A Deep Network Model for Paraphrase Detection in Short Text Messages

Basant Agarwal[a,b,*], Heri Ramampiaro[a], Helge Langseth[a], Massimiliano Ruocco[a,c]

[a]*Dept. of Computer Science, Norwegian University of Science and Technology, Norway*
[b]*Dept. of Computer Science and Engineering, Swami Keshvanand Institute of Technology, India*
[c]*Telenor Research, Trondheim, Norway*

Abstract

This paper is concerned with paraphrase detection, i.e., identifying sentences that are semantically identical. The ability to detect similar sentences written in natural language is crucial for several applications, such as text mining, text summarization, plagiarism detection, authorship authentication and question answering. Recognizing this importance, we study in particular how to address the challenges with detecting paraphrases in user generated short texts, such as Twitter, which often contain language irregularity and noise, and do not necessarily contain as much semantic information as longer clean texts. We propose a novel deep neural network-based approach that relies on coarse-grained sentence modelling using a convolutional neural network (CNN) and a recurrent neural network (RNN) model, combined with a specific fine-grained word-level similarity matching model. More specifically, we develop a new architecture, called DeepParaphrase, which enables to create an informative semantic representation of each sentence by (1) using CNN to extract the local region information in form of important n-grams from the sentence, and (2) applying RNN to capture the long-term dependency information. In addition, we perform a comparative study on existing (state-of-the-art) approaches within paraphrase detection. An important insight from this study is that existing paraphrase approaches perform well when applied on clean texts, but they do not necessarily deliver good performance against noisy texts, and vice versa. In contrast, our evaluation has shown that the proposed DeepParaphrase-based approach achieves good results in both types of text, thus making it more robust and generic than the existing approaches.

Keywords: Paraphrase detection, Sentence Similarity, Deep learning, RNN, CNN.

1. Introduction

Twitter has for some time been a popular means for expressing opinions about a variety of subjects. Paraphrase detection in user-generated noisy texts, such as Twitter texts[1], is an important task for various Natural Language Processing (NLP), information retrieval and text mining tasks, including query ranking, plagiarism detection, question answering, and document summarization. Recently, the paraphrase detection task has gained significant interest in applied NLP because of the need to deal with the pervasive problem of linguistic variation.

Paraphrase detection is an NLP classification problem. Given a pair of sentences, the system determines the semantic similarity between the two sentences. If the two sentences convey the same meaning, then it is labelled as *paraphrase*; otherwise, it is labeled as *non-paraphrase*. Most of the existing paraphrase systems have performed quite well on clean text corpora, such as the Microsoft Paraphrase Corpus (MSRP) [1]. However, detecting paraphrases in user-generated noisy tweets is more challenging due to issues like misspelling, acronyms, style and structure [2]. In addition, measuring the semantic similarity between two

---

short sentences is very difficult due to the lack of common lexical features [3]. Although little attention has been given to paraphrase detection in noisy short-texts thus far, some initial work has been reported on the *SemEval 2015* benchmark Twitter dataset [2, 4, 5]. Unfortunately, the best performing approaches on one dataset perform poorly when evaluated against another. As we discuss later in this paper, the state-of-the-art approach for the SemEval dataset proposed by Dey et al. [5] does not have good performance (in form of F1-score) when evaluated on the MSRP dataset. Similarly, Ji and Eisenstein [6] is the best performing approach on the MSRP dataset, but does not perform well on the SemEval dataset. In conclusion, existing approaches are not very generic; but instead, they are highly dependant on the data used for training.

Focusing on the problem discussed above, the main goal of this work is to develop a robust paraphrase detection model based on deep learning techniques that is able to successfully detect paraphrasing in both noisy and clean texts. More specifically, we propose a hybrid deep neural architecture composed by a convolutional neural network (CNN) and a recurrent neural network (RNN) model, further enhanced by a novel word-pair similarity module. The proposed paraphrase detection model is composed of two main components: (1) sentence modeling and (2) pair-wise word similarity matching. First, *sentence modeling* concerns building an effective model to represent the text. To do this, we build joint CNN and RNN architecture that takes the local features extracted by CNN as input to the RNN. We take word embeddings as input to the CNN model. Then, after convolutions and pooling operations, the encoded feature maps are taken in sequence as input to the RNN model. The last hidden state learned by the RNN model is considered as the sentence level representation. The main rationale behind using both CNN and RNN here is that CNN is able to learn the local features in form of important n-grams of the texts; whereas RNN takes words in a sequential order and is able to learn the long-term dependencies of texts rather than local features. Second, the *pair-wise similarity matching model* is used to extract fine-grained similarity information between pairs of sentences. Initially, a pair-wise similarity matrix is constructed by computing the similarity of each word in a given sentence to all the words in another sentence. We then apply CNN onto this similarity matrix to analyze the patterns in the semantic correspondence between each pair of words in the two sentences that are intuitively useful for paraphrase identification. The idea to apply convolutions over similarity matrix to extract the important word-word similarity pairs is motivated by how convolutions over text can extract the most important parts of a sentence.

In this paper, we show how the proposed model for paraphrase detection can be enhanced by employing an extra set of statistical features extracted from the input text. To demonstrate its robustness, we evaluate the proposed approach and compare it with the state-of-the-art models, using two different datasets, covering both noisy user-generated texts – i.e., the SemEval 2015 benchmark Twitter dataset, and clean texts – i.e., the Microsoft Paraphrase Corpus (MSRP).

The main contributions of this work can be summarized as follows:

1. We propose a novel deep neural network architecture leveraging coarse-grained sentence-level features and fine-grained word-level features for detecting paraphrases on noisy short text from Twitter. The model combines sentence-level and word-level semantic similarity information such that it can capture semantic information at each level. When the text is grammatically irregular or very short, the word-level similarity model can provide useful information; while the semantic representation of the sentence provide useful information otherwise. In this way both model-components compliment each other and provide an efficient overall performance.

2. We show how the proposed pair-wise similarity model can used to extract word-level semantic information, and demonstrate its usefulness in the paraphrase detection task.

3. We propose a method combining statistical textual features and features learned from the deep architecture.

4. We present an extensive comparative study for the paraphrase detection problem.

The rest of the paper is organized as follows: In Section 2, we formally define the problem. In Section 3, we discuss related work concerning paraphrase detection. In Section 4, we motivate our work and present our proposed solution in detail. In Section 5, we describe the experimental setup. In Section 6, we evaluate the approach and discuss the results. Finally, in Section 7, we conclude the paper and outline plans for future research.

2. Problem statement and goals

Let S_1 and S_2 be two sentences, such that $S_1 \neq S_2$. S_1 and S_2 are said to be paraphrased if they convey the same meaning and are semantically equivalent. Now, assume that we have a collection of N annotated sentence pairs (S_1^i, S_2^i), having annotations k_i, for $i = 1, 2, \ldots N$. For a given i, k_i indicates whether the i-th sentence pair is *paraphrased* or *non-paraphrased*. The problem addressed in this paper is to develop a model, which can reliably annotate a previously unseen sentence pair as paraphrased or non-paraphrased. In particular, we aim at answering the following main research question:

> *How to develop a robust and generic method for paraphrase detection?*

To address the above question, this work has the following main goals:

- **Study of existing approaches**: To get full insight about the current status, one of the main goals of this work is to perform a thorough analysis of existing approaches. This includes doing an extensive comparative study about the performance of these approaches, with respect to both robustness and generality.

- **Development of robust and generic neural network-based architecture**: Based on the insight from studying previous work, we aim at developing a neural network-based architecture that can advance the state-of-the-art in terms of both robustness and generality, as well as performance/effectiveness.

Note that as discussed later in this paper, there are several methods that have been proposed, and work well for clean texts, but most of them have failed to provide good results when applied on noisy texts like Tweets. Recently, other approaches have been developed for paraphrase detection on noisy texts, e.g., the work by Xu et al. [2] and Dey et al. [5], but these approaches do not work well on clean texts. In conclusion, there is still a strong need for a robust and reliable method, which can perform well for both clean texts and user-generated noisy short texts.

3. Related work

The use of deep neural network for natural language processing (NLP) has increased considerably over the recent years. Most of the previous work on paraphrase detection have focused on features like n-gram overlap features [7], syntax features [8], linguistic features [9, 10] wikipedia-based semantic networks [11], knowledge graph [12] and machine translation based features [7]. Recently, deep learning-based methods have shifted researchers' attention towards semantically distributed representations [13, 14, 15]. It has been shown that CNN has been used for learning representation of the text that improved the results in sentence classification task [16]. Recurrent neural networks (RNNs) is able to learn the long-term dependencies in sequential data and that has also been used in the literature for representing the text [17]. Kim et al. [18] proposed a model that uses CNN, a highway network over characters, whose output is send to the RNN network. Their combined CNN-RNN model provides better results. Wang et al. [19] proposed combination of convolution and recurrent neural network to learn the sentence representation for sentiment analysis task. A variety of deep neural network-based architectures have been proposed for sentence similarity, which is a strategy we also focus on in this paper.

3.1. Work on clean text

Substantial work has been carried out on paraphrase detection for the clean-text Microsoft Paraphrase corpus. Das and Smith [8] present a probabilistic model for paraphrase detection based on syntactic similarity, semantics, and hidden loose alignment between syntactic trees of the two given sentences. Heilman and Smith [20] propose a tree edit model for paraphrase identification based on syntactic relations among words. They develop a logistic regression model that uses 33 syntactic features of edit sequences to classify a sentence pair. Socher et al. [21] present an approach based on recursive autoencoders for paraphrase detection. Their approach learns feature vectors for phrases in syntactic trees and employs a dynamic pooling layer mechanism, which converts a variable sized matrix into a fixed-sized representation. Parsing is a

powerful tool for identifying the important syntactic structure in the text, but relying on the parsing makes the approach less flexible. Our approach does not use such resources to develop the model. Oliva et al. [22] propose SyMSS based on the syntactic structure of the sentences. They represent the sentences as a syntactic dependence tree, use WordNet to extract meaning of individual words, and further use syntactic connections among them to assess information similarity. Ji and Eisenstein [6] use several hand-crafted features with latent representation from matrix factorization as features to train a support vector machine. The ARC model proposed by Hu et al. [23] is a convolutional Siamese architecture, in which two shared-weight convolutional sentence models are trained. El-Alfy et al. [24] propose a model considering a set of weak textual similarity metrics. They boost the performance of individual metrics using abductive learning. Further, they aim to select an optimal subset of similarity measures and construct a composite score that is used for classification. Filice and Moschitti [25] proposed an approach for paraphrase identification based on whether a text fragment only appearing in one of the sentence is important or ancillary. They represent text with syntactic structures and then apply tree kernels with SVM on them to determine the paraphrases. AL-Smadi et al. [26] proposed a paraphrase detection approach for arabic tweets, which employs a set of extracted features based on lexical, syntactic, word alignment, and semantic computation. Wang et al. [27] decompose the sentence similarity matrix into a similar component matrix and a dissimilar component matrix, and train a two-channel convolutional neural network to compose these components into feature vectors. Lopez-Gazpio et al. [28] proposed an interpretable semantic textual similarity approach based on alignment between pairs of segments across two sentences. Yin et al. [29] proposed an attention-based convolutional neural network for paraphrase identification task that integrates the mutual influence between sentences into CNNs. Wang et al. [30] proposed deep learning based model in which they first match the BiLSTM encoded two sentences in two direction. In each matching direction, each timestep of one sentence is matched with other sentence from multiple perspectives. Finally, another BiLSTM layer aggregate the matching results which is used for final decision. Ferreira et al. [31] propose a supervised machine learning learning approach. They extract various features based on lexical, syntactic and semantic similarity measures, and use various machine learning algorithms such as Bayesian Network, RBF Network, C4.5 decision tree, and support vector machines. Severyn and Moschitti [32] presented CNN based neural network architecture to rank short text pairs for question answering task. They used CNN to learn the representation of the query and document, and construct similarity matrix using intermediate sentence representations. Further, they compute a single similarity score using the query, documents and similarity matrix. Finally, they use semantic representation of both the sentences, similarity score and additional features to the softmax classifier. As compared to the proposed approach, they used a similarity score as one feature which is extracted using sentence representations and similarity matrix, whereas, in the proposed approach, we extract important similarity scores by applying the CNN over similarity matrix followed by max-pooling operation. Kenter and de Rijke [33] extract various word-level to text-level semantic features from word alignment and saliency weighted semantic graph, which they further use to build support vector classifier to predict the semantic similarity score for a given pair of sentences. As compared to our approach, they used the similarities among words in a different way. They transformed the similarity values into bins based on the maximum similarity for the words, and further used those bins as features along with different word embeddings to create the final feature vector to train a supervised learning algorithm

3.2. Work on short text

A number of contributions have also been reported for detecting paraphrases on noisy short-text like Tweets. Xu et al. [2] propose a latent variable model that jointly infer the correspondence between words and sentences. Eyecioglu and Keller [34] use a support vector machine with simple lexical word overlap and character n-grams features for paraphrase detection. Zhao and Lan [35] use various machine learning classifiers, and employ a variety of features like string-based, corpus-based, syntactic features, and word distributional representations. Zarrella et al. [36] present an ensemble approach based on various features such as mixtures of string matching metrics, distance measurements, tweet-specific distributed word representations, and recurrent neural networks for modeling similarity. Karan et al. [37] present a supervised approach that combines semantic overlap and word alignment features. Wieting and Gimpela [38] propose an approach named as Gated Recurrent Averaging Network for sentence similarity task on noisy text pairs that is based

on the combination of averaging word embeddings and LSTM neural network. Vo et al. [39] experiment with various sets of features with different classifiers and show that the combination of word/n-gram, word alignment by METEOR (Metric for Evaluation of Translation with Explicit ORdering), BLEU (Bilingual Evaluation Understudy) and EditDistance is the best feature set for Twitter paraphrase detection; and that VotedPerceptron proved to be the best machine learning algorithm. Huang et al. [40] proposed a deep learning approach for paraphrase detection for short text using Twitter dataset, their approach extracts features at multiple granularity levels from character-level to sentence-level. They extract the subword information for fine-grained features with character-level CNN, the n-gram features with word-level CNN and sentence representation using LSTM. Dey et al. [5] use a set of lexical, syntactic, semantic and pragmatic features for detecting paraphrases on noisy user-generated short-text data such as Twitter.

3.3. Summary

As the above discussion illustrates, there are many methods that have been proposed for paraphrase detection, but existing methods have issues that have to be addressed. In this paper, we focus on using deep learning algorithms to develop a robust and reliable paraphrase detection system, which can work well on both clean-text and noisy short text such as tweets. To the best of our knowledge, this is the first work to fully explore this area, while also including a comprehensive comparative study of exiting approaches.

Table 1 summarizes the approaches discussed in section.

Table 1: Comparison among related approaches.

Work	Description	Resources used	Classification	dataset
[34]	ASOBEK: Word overlap and character n-grams features	POS tagger	Support vector machine (SVM)	Twitter, MSRP
[36]	MITRE: mixtures of string matching metrics	– –	L1-regularized logistic regression	Twitter dataset
[35]	ECNU: Various string based, corpus based, syntactic, and distributed word representation based features	POS tagger, WordNet, various pre-trained word embeddings	SVM, Random Forest (RF), Gradient Boosting (GB)	Twitter dataset
[39]	Various features such as Machine translation, EDIT distance, sentiment features	POS Tagger	Decision Stump, OneR, J48, Baysian Logistic Regression, VotedPerceptron, MLP	Twitter dataset
[40]	Combination of Char-CNN, word-CNN, and LSTM	Pre-trained word embeddings	multi-layer perceptron	Twitter dataset
[37]	Semantic Overlap Features and Word Alignment Features	POS tagger	SVM	Twitter dataset
[2]	Multi-instance Learning Paraphrase Model (MULTIP)	POS tagger	Similarity score	Twitter dataset
[5]	A set of lexical, syntactic, semantic and pragmatic features	WordNet, POS Tagger, NE Tags	SVM	Twitter, MSRP
[41]	Combination of several word similarity measures	POS tagger	Similarity score threshold	MSRP
[42]	Weighted Textual Matrix Factorization (WTMF) with handling missing words	WordNet	Matrix factorization	MSRP

Continue on the next page.

Table 1: Comparison among related works (cont.).

Work	Description	Resources used	Classification	dataset
[25]	Tree kernels applied to syntactic representations of text fragments	POS Tagger, Parser	SVM	MSRP
[33]	Word alignment method and a saliency weighted semantic graph	Pre-trained word embeddings	Support vector classifier with radial basis function kernel	MSRP
[8]	Probabilistic model with syntactic and n-gram overlap features	WordNet, Dependency parser	Logistic regression, SVM	MSRP
[20]	Syntactic features of edit sequences	POS Tagger, Parser, WordNet	Logistic regression	MSRP
[22]	Similarity features based in syntactic dependency tree	WordNet, dependency parser	Similarity score threshold	MSRP
[21]	Representation of feature vectors for phrases in syntactic trees	Dependency Parser	Recursive autoencoder with dynamic pooling	MSRP
[29]	Attention Based Convolutional Neural Network (ABCNN)	Word2vec based embeddings	logistic regression	MSRP
[6]	Matrix factorization with supervised reweighting	– –	SVM with a linear kernel	MSRP
[23]	Hierarchical structures of sentences with their layer-by-layer composition	Pre-trained word embeddings	Convolutional Neural Network	MSRP
[7]	Combination of eight machine translation metrics	WordNet	SVM	MSRP
[24]	Boosting through textual similarity metrics	– –	SVM	MSRP
[27]	Sentence Similarity Learning by Lexical Decomposition and Composition	Pre-trained word Embeddings	CNN	MSRP
[31]	Represent pair of sentence as combination of similarity measures	Dependency Parser	SVM, RBF Network, Bayesian Network	MSRP
This work	Hybrid of deep learning and statistical features	POS Tagger & pre-trained word Embeddings	Multi-layer neural network	MSRP, Twitter dataset

4. DeepParaphrase Architecture

We propose a deep learning-based approach for detecting paraphrase sentences for tweets. We first convert each sentence in a pair into a semantic representative vector, using a CNN and an RNN. Then, a semantic pair-level vector is computed by taking the element-wise difference of each vector in the sentence representations. The resulting difference is the discriminating representative vector of the pair of sentences, which is used as feature vector for learning the similarity between the two sentences. In addition to this coarse-level semantic information, we extract more fine-grained important information using a similarity

matrix which contains word-to-word similarity quantification. Further convolutions are applied over the pair-wise similarity matrix to learn the similarity patterns between the words in the pair of sentences. The aim of the convolution function is to extract more fine-grained similarity features. Finally a third set of features are extracted using statistical analysis of the text, and concatenated with the rest of the learned features. A fully connected neural network is used to produce the classification from this concatenated feature vector. The first layers are activated by the *ReLU* [43] function, while we use the sigmoid link-function to transfer the latent representation into a two-class decision rule. We train the model to optimize binary cross-entropy. The proposed architecture is depicted in Figure 1.
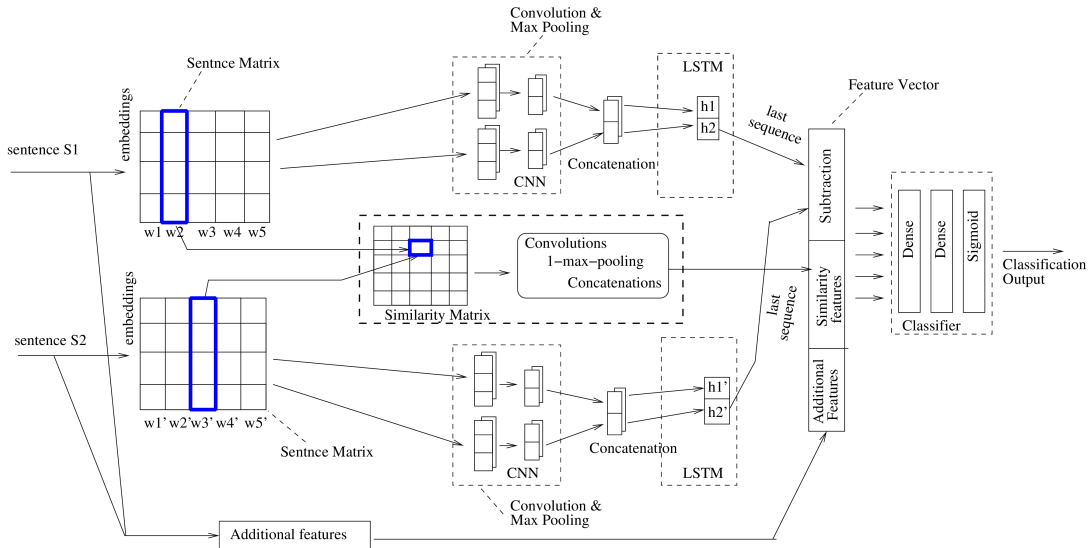

Figure 1: The proposed DeepParaphrase architecture

At a high level of abstraction the proposed model therefore consists of two main components, that will be discussed next.

4.1. Sentence modelling with CNN and RNN

In this component, we represent every sentence using joint CNN and RNN architecture. The CNN is able to learn the local features from words to phrases from the text, while the RNN learns the long-term dependencies of the text [19]. More specifically, we firstly take the word embedding as input to the CNN model. Then, we apply convolutions of different filter lengths and widths; next max-pooling operation is applied to capture the maximum information from the text. The obtained encoded feature maps in sequence are taken as input the RNN network. Finally, the long term dependencies learned by the RNN becomes the semantic sentence representation.

The architecture of the proposed model for mapping the sentences into a feature vector is shown in Figure 1. The main goal of this step is to learn good intermediate semantic representations of the sentences, which are further used for the semantic similarity task. The input to the sentence model is a pair of sentences S_1 and S_2, which we transform into matrices of their words' embeddings. Here each word is represented by a vector $w \in \mathbb{R}^d$, where d is the size of the word embedding. We used pre-trained word embeddings (see Section 6.3 for details). The sentence embedding matrices are then fed into the CNN. We apply convolution operation with different filters widths and filters. To obtain sufficient important features (feature maps), we experiment with different number of filters and filter length. We empirically decided filter width to be (3, 4) and 200 filters in our experiments. Next, we apply max pooling operation onto the output of convolutions to keep only important features. Next, we apply concatenation operator $\oplus$ on the output of max-pooled feature map matrices generated using filter width of 3 and 4. Next, the output is used as input to the RNN network. The features learned from CNN can be considered as n-grams, and are fed in sequence into RNN.

The last hidden state of the RNN layer is considered as the best semantic encoding of the sentence that includes information of the local region as well as long-term distance in the sentence. A RNN composed of Long Short-Term Memory (LSTM) units is often called an LSTM network[44]. The LSTM is a variation of RNN, which processes sequential data and can learn long-term dependencies due to its ability to deal with the exploding and vanishing gradient problem. We use LSTM network in our experiments.

The aim of the convolutional layer is therefore to extract patterns, i.e., important word sequences from the input sentences. The motivation for using convolutions comes from the fact that convolutional filters can learn n-gram discriminating features, which is useful for sentence similarity analysis. The pooling operation in CNN allow to retain only important n-grams as it would omit unimportant features. The important features generated by the convolutional layer are fed in sequence into the RNN. This model component is able to process sequential input with the aim to learn the long-term dependencies and positional relations of the features in the sentences. The CNN-RNN combined model is able to retain local features and their relation along with long-term dependency based global features of the sentence. Eventually, the *last* hidden state of the RNN is taken as the semantic representation of the sentence, and the element-wise difference between these representations is used as a semantic discrepancy measure at the level of the sentence pair.

4.2. Pair-wise word similarity matching

A pair-wise similarity matrix is construed by computing the similarity of each word in S_1 to another word in S_2. Convolutions are applied onto this similarity matrix to analyze patterns in the pair-wise word to word similarities. Figure 2 illustrates this process.

It is intuitive that given two sentences, semantic correspondence between words provide important semantic information for detecting similar sentences, and the pair-wise word similarity matching model learns the word-level similarity patterns between the two sentences. Because important n-grams are extracted by applying convolutional neural network over text, we obtain the important word-word similarity pairs from the similarity matrix. This similarity matrix is further used as features for the classification of the paraphrase detection problem. The goal of the pair-wise word similarity matching model is to compare the semantic embedding of each word in one sentence against all the semantic embeddings of the words from the other sentences. This means that we compute the dot product as a similarity measure between all the word embeddings of the two sentences. Finally, we match the two sentences and generate a similarity matrix $\mathbb{S}$ of size $m \times n$, where m and n denote the lengths of sentence S_1 and S_2, respectively. Next, we apply the CNN onto the similarity matrix to learn the patterns in the semantic correspondence between the two sentences. We convolve over $\mathbb{S}$ in two directions; both from left to right and from top to bottom. This gives two separate results, F_1 and F_2. After the convolution layer, global max-pooling is applied to obtain the most informative feature vectors from F_1 and F_2, and finally these are concatenated to produce the output from this module. We experiment with different number of filters and filter width to extract the maximum information, and we empirically decided to use 200 filters and 3 filter width.
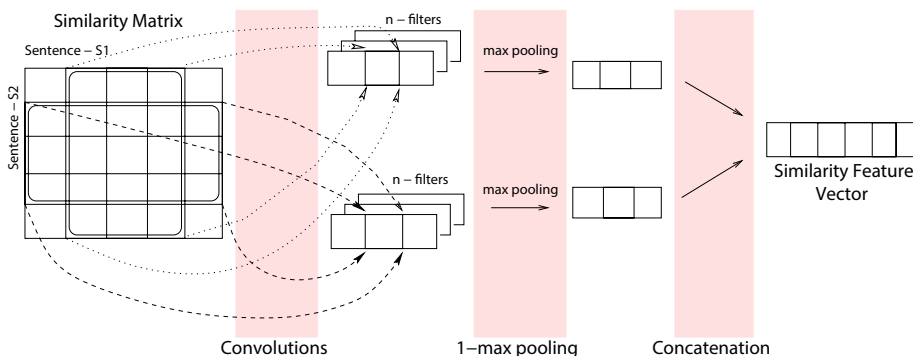

Figure 2: Pair-wise word similarity matching model

4.3. Statistical features

We extracted a third set of features to enhance the discriminating representation of the sentences. These features consist of the following:

1. *TF-IDF*-based similarity between sentences S_1 and S_2.
2. Cosine similarity between the vectors of sentences S_1 and S_2.
3. The average Wordnet-based similarity between the verbs[2] in sentence S_1 and those in S_2.
4. The average Wordnet-based similarity between the nouns[2] in sentence S_1 and S_2.
5. The average Wordnet-based similarity between the adjectives[2] in sentence S_1 and S_2.
6. The cosine similarity between the semantic representation of each sentence pair.
7. Six n-gram overlap features computed by the number of unigrams, bigrams, and trigrams that are common to the given sentence pair, divided by the total n-grams in S_1 and S_2 respectively.

We use all these additional features for the experiments performed on the Microsoft Paraphrase corpus, while only the two latter features were used for the experiments on Twitter corpus.

5. Experimental Setup

Before evaluating our proposed method for paraphrase identification, and compare it against the state-of-the-art approaches, we first describe how our experiments have been set up, including the datasets, and performance measures that we have used. We reported the baseline results from the previous work, and did not re-run them again. By doing so, we assumed that by using exactly the same dataset, we would get the same results. Also, in order to get fair comparisons, we made sure that the experimental settings are same in all these baseline methods which are compared.

5.1. Datasets

We consider two widely-used benchmark datasets, which we briefly describe in the following:

1. **Twitter Paraphrase SemEval 2015 dataset:** The dataset provided by SemEval 2015 [4] has been used by all the recent works for paraphrase detection in Tweets. It consists of noisy and short text, containing 3996 paraphrase and 7534 non-paraphrase pairs in the training dataset, 1470 paraphrase and 2672 non-paraphrase sentence pairs in development set, and 838 tweets in the test set. We have ignored the "debatable" entries, that were marked in [4]. All the existing models that use this dataset also ignored these debatable samples while reporting the results.
   The statistics of the dataset are shown in Table 2.

Table 2: Statistics of the SemEval-2015 Twitter paraphrase corpus

	Unique sent.	Sent. pair	Paraphrase	Non-paraphrase	Debatable
Train	13231	13063	3996	7534	1533
Dev	4772	4727	1470	2672	585
Test	1295	972	175	663	134

2. **Microsoft Paraphrase dataset:** We also investigate the empirical performance of the proposed model on a clean text corpus. More specifically, we use the Microsoft Paraphrase dataset [1], which is considered the evaluation standard for paraphrase detection algorithms. This dataset comprises candidate paraphrase sentence pairs, obtained from Web news sources. In this corpus, the length of each sentence varied from 7 to 35 words with an average 21 words in a sentence. Furthermore, the data is split into training and test sets, containing 4076 and 1725 samples respectively. This same train/test partitioning has been applied on all the approaches evaluated in this paper.

---

[2] In our implementation we use `nltk`'s part of speech tagger to extracts the verbs, nouns and adjectives from each sentence.

Table 3: Statistics of the MSRP corpus

	Sent. pair	Paraphrase	Non-paraphrase
Train	4076	2753	1323
Test	1725	1147	578

Despite being the most widely-used datasets for evaluating paraphrase detection models, their sizes are too small to reliably train a deep learning architecture. We have therefore applied a simple augmentation scheme to double the number of sentence pairs in the corpus: For every pair of sentences (S_1, S_2) we simply exchange the order of sentences to obtain the new pair (S_2, S_1), and add this new pair to the corpus [45].

5.2. Performance measures

We adopted the standard performance measure that are widely used in the literature for paraphrase detection. These measures are *precision*, *recall*, *F1-score* and *accuracy*. Precision is defined as number of correctly classified paraphrase pairs out of total paraphrase sentence pairs extracted. Recall is the ratio between predicted sentence pairs that are actual paraphrases, and total true paraphrase pairs. The F1-score combines the precision and recall. Finally, accuracy is the fraction of the paraphrase sentence pairs that are classified correctly.

6. Results and Discussion

In this section we present the results from using both datasets that we presented in Section 5.1.

6.1. Results and Discussion on Twitter Corpus

We train our model using the training dataset with development set for tuning the parameters, and then we test the system with the provided testing dataset of 838 test entries, ignoring the "debatable" entries. These results are provided in Table 4.

Recall that there are mainly two components in the proposed approach: (*i*) Sentence modelling using *CNN* and *LSTM*, and (*ii*) Pair-wise word similarity matching. Our intuition for using the two models is that both coarse-grained sentence-level and fine-grained word-level information should be important for the paraphrase detection task. In our experiments, we firstly use only sentence modelling architecture to develop the paraphrase detection model. We call this experiment the *SentMod Architecture* for paraphrase detection. It can be seen from the results in Table 4 that the *SentMod* Architecture performs quite well, giving an *F1-score* of 0.692. Next, we use the pair-wise word similarity matching model to extract the word-level similarity information-based features. When we use only these features to train the paraphrase model, the model provides an F1-score of 0.702. We call these features the *pair-wise* features.

Further, we augment these word-level pair-wise features with the sentence-level features extracted using the *SentMod Architecture*, and feed it to train the proposed deep learning model for paraphrase detection task. We call the architecture for this model *DeepParaphrase Architecture*. The experimental results show the significant improvement in the performance of the paraphrase detection task. Specifically, it gives an *F1-score* of 0.742 (an improvement of 7.2 percentage points). It also shows that the pair-wise word similarity information in fusion with sentence-level similarity information provides good performance for paraphrase detection task. Finally, we add two features, namely the overlap features and similarity features (items 6 and 7 in the description in Section 4.3). This gives an additional improvement in the performance of the model, resulting in an F1-score of 0.751, which is significantly better than other existing methods for paraphrase detection on the Twitter dataset. We refer to this final model as the *AugDeepParaphrase* model.

The comparison between the proposed method and existing state-of-the-art methods is provided in Table 4. Firstly, we compare the results of the proposed approach with the methods which perform best on clean text Microsoft Paraphrase dataset, and then with the state-of-the-art methods on noisy Twitter

Table 4: Results of the proposed approach along with comparison with state-of-the-art results on SemEval 2015 Twitter dataset.

Model	Precision	Recall	F1-score
SentMod Architecture	0.725	0.663	0.692
Pair-wise Features	0.708	0.697	0.702
DeepParaphrase Architecture	0.753	0.731	0.742
AugDeepParaphrase	0.760	0.742	**0.751**
Random	0.208	0.500	0.294
Guo and Diab [42]	0.583	0.525	0.655
Das and Smith [8]	0.629	0.632	0.630
Ji and Eisenstein [6]	0.664	0.628	0.645
Eyecioglu and Keller [34]	0.680	0.669	0.674
Zarrella et al. [36]	0.569	0.806	0.667
Zhao and Lan [35]	0.767	0.583	0.662
Vo et al. [39]	0.685	0.634	0.659
Karan et al. [37]	0.645	0.674	0.659
Huang et al. [40]	0.643	0.657	0.650
Xu et al. [2]	0.722	0.726	0.724
Dey et al. [5]	0.756	0.726	0.741

dataset. Guo and Diab [42] proposed a weighted textual matrix factorization method for paraphrase detection based on modeling the semantic space of the words that are present or absent in the sentences. Their model uses WordNet, OntoNotes, Wiktionary, and the Brown corpus.Their approach performed quite well on MSRP dataset, but provide worse results on Twitter dataset. Das and Smith [8] used logistic regression based classifier based on simple n-gram features and overlapping features which shows competitive results on MSRP dataset. Ji and Eisenstein [6] presented a state-of-the-art model for paraphrase detection on MSRP dataset which is still the best known performance on clean text. However, it can be seen from the results presented in Table 4 that their method performed worse than other methods on the Twitter data. We further compare the proposed approach with existing state-of-the-art methods on twitter dataset. The ensemble-based method proposed by Zarrella et al. [36] obtained higher recall as compared to our results, but our model gave higher overall F1-score. While the method suggested by Zhao and Lan [35] got slightly higher precision as compared to proposed approach, our approach is superior with respect to the F1-score. As can be observed, the results from this comparison, our approach outperforms all related methods with respect to the F1-score. The main reason for this is that our approach leverages the semantic information at both coarse-grained sentence-level features and fine-grained word-level features for detecting paraphrases on tweets.

In conclusion, the state-of-the-art algorithms, that perform well when trained on clean texts, do not necessarily work very well for noisy short texts, and vice versa. In contrast to this, our approach is robust in the sense that it performs well on both types of datasets. More specifically, it outperformed all the existing methods when applied on noisy texts, and produced very competitive results against the state-of-the-art methods on clean texts.

Next, we analyze the misclassifications on test data using the proposed approach. Some example tweets pairs including both correct and incorrect detection by our model are reported in Table 5. We show some examples from the test data which cases our method could correctly classify. For example, our proposed approach could correctly identify the tweet pair, *"Terrible things happening in Turkey"* and *"Children are dying in Turkey"* as "paraphrase". It could understand the semantic meaning, despite the fact that the pair only has one common word. Similarly, the proposed approach could determine correct label as "non-paraphrase" for the sentence pairs on row 2 and 3 in Table 5, although the sentence-pairs have several words

Table 5: Examples of tweet pairs from the Twitter Paraphrase Corpus.

S. No.	Tweet 1	Tweet 2	Gold annotation	Prediction	Remark
1	Terrible things happening in Turkey	Children are dying in Turkey	paraphrase	paraphrase	correct
2	Anyone trying to see After Earth sometime soon	Me and my son went to see After Earth last night	non-paraphrase	non-paraphrase	correct
3	hahaha that sounds like me	That sounds totally reasonable to me	non-paraphrase	non-paraphrase	correct
4	I dont understand the hatred for Rafa Benitez	Top 4 and a trophy and still they dont give any respect for Benitez	paraphrase	non-paraphrase	incorrect
5	Shonda is a freaking genius	Dang Shonda knows she can write	paraphrase	non-paraphrase	incorrect
6	Terrible things happening in Turkey	Be with us to stop the violence in Turkey	paraphrase	non-paraphrase	incorrect
7	I must confess I love Star Wars	Somebody watch Star Wars with me please	paraphrase	non-paraphrase	incorrect
8	Family guy is really a reality show	Family guy is such a funny show	non-paraphrase	paraphrase	incorrect
9	I see everybody watching family guy tonight	I havent watched Family Guy in forever	non-paraphrase	paraphrase	incorrect

in common.

Nevertheless, there are several examples where it has been difficult to provide correct classifications. Consider, for example, the tweet pair no. 4 in Table 5. Our approach determines this pair as "non-paraphrase", which is incorrect according to the gold-standard annotation. The two tweets do not share many words, and common-sense knowledge is required to understand that a person who has won lots of trophies and prizes should be respected rather than hated. Another similar example is the tweet pair no. 5 The gold-standard annotation for this pair is that it is a paraphrase. To correctly classify this pair, the system needs to know that if a person is genius, then it is obvious that he/she would be able to write well.

Finally, consider the pair *"Family guy is really a reality show"* and *"Family guy is such a funny show"*. Our approach identifies this pair as "paraphrase", which is wrong according to the gold-standard annotation. The possible reason for this error is the misleading lexical overlap information between the sentences in the pair, that are overshadowed by the few different words.

To summarize, after looking at the misclassified examples in Table 5, there are several cases that could cause our system to fail to correctly classify pairs of tweets. This includes cases where common-sense knowledge is required. What could be learned from the examples is, however, that our proposed approach is able to capture the semantic information from short, noisy texts, which can, in turn, help in correctly classifying pairs that would otherwise be difficult by only looking at the syntactic contents.

6.2. Results and discussions on MSRP dataset

The results of our experiments with the Microsoft Paraphrase dataset are summarized in Table 6. Firstly, we extract the coarse-grained sentence-level features with the *SentMod Architecture* and further feed to train

paraphrase detection model. As can be observed in Table 6, this architecture gives an accuracy of 74.5% and F1-score of 81.5%. Next, we evaluate the *pair-wise* features to train the paraphrase detection model, these features individually provide 81.9% F1-score. Further, we fuse these pair-wise features with the sentence-level features extracted using the SentMod Architecture to train the paraphrase detection model. This *DeepParaphrase Architecture* provides a significant improvement in the performance. With this deep learning model we obtain the accuracy of 77.0% and F1-score of 84.0%. The final paraphrase model, *AugDeepParaphrase* model, is built by including the additional features described in Section 4.3. Here, we see an improvement of 0.5% in F1-score. Overall, the experimental results show that both sentence-level semantic information and word-level similarity information are important for paraphrase detection task.

Table 6: Results of the proposed approach along with comparison with state-of-the-art methods on MSRP dataset.

Model	Accuracy	F1
SentMod Architecture	74.5	81.5
Pair-wise Features	74.8	81.9
DeepParaphrase Architecture	77.0	84.0
AugDeepParaphrase	**77.7**	**84.5**
All positive (Baseline)	66.5	79.9
Socher et al. [21]	76.8	83.6
Ji and Eisenstein [6] (Inductive setup)	77.8	84.3
Hu et al. [23] ARC- I	69.6	80.3
Hu et al. [23] ARC-II	69.9	80.9
Madnani et al. [7]	77.4	84.1
Eyecioglu and Keller [34]	74.4	82.8
Kenter and de Rijke [33]	76.6	83.9
El-Alfy et al. [24]	73.9	81.2
Wang et al. [27]	**78.4**	**84.7**
Dey et al. [5]	-	82.5
Pagliardini et al. [15]	76.4	83.4
Ferreira et al. [31]	74.08	83.1

As with the Twitter dataset, we also compared our approach with several related methods. We present the results of the experiments in Table 6, in which we report the measured accuracy and the F1-scores. The experimental results show that the proposed approach outperforms all the related methods, except quite recent method by Wang et al. [27]. As discussed in Section 3, they also employ a neural network-based approach. Nevertheless, the large number of options introduced in the final model, such as: the semantic matching functions {*max, global, local-l*}, decomposition operations {*rigid, linear, orthogonal*} and filter types {*unigrams, bigrams, trigrams*}, makes it less applicable to re-implement or scale for other datasets or other similar problems. In contrast, we have developed our approach to be more robust and generic, such that it can easily be applied for other datasets.

In [6], the authors reported the best results as 80.4% accuracy and 85.9% F1-score on this dataset. However, to achieve these results, they seemed to have relied on using testing data with training dataset to build the model. They called it a form of transductive learning, in which they assumed that they have access to a test set. In contrast, in our approach, the test data is kept totally disjoint from the training process. Using the same experimental setup – i.e., applying "Inductive" setup without using test data in training the model – the approach by Ji and Eisenstein [6] gives an accuracy of 77.8% and F1-score of 84.3%, which is very close to the results of our approach.

Focusing on the performance of our approach in relation to the existing methods, our experimental results show that our approach produces competitive results, achieving accuracy of 77.7% and *F*1 score of

84.5%. More importantly, we achieved these with less extra annotated resources and no special training strategy, compared to the current state-of-the-art methods.

Table 7: Example sentence pairs from MSRP Paraphrase Corpus.

S. No.	Sentence 1	Sentence 2	Gold annotation	Prediction	Remark
1	Ricky Clemons' brief, troubled Missouri basketball career is over.	Missouri kicked Ricky Clemons off its team, ending his troubled career there.	paraphrase	paraphrase	correct
2	But 13 people have been killed since 1900 and hundreds injured.	Runners are often injured by bulls and 13 have been killed since 1900.	non-paraphrase	non-paraphrase	correct
3	I would rather be talking about positive numbers than negative.	But I would rather be talking about high standards rather than low standards.	paraphrase	paraphrase	correct
4	The tech-heavy Nasdaq composite index shot up 5.7 percent for the week.	The Nasdaq composite index advanced 20.59, or 1.3 percent, to 1,616.50, after gaining 5.7 percent last week.	non-paraphrase	paraphrase	incorrect
5	The respected medical journal Lancet has called for a complete ban on tobacco in the United Kingdom.	A leading U.K. medical journal called Friday for a complete ban on tobacco prompting outrage from smokers groups.	non-paraphrase	paraphrase	incorrect
6	Mrs. Clinton said she was incredulous that he would endanger their marriage and family.	She hadn't believed he would jeopardize their marriage and family.	paraphrase	non-paraphrase	incorrect

Table 7 shows some examples of sentence pairs that our approach has classified both correctly and incorrectly. Sentence pair no. 1 was correctly classified as "paraphrase", even though the sentences do not have many words in common. Sentence pair no. 2 was correctly classified as "non-paraphrase", even though the two sentences have four words in common words and share the context. Conversely, sentence pair no. 5 was incorrectly predicted as "paraphrase". This pair is difficult to classify correctly for humans. Sentence pair no. 6 was incorrectly classified as "non-paraphrase". The main reason for this misclassification is the presence of possibly rare words, such as incredulous, jeopardize, endanger, which seemed to have made this sentence pair hard to classify.

In summary, it seems that our proposed approach is able to capture the semantic information from clean texts, just as it was when analyzing tweets. This can, in turn, help in correctly classifying pairs that would otherwise be difficult by only looking at the syntactic contents. There are, however, cases that are hard to classify due to both the lack a complete vocabulary and common-sense knowledge.

6.3. Hyperparameter Setting

Hyperparameters were chose by rough investigations into the training data to choose optimization algorithm, learning rates, regularization, and size of training dataset. We empirically optimized the hyperparameters using grid-search, where each model was evaluated on a separate validation-set. We chose the

parameters with which the model performed best. In Figure 3, we demonstrate the network architecture of the proposed model. It is important to note that the convolutions over sentence matrices for S1 and S2 are the same, and their weights are identical.
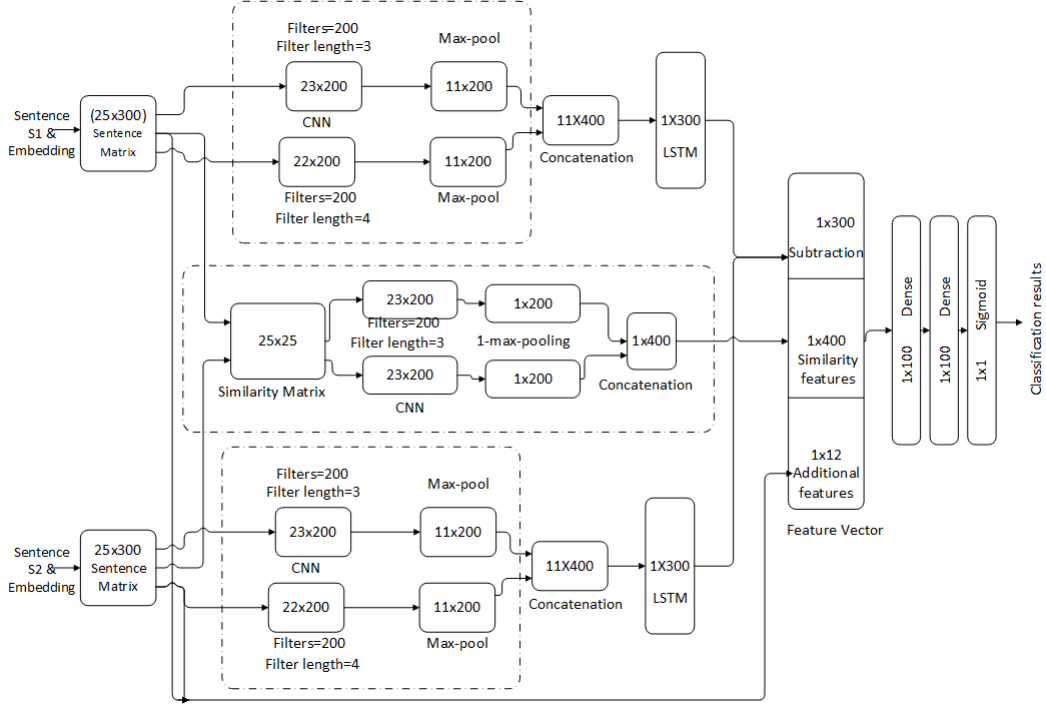

Figure 3: Network architecture

The optimal settings for these hyperparameters vary between datasets, hence we choose separately for the Twitter and MSRP datasets.

6.3.1. Hyperparameter settings on the Twitter dataset

We empirically experiment with various optimizers, see Figure 4a, and chose *Adadelta* to optimize the learning process. We further tune the learning rate for this optimizer, see Figure 4b, with learning rate 0.70 appearing to be optimal. Dropout is used for regularization of the proposed model. This prevents feature co-adaptation by randomly setting a portion of the hidden units to zero during training. We applied dropout to every layer, and set the dropout ratio to 0.2, cf. Figure 4c. Finally, we investigate the sensitivity of the approach with respect to the amount of training data supplied. Figure 4d shows the learning curve, i.e., the learning quality as a function of the amount of the training data used. We clearly see an increasing trend in the learning curve, which indicates that more training data may further improve the performance of the proposed model.

In the absence of a large supervised training set, it is common to initialize word embeddings with pretraining values that have been obtained from an unsupervised neural language model [46]. We follow this strategy, and used the popular `glove` embeddings[3] during our experiments on Twitter dataset. We chose the embeddings pretrained on 2 billion tweets, and use the 200-dimensional version.

6.3.2. Hyperparameter settings for MSRP dataset

The parameter selection process for the MSRP dataset is similar to what was discussed for the Twitter data above, see Figure 5 for results. For this dataset we chose the *Adadelta* optimizer with learning rate set

---

[3]The embeddings are available at https://nlp.stanford.edu/projects/glove/.

(a) Performance of optimization algorithms

(b) Performance vs. learning rate
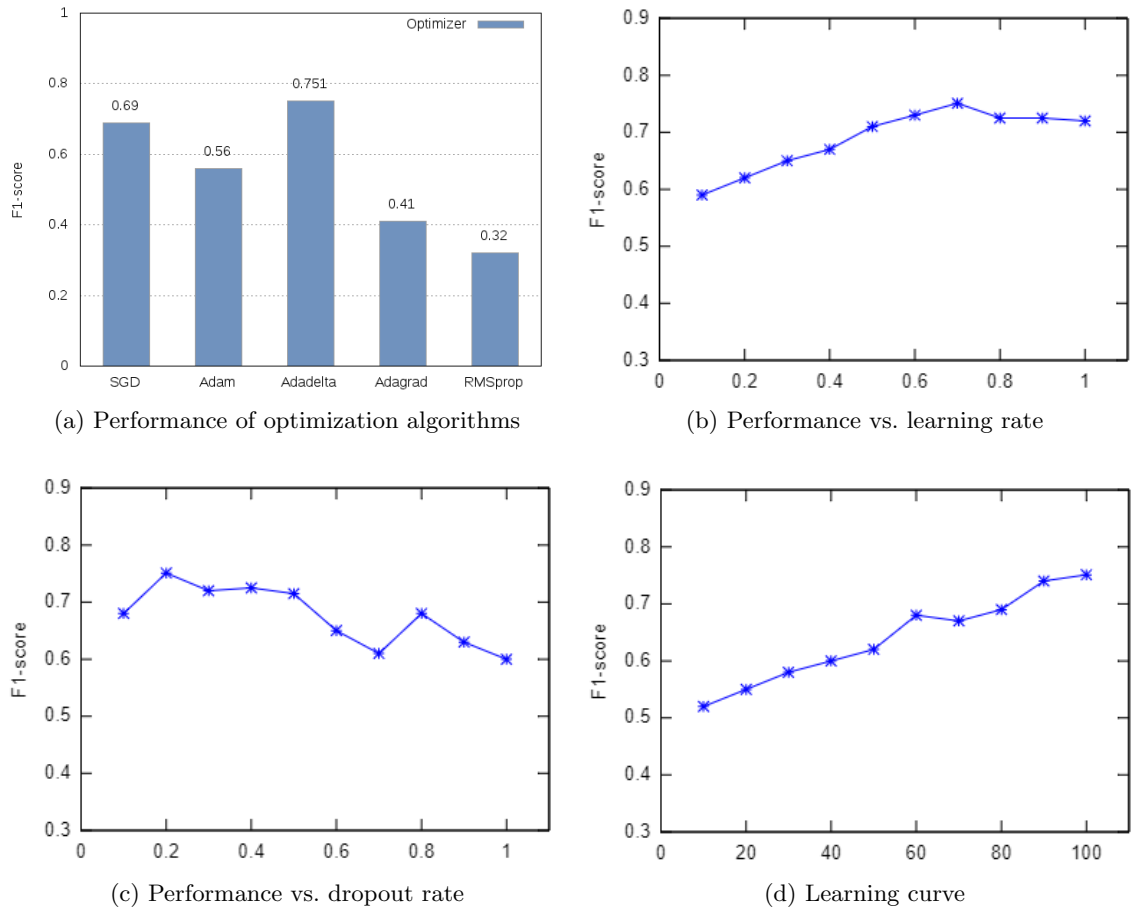
(c) Performance vs. dropout rate

(d) Learning curve

Figure 4: Evaluation of different hyperparameters for the SemEval Twitter dataset

to 0.9. The dropout-rate was chosen to be 0.5. When examining the effect of the size of the training data, we can again see an increasing trend both with respect to accuracy, and (somewhat less pronounced) with respect to F1-score. Further increase in the training dataset would therefore provide slight improvements in the final performance of the model also on this dataset. We used the 300-dimensional version of the publicly available[4] Google *word2vec* vectors [47] to initialize the word embeddings. These vectors are trained on 100 billion words from Google News using the continuous bag-of-words architecture. There is a reason to use different word embeddings both the datasets. As Google's pre-trained word embeddings generally do not work well for twitter dataset as google's embeddings are generated by training on clean text, such as news corpus. When used on twitter data, most of the words in tweets would not be matched in the Google's embeddings, thus causing out-of-vocabulary issues. On the other hand, Stanford provides pre-trained word embeddings (GloVe) which were trained on 2 Billion tweets. Therefore, it works better on twitter corpus than the Google's pre-trained word embeddings. Similar to our approach, the best performing results of paraphrase detection by Dey et al. [5] also used Stanford's GloVe for Twitter dataset.

7. Conclusions

In this paper, we introduced a robust and generic paraphrase detection model based on deep neural network model, which is able to perform well on both user-generated noisy short texts such as tweets,

---

[4] https://code.google.com/archive/p/word2vec/

(a) Performance of optimization algorithms

(b) Performance vs. learning rate

(c) Performance vs. dropout rate
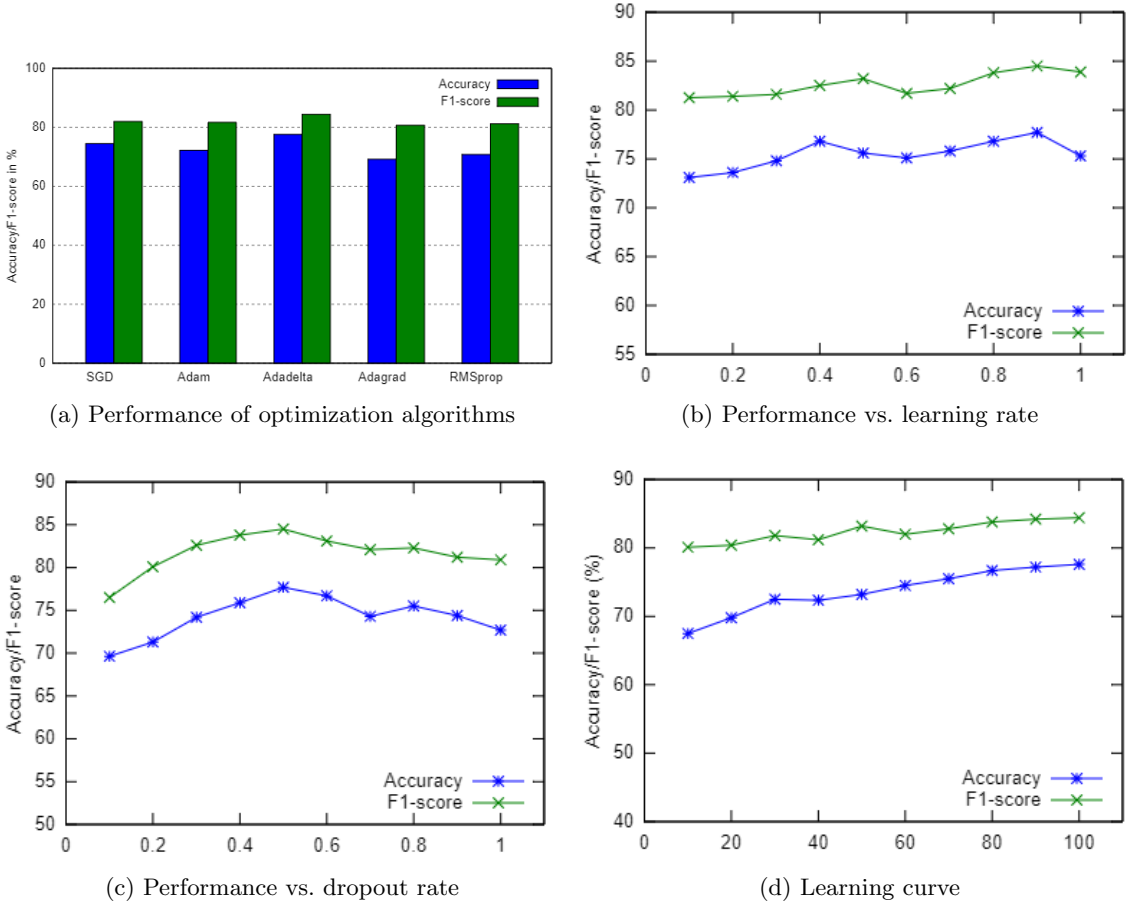
(d) Learning curve

Figure 5: Evaluation of different hyperparameters for the MSRP dataset

and high-quality clean texts. We proposed a pair-wise word similarity model, which can capture fine-grained semantic corresponding information between each pair of words in given sentences. In addition, we used a hybrid deep neural network that extracts coarse-grained information by developing best semantic representation of the given sentences based on CNN and RNN. The model that we developed consisted of both sentence modelling and pair-wise word similarity-matching model. As discussed in this paper, this model proved to be useful for paraphrase detection. In our evaluation, we included a comprehensive comparison against state-of-the-art approaches. This showed that our approach produced better results than all the existing approaches, in terms of F1-score, when applied on noisy short-text Twitter Paraphrase corpus; and provided very competitive results when applied on clean texts from the Microsoft Paraphrase corpus. Overall, our experimental results have shown the robustness and effectiveness of the proposed method for paraphrase detection.

For future work, we plan to investigate how our method works on related tasks such as question answering, sentence matching and information retrieval. Further, we will study the impacts of applying *FastText* as word embeddings on the overall paraphrase detection performance. FastText is trained on character *n-grams*, which could make it more robust against misspellings and noisy text, in general. Finally, we will study the robustness of the proposed approach under various noise degrees that are artificially inserted.

17

References

[1] B. Dolan, C. Quirk, C. Brockett, Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources, in: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), Association for Computational Linguistics, 350, 2004.

[2] W. Xu, A. Ritter, C. Callison-Burch, W. Dolan, Y. Ji, Extracting Lexically Divergent Paraphrases from Twitter, Transactions of the Association for Computational Linguistics 2 (2014) 435–448, ISSN 2307-387X.

[3] T. Kajiwara, D. Bollegala, Y. Yoshida, K.-i. Kawarabayashi, An iterative approach for the global estimation of sentence similarity, PloS one 12 (9) (2017) 1–15.

[4] W. Xu, C. Callison-Burch, B. Dolan, SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT), in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015), Association for Computational Linguistics, 1–11, 2015.

[5] K. Dey, R. Shrivastava, S. Kaushik, A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs, in: Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016), Association for Computational Linguistics, 2880–2890, 2016.

[6] Y. Ji, J. Eisenstein, Discriminative Improvements to Distributional Sentence Similarity, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), Association for Computational Linguistics, 891–896, 2013.

[7] N. Madnani, J. Tetreault, M. Chodorow, 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 (NAACL HLT 2012), Association for Computational Linguistics, 182–190, 2012.

[8] D. Das, N. A. Smith, Paraphrase Identification As Probabilistic Quasi-synchronous Recognition, in: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL 2009), Association for Computational Linguistics, 468–476, 2009.

[9] M. Sahi, V. Gupta, A Novel Technique for Detecting Plagiarism in Documents Exploiting Information Sources, Cognitive Computation 9 (6) (2017) 852–867.

[10] V. K., D. Gupta, Unmasking text plagiarism using syntactic-semantic based natural language processing techniques: Comparisons, analysis and challenges, Information Processing and Management 54 (3) (2018) 408–432.

[11] Y. Jiang, W. Bai, X. Zhang, J. Hu, Wikipedia-based information content and semantic similarity computation, Information Processing and Management 53 (1) (2017) 248–265.

[12] M. Franco-Salvador, P. Rosso, M. Montes-y Gómez, A systematic study of knowledge graph analysis for cross-language plagiarism detection, Information Processing and Management 52 (4) (2016) 550–570.

[13] S. Arora, Y. Liang, T. Ma, A Simple but Tough-to-beat Baseline for Sentence Embeddings, in: Proceedings of the 5th International Conference for Learning Representations (ICLR 2017), URL http://arxiv.org/abs/1703.02507, 2017.

[14] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146, ISSN 2307-387X.

[15] M. Pagliardini, P. Gupta, M. Jaggi, Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features, CoRR abs/1703.02507, URL http://arxiv.org/abs/1703.02507.

[16] Y. Kim, Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), Association for Computational Linguistics, 1746–1751, 2014.

[17] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, S. Fidler, Skip-thought Vectors, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS'15, MIT Press, 3294–3302, 2015.

[18] Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, Character-Aware Neural Language Models, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016), AAAI Press, 2741–2749, 2016.

[19] X. Wang, W. Jiang, Z. Luo, Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts, in: Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016), Association for Computational Linguistics, 2428–2437, 2016.

[20] M. Heilman, N. A. Smith, Tree edit models for recognizing textual entailments, paraphrases, and answers to questions, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT 2010), Association for Computational Linguistics, 1011–1019, 2010.

[21] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, C. D. Manning, Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection, in: Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS 2011), Curran Associates Inc., 801–809, 2011.

[22] J. Oliva, J. I. Serrano, M. D. del Castillo, Á. Iglesias, SyMSS: A syntax-based measure for short-text semantic similarity, Data and Knowledge Engineering 70 (4) (2011) 390–405, ISSN 0169-023X.

[23] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional Neural Network Architectures for Matching Natural Language Sentences, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, MIT Press, 2042–2050, 2014.

[24] E.-S. M. El-Alfy, R. E. Abdel-Aal, W. G. Al-Khatib, F. Alvi, Boosting paraphrase detection through textual similarity metrics with abductive networks, Applied Soft Computing 26 (2015) 444–453.

[25] S. Filice, A. Moschitti, Learning to Recognize Ancillary Information for Automatic Paraphrase Identification, in: Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016), Association for Computational Linguistics, 1109–1114, 2016.

[26] M. AL-Smadi, Z. Jaradat, M. AL-Ayyoub, Y. Jararweh, Paraphrase Identification and Semantic Text Similarity Analysis

in Arabic News Tweets Using Lexical, Syntactic, and Semantic Features, Information Processing and Management 53 (3) (2017) 640–652.

[27] Z. Wang, H. Mi, A. Ittycheriah, Sentence Similarity Learning by Lexical Decomposition and Composition, in: Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016), Association for Computational Linguistics, 1340–1349, 2016.

[28] I. Lopez-Gazpio, M. Maritxalar, A. Gonzalez-Agirre, G. Rigau, L. Uria, E. Agirre, Interpretable semantic textual similarity: Finding and explaining differences between sentences, Knowledge-Based Systems 119 (2017) 186–199.

[29] W. Yin, H. Schütze, B. Xiang, B. Zhou, ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs, Transactions of the Association for Computational Linguistics 4 (2016) 259–272.

[30] Z. Wang, W. Hamza, R. Florian, Bilateral Multi-Perspective Matching for Natural Language Sentences, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017), ijcai.org, 4144–4150, 2017.

[31] R. Ferreira, G. D. Cavalcanti, F. Freitas, R. D. Lins, S. J. Simske, M. Riss, Combining sentence similarities measures to identify paraphrases, Computer Speech and Language 47 (2018) 59–73.

[32] A. Severyn, A. 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, SIGIR'15, 373–382, 2015.

[33] T. Kenter, M. de Rijke, Short Text Similarity with Word Embeddings, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM'15, ACM, 1411–1420, 2015.

[34] A. Eyecioglu, B. Keller, Twitter Paraphrase Identification with Simple Overlap Features and SVMs, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015), Association for Computational Linguistics, 64–69, 2015.

[35] J. Zhao, M. Lan, ECNU: Leveraging Word Embeddings to Boost Performance for Paraphrase in Twitter, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015), Association for Computational Linguistics, 34–39, 2015.

[36] G. Zarrella, J. C. Henderson, E. M. Merkhofer, L. Strickhart, MITRE: Seven Systems for Semantic Similarity in Tweets, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015), Association for Computational Linguistics, 12–17, 2015.

[37] M. Karan, G. Glavas, J. Snajder, B. D. Basic, I. Vulic, M. Moens, TKLBLIIR: Detecting Twitter Paraphrases with TweetingJay, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015), Association for Computational Linguistics, 70–74, 2015.

[38] J. Wieting, K. Gimpela, Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), Association for Computational Linguistics, 2078–2088, 2017.

[39] N. P. A. Vo, S. Magnolini, O. Popescu, Paraphrase Identification and Semantic Similarity in Twitter with Simple Features, in: Proceedings of the 3rd International Workshop on Natural Language Processing for Social Media (SocialNLP 2015@NAACL-HLT), Association for Computational Linguistics, 10–19, 2015.

[40] J. Huang, S. Yao, C. Lyu, D. Ji, Multi-Granularity Neural Sentence Model for Measuring Short Text Similarity, in: Proceedings of the 22nd International Conference on Database Systems for Advanced Applications (DASFAA 2017), Springer, 439–455, 2017.

[41] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and Knowledge-based Measures of Text Semantic Similarity, in: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), AAAI Press, 775–780, 2006.

[42] W. Guo, M. Diab, Modeling Sentences in the Latent Space, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012), Association for Computational Linguistics, 864–872, 2012.

[43] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML 2010), Omnipress, 807–814, 2010.

[44] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (8) (1997) 1735–1780.

[45] G. S. Tomar, T. Duque, O. Täckström, J. Uszkoreit, D. Das, Neural Paraphrase Identification of Questions with Noisy Pretraining, in: Proceedings of the 1st Workshop on Subword and Character Level Models in NLP, Association for Computational Linguistics, 142–147, 2017.

[46] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural Language Processing (Almost) from Scratch, Journal of Machine Learning Research 12 (2011) 2493–2537, ISSN 1533-7928.

[47] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013), Curran Associates Inc., 3111–3119, 2013.