts with comparing various types of word embedding techniques. Then a brief high-level comparison is done between various techniques namely, traditional, SMT metrics, Machine Learning and Deep Learning approach used to solve the task of PI which is followed by specifying advantages and disadvantages/limitations of each work which is proposed in literature survey by respective authors. Finally, an in-depth parametric comparative analysis of Machine Learning and Deep Learning techniques is conducted, as the Machine Learning and Deep Learning techniques are widely used currently and outperforms other techniques.

5.1 Word Embedding Approaches

A comparative analysis of word embedding techniques is conducted in Table 5 with respect to type in which these methods belong, whether they capture semantics and contexts, computational time and memory requirement. The prediction-based word embeddings (distribute word representations) are more preferred in NLP tasks because they preserve the context of the words in which they appear. Most widely used word embedding approaches in PI task is word2vec model.

One-hot encoding technique is easy to compute and also has low computational time. But it faces the curse of dimensionality as the number of vocabulary size increases. Also, it fails to capture the context and semantics. Count vector technique is simple to implement and considers a single document at a time and not the entire corpus thereby failing to capture semantics and context. TF-IDF overcomes limitations of count vector and considers the entire corpus. The co-occurrence matrix makes use of Singular Value Decomposition (SVD) which generates more accurate vectors. Large memory is required to store the co-occurrence matrix which limits its practical usage. Word2vec CBOW model is better than deterministic models as it is probabilistic in nature. Also, the memory required is less. If the training is started from scratch, then the model may take huge computation time. Word2vec Skip-gram model computed two different semantics for same word occurring in the different context. Word2vec is known to train in separate local context rather than at a global level. Hence, it poorly utilizes the corpus statistics.

5.2 Paraphrase Identification Techniques

Table 6 depicts the advantages and limitations of traditional, SMT metrics, Machine Learning and Deep Learning techniques to be used for the task of PI. The aim of Table 6 is to get a high-level understanding of all the techniques before proceeding to attempt the task of PI. It gives a rough idea of the complexity of the model and the computational time that might be required to train the model. This table also states the challenges faced by each technique and to which granularity level we want to proceed with the Paraphrase Identification problem.

5.3 Comparative Analysis with respect to Parameters

The in-depth comparative analysis with respect to parameters is done in Table 7 where the Machine Learning and Deep learning algorithms are compared by considering various parameters like Granularity Level, Final Classification Layer, Architecture, Activation Function, word embedding dimension, Context Window Size, Pre-training and Datasets used for the same and dataset used for experiments (training and testing).

The granularity level is one of the important parameters as it represents the various levels of representation with respect to words, phrases, and sentences. In the task of paraphrase Identification, there is a necessity to represent the natural language in the form of learned representations by considering words, phrases as well as sentences. Final classification layer helps to classify the sentence pair as paraphrases or not. Here, the binary classification takes place. The final classification layer has to be rightfully coupled

Type of Approach	**Techniques**	**Type**	**Capture Semantics**	**Capture Context**	**Memory Requirement**	**Computational Time**
Discrete Word Representations	One-Hot Encoding	Frequency based	No	No	Low	Low
	Count Vector	Frequency based	No	No	Low	Low
	TF_IDF	Frequency based	No	No	Low	Low to Moderate
	Co-occurence Vector	Frequency based	No	Yes	High	Low to Moderate
Distributed Word Representations	Word2vec CBOW model	Probability Based	Yes	Yes	Low to Moderate	Moderate to High
	Word2vec Skip-Gram model	Probability Based	Yes	Yes	Low to Moderate	Moderate to High

Table 5 Word Embedding Approaches Comparative Analysis

Approach	**Highlights**	**Shortcomings**
Traditional Measures (String-based, Corpus-based, Knowledge-based)	• Simple and straightforward to implement. • Computation is comparatively fast. No need for separate training.	• Only the most similar words or tokens in the other sentence are considered. • No consideration of context. • No Phrasal consideration.
Machine Translation Metrics	• Solely based on Machine Translation Metrics which makes it easier and straightforward. • Availability of various metrics with features of stemming, lemmatisation, synonyms and upto certain point phrasal consideration.	• Less robust in handling misleading lexical overlap
Supervised Machine Learning	• Using annotated datasets plus SVM to induce a larger monolingual corpora from a news clusters corpus. • Large training data means improved performance. • Combination of lexical and semantic properties along with voting provided a boost in accuracy.	• Captures granularity only at word level, thus failing to capture phrases. • Fails to capture information at multiple levels of granularity.
Deep Learning	• Successfully captures information at multiple levels of granularity (Feature Matrices). • Makes use of efficient convolution and pooling layers to extract important features	• Required amount of training data is high for a model to efficiently learn. • Data sparsity due to lack of training data. • Variable Length input data has to be handled carefully while applying kernel filters.

Table 6 Advantages and Limitations of each approach

with the activation functions and other parameters used in hidden computation layers. There are various Deep Learning Neural Network architectures that can be tried to approach the problem. It is not the case that one architecture is universally superior to others. The architectures are decided as per the trial and error techniques and vary from problem to problem. There are lots of choices to choose the activation functions. Again it depends on trial and error to choose the activation function which provides high performance in terms of accuracy. The context size and dimension of word embeddings are important parameters while generating the vector representation of words and the output from this step may affect the overall accuracy. Context window size can be considered as a radius around a target word. If the context size is of two, then the two word preceding and two words succeeding to the target words are considered. The context word size has to be chosen carefully as the small window cannot efficiently capture the context and the large window size increases the computational time. The dimension of word embeddings is the number of features of each

Work	**Granularity Level**	**Final Classification Layer**	**Architecture**	**Parameters, Keyterms, Activation function**	**Dimensions of word embeddings**	**Context Window Size**	**Is Pre-training applied?**	**Dataset(s) used for pre-training**	**Dataset used in Experiments**
Bengio et al. (2003)	Word	Softmax	Multi Layer Perceptron with two hidden layers	Learning rate: 0.001 tanh	100	3	No	NA	Brown Corpus
Collobert & Weston (2007)	Word	Softmax	Multi Layer Perceptron with two hidden layers	Position of verb w.r.t. each other word. Learning rate: 0.001 tanh	100	5	No	NA	PropBank version 1
Socher et al. (2011)	Sentence, Phrase, Word	Softmax	Folding and Unfolding Recursive AutoEncoding (RAE)	Pooling size - 15 Binary Parse tree Encoding Decoding Reconstruction	100	5	Yes	NYT and AP sections of the Gigaword Corpus	MSRP
Kalchbrenner et al. (2014)	Word	Softmax	CNN with dynamic K-max pooling	For Question Type Classification: 2 convolutional layer with 8 and 5 feature maps respectively k-max pooling: 4, tanh	32	5	No	NA	TREC Questions dataset
Yin & Schütze (2015)	Sentence, Phrase, Word	Logistic Regression	Bi-CNN-MI (Double CNN-Multi Granular interaction features)	Feature Maps: 6, 14 Filter Width: 3,5 Pooling: 10, 6 tanh	100	5	Yes	English Gigaword and MSRP	MSRP
Hu et al. (2014)	Word	MLP	ARC I (Siamese architecture) and ARC II.	8 layers (3 convolution, 3 pooling, and 2 MLP), ReLu	50	3	Yes	English-Wikipedia Chinese-Weibo	MSRP
He et al. (2015)	Word	Softmax	CNN with similarity measurement layer with min, max and mean pooling, Filter size= 1,2,3, infinity (no convolution)	Number of Holistic dimension: 525 Number of Per-dimension: 20, tanh	525 (GloVE: 300-dimension +PARAGRAM: 25) +POS: 200))	2	Yes	Pennington	MSRP
Zhang et al. (2017)	Sentence, Phrase, Word	Softmax, Logistic Regression, MLP, CNN	Vector representation by RAE, CNN to find complex relationships, Supervised Machine Learning for Prediction	Number of featuremaps: 5Number of hidden layers in MLP: 2tanh	100	5	No	NA	MSRP
Brockett & Dolan (2005)	Word	SVM	Sequential Minimal Optimization (SMO)	Based on Sentence Length. Levenshtein (all), First two and First three	NA	All, First two, First three of the sentences	Yes	News clusters from World Wide Web	10,000 sentence pairs randomly extracted from News clusters
Kozareva & Montoyo (2006)	Word	SVM, Max-Entropy, K-NN	Voting Scheme to resolve the conflicts between multiple classifiers	Word overlap feature set, combination of lexical and semantic information in a single feature set, voting scheme	NA	NA	No	NA	MSRP
Lintean & Rus (2011)	Word	SVM	Kernel Dissimilarity approach, Dual Learning method (Linear Kernels in conjunction with SVM)	Unigram, bi-gram kernels, punctuation, stemming	NA	Unigram and Bi-gram (1 and 2)	No	NA	MSRP

Table 7 Comparative Analysis with respect to Parameters

word which is now converted in its vector form. The pre-training also plays an important role in the case where the model lacks the training data. In such cases, a large corpus is chosen which is used to get the vector representation of words to identify the context in which the word occurs. There is various publicly available corpus available for pre-training such as English Gigaword, English Wikipedia, Chinese Weibo, Pennington, News clusters from World Wide Web (WWW), etc. The most widely and commonly used dataset is Microsoft Research Paraphrase Corpus Dolan & Brockett (2005) consisting of total 5800 labeled sentence pair. The other datasets which can be used for training and testing are Brown Corpus, PropBank version 1, Stanford Sentiment Treebank, News Clusters from WWW, etc. The right combination of all these parameters for Machine Learning and Deep Learning significantly improves the performance in terms of accuracy in the task of Paraphrase Identification.

6 Conclusion

Finding semantic similarity is a vital task in NLP applications that can be accomplished by identifying the paraphrases. Semantic similarity using Paraphrase Identification can be used in a wide range of NLP applications like Question-Answering, Information Retrieval, Recommendation Engine, Machine Translation, Summarization, etc. In order to identify the paraphrases in sentences, n-gram (n>1) words in a sentence has to be considered. The traditional measures which are largely dependent on individual words fail to consider n-grams. Supervised Machine Learning techniques solved the problem to some extent but they struggled to represent the data at multiple levels. This gave rise to Deep Learning which is based on Representation Learning. Deep Learning has the ability to represent the data through various levels of granularity (word level, phrase level, and sentence level). Deep Learning algorithms such as Convolutional Neural Networks and Recurrent Neural Networks outperformed traditional Machine Learning methods because of the property of Sparse Interactions, Parameter Sharing and Equivariant Representation. These three important properties boosted the statistical efficiency of Deep Learning techniques for finding the semantic similarity with the help of Paraphrase Identification which in turn improved accuracy. There are various architectures that can be tried and tested with Convolutional Neural Networks with varying complexity levels. Also, the output representation given by Deep Learning approaches can effectively be combined with traditional supervised Machine Learning techniques to gain better classification accuracy. The input to be given to these Deep Learning approaches have to be in a vector form. Hence, 'Word to vector' step is also important to get the most out of Deep Learning techniques. The initial step where the vectors are generated of each word is crucial since the meaning and context has to be preserved. Then the next important step is how the features are extracted at each step. The aim should be to achieve three types of granularity which is at word-level, phrase-level and at sentence-level. There is not a single superior architecture in terms of the number of layers and other parameters available for Machine Learning and Deep Learning.

References

Abdul-Rauf, S. & Schwenk, H. (2009), Exploiting comparable corpora with ter and terp, *in* 'Proceedings of the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora', Association for Computational Linguistics, pp. 46–54.

Agarwal, A. & Lavie, A. (2008), Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output, *in* 'Proceedings of the Third Workshop on Statistical Machine Translation', Association for Computational Linguistics, pp. 115–118.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M. & Soroa, A. (2009), A study on similarity and relatedness using distributional and wordnet-based approaches, *in* 'Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics', Association for Computational Linguistics, pp. 19–27.

Agirre, E., Banea, C., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., Mihalcea, R., Rigau, G. & Wiebe, J. (2016), Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation., *in* 'SemEval@ NAACL-HLT', pp. 497–511.

Barrón-Cedeño, A., Vila, M., Martí, M. A. & Rosso, P. (2013), 'Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection', *Computational Linguistics* **39**(4), 917–947.

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

Bouma, G. (2009), 'Normalized (pointwise) mutual information in collocation extraction', *Proceedings of GSCL* pp. 31–40.

Brockett, C. & Dolan, W. B. (2005), Support vector machines for paraphrase identification and corpus construction, *in* 'Proceedings of the 3rd International Workshop on Paraphrasing', pp. 1–8.

Callison-Burch, C., Koehn, P., Monz, C., Peterson, K., Przybocki, M. & Zaidan, O. F. (2010), Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation, *in* 'Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR', Association for Computational Linguistics, pp. 17–53.

Callison-Burch, C., Koehn, P. & Osborne, M. (2006), Improved statistical machine translation using paraphrases, *in* 'Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics', Association for Computational Linguistics, pp. 17–24.

Casacuberta, F. & Vidal, E. (2007), 'Giza++: Training of statistical translation models'.

Chan, Y. S. & Ng, H. T. (2008), Maxsim: A maximum similarity metric for machine translation evaluation., *in* 'ACL', pp. 55–62.

Collins, M. J. (1996), A new statistical parser based on bigram lexical dependencies, *in* 'Proceedings of the 34th annual meeting on Association for Computational Linguistics', Association for Computational Linguistics, pp. 184–191.

Collobert, R. & Weston, J. (2007), Fast semantic extraction using a novel neural network architecture, *in* 'Annual meeting-association for computational linguistics', Vol. 45, p. 560.

De Marneffe, M.-C., Rafferty, A. N. & Manning, C. D. (2008), Finding contradictions in text., *in* 'ACL', Vol. 8, pp. 1039–1047.

Dolan, W. B. & Brockett, C. (2005), Automatically constructing a corpus of sentential paraphrases, *in* 'Proc. of IWP'.

Duclaye, F., Yvon, F. & Collin, O. (2003), Learning paraphrases to improve a question-answering system, *in* 'Proceedings of the EACL Workshop on Natural Language Processing for Question Answering Systems', pp. 35–41.

Fader, A., Zettlemoyer, L. & Etzioni, O. (2013), Paraphrase-driven learning for open question answering, *in* 'Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)', Vol. 1, pp. 1608–1618.

Gabrilovich, E. & Markovitch, S. (2007), Computing semantic relatedness using wikipedia-based explicit semantic analysis., *in* 'IJcAI', Vol. 7, pp. 1606–1611.

Goldberg, Y. & Levy, O. (2014), 'word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method', *arXiv preprint arXiv:1402.3722* .

Gomaa, W. H. & Fahmy, A. A. (2013), 'A survey of text similarity approaches', *International Journal of Computer Applications* **68**(13).

Gonzalo, J., Verdejo, F., Chugur, I. & Cigarran, J. (1998), 'Indexing with wordnet synsets can improve text retrieval', *arXiv preprint cmp-lg/9808002* .

Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T. & Saenko, K. (2013), Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 2712–2719.

Habash, N. & Elkholy, A. (2008), Sepia: surface span extension to syntactic dependency precision-based mt evaluation, *in* 'Proceedings of the NIST metrics for machine translation workshop at the association for machine translation in the Americas conference, AMTA-2008. Waikiki, HI'.

Harabagiu, S. M., Maiorano, S. J. & Paşca, M. A. (2003), 'Open-domain textual question answering techniques', *Natural Language Engineering* **9**(3), 231–267.

He, H., Gimpel, K. & Lin, J. J. (2015), Multi-perspective sentence similarity modeling with convolutional neural networks., *in* 'EMNLP', pp. 1576–1586.

Hu, B., Lu, Z., Li, H. & Chen, Q. (2014), Convolutional neural network architectures for matching natural language sentences, *in* 'Advances in neural information processing systems', pp. 2042–2050.

Islam, A. & Inkpen, D. (2006), Second order co-occurrence pmi for determining the semantic similarity of words, *in* 'Proceedings of the International Conference on Language Resources and Evaluation', pp. 1033–1038.

Kalchbrenner, N., Grefenstette, E. & Blunsom, P. (2014), 'A convolutional neural network for modelling sentences', *arXiv preprint arXiv:1404.2188* .

Koch, G., Zemel, R. & Salakhutdinov, R. (2015), Siamese neural networks for one-shot image recognition, *in* 'ICML Deep Learning Workshop', Vol. 2.

Kozareva, Z. & Montoyo, A. (2006), Paraphrase identification on the basis of supervised machine learning techniques, *in* 'FinTAL', Springer, pp. 524–533.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Landauer, T. K., Foltz, P. W. & Laham, D. (1998), 'An introduction to latent semantic analysis', *Discourse processes* **25**(2-3), 259–284.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Levy, O. & Goldberg, Y. (2014), Dependency-based word embeddings., *in* 'ACL (2)', pp. 302–308.

Lintean, M. C. & Rus, V. (2011), Dissimilarity kernels for paraphrase identification., *in* 'FLAIRS Conference'.

Loper, E. & Bird, S. (2002), Nltk: The natural language toolkit, *in* 'Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1', Association for Computational Linguistics, pp. 63–70.

Madnani, N., Ayan, N. F., Resnik, P. & Dorr, B. J. (2007), Using paraphrases for parameter tuning in statistical machine translation, *in* 'Proceedings of the Second Workshop on Statistical Machine Translation', Association for Computational Linguistics, pp. 120–127.

Madnani, N., Tetreault, J. & Chodorow, M. (2012), Re-examining machine translation metrics for paraphrase identification, *in* 'Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies', Association for Computational Linguistics, pp. 182–190.

Marton, Y., Callison-Burch, C. & Resnik, P. (2009), Improved statistical machine translation using monolingually-derived paraphrases, *in* 'Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1', Association for Computational Linguistics, pp. 381–390.

Matsuo, Y. & Ishizuka, M. (2004), 'Keyword extraction from a single document using word co-occurrence statistical information', *International Journal on Artificial Intelligence Tools* **13**(01), 157–169.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'Advances in neural information processing systems', pp. 3111–3119.

Mollá, D. & Vicedo, J. L. (2007), 'Question answering in restricted domains: An overview', *Computational Linguistics* **33**(1), 41–61.

Och, F. J. & Ney, H. (2000), Improved statistical alignment models, *in* 'Proceedings of the 38th Annual Meeting on Association for Computational Linguistics', Association for Computational Linguistics, pp. 440–447.

Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. (2002), Bleu: a method for automatic evaluation of machine translation, *in* 'Proceedings of the 40th annual meeting on association for computational linguistics', Association for Computational Linguistics, pp. 311–318.

Parker, S. (2008), 'Badger: A new machine translation metric', *Metrics for Machine Translation Challenge* pp. 21–25.

Pedersen, T., Patwardhan, S. & Michelizzi, J. (2004), Wordnet:: Similarity: measuring the relatedness of concepts, *in* 'Demonstration papers at HLT-NAACL 2004', Association for Computational Linguistics, pp. 38–41.

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

Perkins, J. (2010), *Python text processing with NLTK 2.0 cookbook*, Packt Publishing Ltd.

Quirk, C., Brockett, C. & Dolan, W. (2004), Monolingual machine translation for paraphrase generation, *in* 'Proceedings of the 2004 conference on empirical methods in natural language processing'.

Ramos, J. et al. (2003), Using tf-idf to determine word relevance in document queries, *in* 'Proceedings of the first instructional conference on machine learning', Vol. 242, pp. 133–142.

Robertson, S. (2004), 'Understanding inverse document frequency: on theoretical arguments for idf', *Journal of documentation* **60**(5), 503–520.

Royer, C. (2005), 'Term representation with generalized latent semantic analysis', *Recent advances in natural language processing IV: selected papers from RANLP* **292**, 45.

Sekine, S. & Ranchhod, E. (2009), *Named entities: recognition, classification and use*, Vol. 19, John Benjamins Publishing.

Shinyama, Y., Sekine, S. & Sudo, K. (2002), Automatic paraphrase acquisition from news articles, *in* 'Proceedings of the second international conference on Human Language Technology Research', Morgan Kaufmann Publishers Inc., pp. 313–318.

Sinha, R. & Mihalcea, R. (2007), Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity, *in* 'Semantic Computing, 2007. ICSC 2007. International Conference on', IEEE, pp. 363–369.

Snover, M. G., Madnani, N., Dorr, B. & Schwartz, R. (2009), 'Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate', *Machine Translation* **23**(2), 117–127.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D. & Ng, A. Y. (2011), Dynamic pooling and unfolding recursive autoencoders for paraphrase detection, *in* 'Advances in Neural Information Processing Systems', pp. 801–809.

Sorg, P. & Cimiano, P. (2008), Cross-lingual information retrieval with explicit semantic analysis, *in* 'Working Notes for the CLEF 2008 Workshop'.

Voorhees, E. M. (2001), 'The trec question answering track', *Natural Language Engineering* **7**(4), 361–378.

Yin, W. & Schütze, H. (2015), Convolutional neural network for paraphrase identification, *in* 'Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies', pp. 901–911.

Zhang, X., Rong, W., Liu, J., Tian, C. & Xiong, Z. (2017), Convolution neural network based syntactic and semantic aware paraphrase identification, *in* 'Neural Networks (IJCNN), 2017 International Joint Conference on', IEEE, pp. 2158–2163.