A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs

Kuntal Dey
IBM Research India
New Delhi, India
kuntadey
@in.ibm.com

Ritvik Shrivastava
NSIT Delhi
New Delhi, India
ritviks.it
@nsit.net.in

Saroj Kaushik
IIT Delhi
New Delhi, India
saroj
@cse.iitd.ac.in

Abstract

Existing systems deliver high accuracy and F1-scores for detecting paraphrase and semantic similarity on traditional clean-text corpus. For instance, on the clean-text Microsoft Paraphrase benchmark database, the existing systems attain an accuracy as high as 0.8596. However, existing systems for detecting paraphrases and semantic similarity on user-generated short-text content on microblogs such as Twitter, comprising of noisy and ad hoc short-text, needs significant research attention. In this paper, we propose a machine learning based approach towards this. We propose a set of features that, although well-known in the NLP literature for solving other problems, have not been explored for detecting paraphrase or semantic similarity, on noisy user-generated short-text data such as Twitter. We apply support vector machine (SVM) based learning. We use the benchmark Twitter paraphrase data, released as a part of SemEval 2015, for experiments. Our system delivers a paraphrase detection F1-score of 0.717 and semantic similarity detection F1-score of 0.741, thereby significantly outperforming the existing systems, that deliver F1-scores of 0.696 and 0.724 for the two problems respectively. Our features also allow us to obtain a rank among the top-10, when trained on the Microsoft Paraphrase corpus and tested on the corresponding test data, thereby empirically establishing our approach as ubiquitous across the different paraphrase detection databases.

1 Introduction

Detecting paraphrase, and more specifically, given a pair of input natural language texts identifying whether one is a paraphrase of the other, has been a challenging research problem. Over a number of years and volumes of research, the paraphrase detection systems have emerged as robust and well-performing ones, especially for "clean text corpus", such as the Microsoft Paraphrase Corpus (Dolan et al., 2004). Semantic similarity detection has been another related problem of interest, where the objective is to identify how similar is one text with respect to another (Harispe et al., 2015). Since, paraphrases are expected to have significant semantic similarity (Corley and Mihalcea, 2005) (Mihalcea et al., 2006), systems performing well for paraphrase detection can also be intuitively expected to perform well for semantic similarity detection of a pair of text inputs. It is worth noting that, for benchmark "clean text corpus" data such as the Microsoft Paraphrase Corpus database, the paraphrase detection systems deliver an impressive performance, with a high F-score of 0.8596.

While the literature around paraphrase and semantic similarity detection has matured along the directions of clean text corpus, much work remains to be done to attain commendable performances in the paradigm of noisy short-text inputs, such as user-generated short-text content found on Twitter. Some initial work has been recently carried out on the benchmark SemEval 2015 data (Xu, 2014) (Xu et al., 2015); however, the performance of the systems developed till date leave much desired. Detecting paraphrases and semantic similarity on such text is inherently difficult; however, we believe there is scope for improvement over the current 0.724 performance, that the current state of the art delivers (Xu et al., 2014). Given the well-understood importance of paraphrase detection in multiple fields of NLP, such as

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2880–2890, Osaka, Japan, December 11-17 2016.

review summarization, opinion mining and information matching, to name a few, and practical applications such as first-story detection (Petrović et al., 2012), it is of paramount importance to develop highly accurate paraphrase detectors for such user-generated noisy short-text corpus.

This motivates us to attempt to improve the performance of detecting paraphrases and semantic similarity on user-generated short-text on noisy platforms such as Twitter. We propose a feature-set driven approach, and use support vector machine (SVM) based machine learning. We rely upon lexical, syntactic, semantic and pragmatic features of a given pair of tweets, to label whether these two tweets are paraphrases of each other. We further obtain the semantic similarity scores of the pairs, and thereby assign semantic similarity labels. We validate our work against the benchmark SemEval 2015 data, and find our system (F1-score 0.741) to outperform the known state of the art (best known F1-score 0.724), that includes all of the feature-based approaches as well as deep-learning based approaches.

The main contributions of our work are the following.

- We provide a machine learning based model, and a set of lexical, syntactic, semantic and pragmatic features, for detecting paraphrases on user-generated noisy Twitter short-text content data.
- We empirically show the goodness of the proposed features, on a benchmark Twitter paraphrase corpus. Our system outperforms all the systems that exist in the state of the art.
- We further demonstrate the effectiveness of our feature set on benchmark clean-corpus text data, namely Microsoft Paraphrase dataset, and obtain a rank within the top 10 existing systems, by training our system (our features) on their dataset, and testing on their dataset as well. This empirically establishes the goodness of our proposition across different types of text.

2 Related Work

The importance of detecting paraphrases for different information processing applications, has been well-accepted by researchers. Over the long-standing history of research on paraphrase detection, a volume of significant work has been carried out.

Substantial work has been carried out in the space of detecting paraphrases on traditional clean-text corpus, by several researchers. The Microsoft Paraphrase corpus (Dolan et al., 2004), that presents sentence pairs from newswire text, serves as the long-standing baseline for such text. As observed by (Ji and Eisenstein, 2013), there are three high-level approaches. (1) String similarity metrics, comprising of n-gram overlaps and BLEU features, that have been proposed by (Wan et al., 2006), and (Madnani et al., 2012). (2) Parse structure based syntactic operations, such as by (Wu, 2005), and (Das and Smith, 2009). (3) Distributional methods, such as latent semantic analysis (LSA), by (Landauer et al., 1998). (Kauchak and Barzilay, 2006) propose a distributionally similar alternative based approach. (Socher et al., 2011) propose a feature auto-encoder based approach, combining word representations, building upon recursive auto-encoding. (Blacoe and Lapata, 2012) demonstrate the effectiveness of combining latent representations with simple element-wise operations, for the purpose of identifying semantic similarity amongst larger text units. (Ji and Eisenstein, 2013) propose a discriminative term weighting metric, namely TF-KL_D to outperform TF-IDF, and show that "using the latent representation from matrix factorization as features in a classification algorithm substantially improves accuracy". They combine the latent features with fine-grained n-gram overlap features, improving the state of the art significantly.

Recently, a few initial works have been carried out for detecting paraphrases and semantic similarities on user-generated noisy social network and microblog short-text, such as Twitter. In a recent seminal paper, (Xu et al., 2014) proposed a joint word-sentence approach based model, and applied a multi-instance learning assumption (Dietterich et al., 1997) that "two sentences under the same topic are paraphrases if they contain at least one word pair that is indicative of sentential paraphrase". They observe that the "at least one anchor" overlap works well for short text situations, such as on Twitter, though they are not necessarily effective for traditional long texts, and thereby design a graphical model to discriminatively determine whether a given pair of words are paraphrases. They create a dataset (the collection of which is found in further detail in (Xu, 2014)), and empirically demonstrate their system. They obtain an F1-score

for semantic similarity of 0.724 by their system, and claim the human upper bound to have an F1-score of 0.823. While they do not report their paraphrase detection score, our experimentations using the data[1] and the corresponding released code[2] finds their system to yield an F1-score of 0.696.

Subsequently, a number of researchers attempted to solve the problem of paraphrase detection, as part of the task proposed by (Xu et al., 2015). The model by (Eyecioglu and Keller, 2015) proposes to have unigram and bigram features for characters and words, and explores the overlap (and non-overlap) of these features for determining whether one tweet is a paraphrase of the other. While the authors do not address the semantic similarity detection problem, the paraphrase detection problem yields an F1-score of 0.674. In another work applied on the same data, (Zarrella et al., 2015) obtain a paraphrase detection F1-score of 0.667, and a semantic similarity score of 0.724. They use "mixtures of string-matching metrics, tweet-specific distributed word representation alignments, recurrent neural networks to model similarity between those alignments, and distance measurements on pooled latent semantic features", and tie these systems together using logistic regression. In addition to using string based, corpus based and syntactic features, (Zhao and Lan, 2015) propose "novel features based on distributed word representations, learned using deep learning paradigms". They attained an F1-score of 0.662 for paraphrase detection on the given dataset.

As apparent from the above discussion, the state of the art for paraphrase detection on noisy short-text can potentially be significantly improved. Different classes of features, with different implications, need to be applied for making such improvement. Our work addresses this, and also attempts to explore the applicability of the proposed model across other, clean-text corpus. Our system, with its feature set that have not been used on noisy Twitter data before, outperforms the state of the art systems for both paraphrase detection as well as semantic similarity identification.

3 Methodology

We choose a machine-learning based approach to solve the problem at hand. We identify several features of different types, and apply Support Vector Machine (SVM) based learning.

3.1 Preprocessing

While by itself we do not use any *preprocessing feature* per se, we carry out preprocessing steps that impact the performance of our overall system. The preprocessing steps include the following.

- **Topic phrase removal:** We remove the topic phrase from the tweet pairs, as this does not provide any information to distinguish given tweet pairs, but causes unnecessary gram overlaps. To illustrate with an example, in the test dataset, a given pair of texts are: *"Which Star Wars episode should I watch"* and *"I have so much of a life that Im at home watching Star Wars"*. The topic given here, explicitly noted as part of the dataset, is *Star Wars*.

- **Tweet normalization using net slang and Han-Baldwin dictionary:** We normalize the tweets, using net slang and Han-Baldwin normalization dictionary knowledge (Han and Baldwin, 2011). This would help resolve non-dictionary colloquial expressions. E.g.: *aaf* maps to *as a friend*. We use an online version of one of the many net slang dictionaries that are available today. For experiments, we use an online net slang dictionary.

- **Named Entity (NE) boundary correction:** We align NE boundaries across tweets, for appropriate matching. For example, for a given pair of tweets having the same topic, words like *Colorado Ravens* and *Ravens* probably convey the same concept, but would create different gram features. For experimentation, we use the NE tags provided in the dataset, in the CoNLL IOB format.

- **NE tag cleaning:** Inconsistent capitalization of tweets lead to named entity recognition (NER) errors, jeopardizing system performance. For example, if within a given pair of tweets with the

[1] https://github.com/cocoxu/SemEval-PIT2015, crawled on July 10, 2016
[2] https://github.com/cocoxu/multip, crawled on July 10, 2016

same topic, one tweet consists of a named entity *Colorado Ravens* and the other one has the word *ravens* with a NE-tag **O**, they are likely to be discussing the same Raven, with the inconsistency of capitalization confusing the named entity recognizer. We clean NE tags by cross-checking the given pair of tweets, and matching unigrams and bigrams in a case-insensitive manner.

- **Synonym and hypernym replacement using Wordnet:** We use Wordnet (Miller, 1995) to carry out a synonym and hypernym replacement process. For a pair of given tweets, we aim to unify the diverse words appearing in the pair of input texts (tweets), by replacing a word with its synonym (or hypernym). For instance, if *Tweet 1* contains the text *"He made a fast move"* and if *Tweet 2* contains the text *"He performed a quick move"*, then the content of the second tweet (*tweet 2*) gets updated to *"He performed a fast move"*, as long as "fast" and "quick" appear as synonyms in Wordnet. This in turn is fed to the main processing stream for feature selection. This replacement process proves to be useful for adjectives and verbs, so the synonym replacement process considers all the words having adjective and verb POS tags.

3.2 Feature Selection

The challenges to perform the task of paraphrase and semantic similarity detection, exist at different layers of NLP. These include several challenges at (a) lexical, (b) syntactic, (c) semantic and (d) pragmatic levels. We observe that the existing systems, that have specifically attempted to solve the paraphrase and/or semantic similarity detection problem for noisy short-text user generated content platform as Twitter, such as (Xu et al., 2014) and (Eyecioglu and Keller, 2015) *etc.*, have opportunities to be improved, by enhancing the features to solve for all these levels. With this observation, we attempt to explore features across these dimensions, as described below.

3.2.1 Lexical Features

Character-level gram features: We create character-level gram features, notionally borrowing from the features described by (Eyecioglu and Keller, 2015), and thereby include the number of overlaps of character trigrams as features. We further include the number of character trigrams in each of the two input texts (tweets), the size of union of character trigrams over the pair of input texts, the size of intersection, and the size of difference of number of character trigrams - in the form of (a) the number of trigrams that occur in the first text but not the second, and (b) those which occur in the second text but not the first - as features.
Word-level gram features: Apart from character-grams, we also construct word-level gram features, again borrowing in principle from (Eyecioglu and Keller, 2015). We construct features for word-level unigrams, bigrams and trigrams, that are similar in nature with the character gram features, in that, we consider the number of grams in each of the two input texts, and the union, intersection and difference sizes. Further, we also consider **unordered gram overlap** features for word grams, rewarding the system for unordered match in word bigrams and trigrams. This significantly helps the system performance.
Stemming: We repeat the use of the word-level gram features, after having stemmed the text (tweet content). We use the Porter stemmer (Porter, 2001) for performing the stemming in our experimentation.
Stopword removal: We perform stopword removal, using a list of well-accepted set of stopwords. We repeat the use of the word-level gram features after stopword removal.
String-level features: Stemming and Han-Baldwin normalization, done as part of the earlier steps, are two of the string-level features used by our methodology. Further, motivated by (Xu et al., 2014), we use the Jaro-Winkler string similarity (Winkler, 1999) of each given pair of input texts, as a feature.

3.2.2 Syntactic Features

Part-of-Speech (POS) agreement features: We construct two features under this subcategory, that attempt to capture the POS agreement features across the pair of input texts (tweets).

- **Number of matching words with matching POS:** We count the number of word unigrams across the pair of texts, where both the actual word as well as the POS tags of the pair of words, match with each other. We use this count as a feature. We empirically observe that the preprocessing step

of performing synonym replacement with Wordnet, significantly improves the system performance, when deployed along with this feature.

- **Verb similarity:** We count the number of verbs in each of the two texts, where the verb tag (such as VB, VBP, VBZ, VBN *etc.*) match with each other, and use this count as a feature.

Named entity (NE) features: Using the preprocessed named entities, we explore the degree of overlap of NEs across the tweet pairs, over and beyond the core topic phrase. Note that the core topic words are often named entities in our case, and hence they are removed earlier in the preprocessing phase.

3.2.3 Semantic Features

We extensively use Wordnet (Miller, 1995), to enable the construction of our semantic features.
Word overlap features: We compute the degree of overlap between words that appear across the pair of input texts, considering their semantics. Note that, this is inherently different from the word gram features computed as part of the lexical features, as the lexical features are agnostic of semantic attributes of words. Using Wordnet, we construct two features under this subcategory, as given below.

- **Best word overlap:** For each pair of words of a given POS that appear in each of the input texts (tweets), we compute the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of each word of the pair, as well as the union of the synonym and hypernym set. We repeat the above for the stemmed words. If one word is a part of the synonym or hypernym set of the other word, then we count the word as a whole (with $weight = 1$), and the value of this feature becomes unity (1). Otherwise, if none of the two words is included in any of the synonym or hypernym sets of the other, then we use the sum of Jaccard coefficients of the above-computed overlap, of the combination of the given words and stemmed words, as the value of the best word overlap feature. Thus, for the four different POS tags we consider, namely noun, verb, adjective and adverb, this gives us a set of four different features.

- **Adjective overlap:** For each pair of words tagged as adjectives by the POS tagging process, we compute the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of each word of the pair, as well as the union of the synonym and hypernym set. Just like in the case of finding the best word overlap feature, we repeat the above for the stemmed words. We use the sum of Jaccard coefficients of the above-computed overlap, of the combination of the given words and stemmed words, as a feature. Note that, in this feature, even if an adjective from one text appears directly as a synonym or hypernym of the other adjective from the other text, we never consider the feature value to be unity (which is captured by the best word overlap feature separately). While intuitively this feature is extremely "close" to the best word overlap feature, we still retain this feature as during the feature construction (development) process, we observe this feature to be beneficial.

Phrase overlap features: We compute the degree of direct overlap of (a) noun phrases (b) and verb phrases across tweet pairs. We also use the stemmed versions to compute stemmed phrase overlaps. Treating each input text (tweet) as a disconnected graph and each phrase as a vertex in the graph, we connect the pairs of graphs (one graph for each tweet) using the best matching phrase. The matching of a pair of phrases is carried out, by computing the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of the individual words that appear in the phrase pair. Similar to the case of the best word overlap feature, we set the value of this feature of unity if, any one word belonging to one phrase, is directly included in the set union of synonyms and hypernyms of any one word belonging to the other phrase. Otherwise, we compute the sum of Jaccard coefficients of the above-computed overlap for all the word pairs across the text (tweet) pairs, and use this sum as the value of the feature. This is constructed for noun and verb phrase types, thereby adding two features to the overall set of features. Effectively, this is a manifestation of MAXSIM (Chan and Ng, 2008), that is used in the machine translation paradigm.

3.2.4 Pragmatic Features

Subjectivity/objectivity agreement feature: In one implementation, we include the agreement of the nature of the tweet pair with respect to subjectivity/objectivity, as a boolean feature. We make use of the MPQA dictionary, that indicates strong and weak subjectivity and objectivity of words, to construct this feature. The final experimental results we present for the Twitter, does not include this feature, when we attain an F1-score of 0.741; however, if we compromise our F1-score on the Twitter noisy short-text data to 0.740, we obtain a marginal lift from 0.824 to 0.825 on the Microsoft Paraphrase data, helping our system outperform other systems that also yield an F1-score of 0.824 on the Microsoft Paraphrase data.
Close attachment of negations: We also perform close attachment of negations, by attaching the semantic sense of a negation (not *etc.*) with an appropriate term, using well-known techniques from the literature. This in turn assigns appropriate polarity to the subjectivity features.

4 Experiments

In this section, we provide the details of the experiments we conducted.

4.1 Data Description and Tools Used

For experimentation, we select the benchmark dataset by (Xu, 2014) provided as part of the SemEval 2015 task (Xu et al., 2015). This dataset has been used by all the recent works in this space, and further, existing paraphrase detection algorithms, developed optimizing for clean corpus and tested with other clean-text datasets such as the Microsoft Paraphrase database, have also been tested on this dataset. Thus, it provides a well-tested foundation for empirically observing the performance of our system, and benchmarking the performance of our system against the other existing systems.

The original dataset of (Xu et al., 2014) is shown on Table 1. Note that, the dataset based on which the final scores are reported by (Xu et al., 2014), which is also released as part of SemEval 2015 (Xu et al., 2015), consists of 838 tweets in the test set, as opposed to the 972 that were present in the original dataset. This is arrived at, by ignoring the 134 "debatable" entries, that were marked in (Xu et al., 2015), All the systems that use this dataset to report their scores, are against the modified test dataset with 838 test entries ignoring the "debatable" ones, and our reports are also based upon this modified dataset. We used Weka (Hall et al., 2009) for machine learning, and used the POS and NE tags already provided by the SemEval 2015 (Xu et al., 2015) dataset.

	Num. Unique Sentences	**Num. Pair of Sentences**	**Num. Paraphrases**	**Num. Non-Paraphrases**	**Num. Debatable**
Train	$13,231$	$13,063$	$3,996$ (30.6%)	$7,534$ (57.7%)	$1,533$ (11.7%)
Dev	$4,772$	$4,727$	$1,470$ (31.1%)	$2,672$ (56.5%)	585 (12.4%)
Test	$1,295$	972	175 (18.0%)	663 (68.2%)	134 (13.8%)

Table 1: The benchmark dataset released by SemEval 2015 (Xu et al., 2015).

4.2 Performance of Our System

As detailed in Section 3, we train our SVM model on the training data provided, using the identified features. After identifying the effective features, we train our final model on the combination of training and development data provided, and subsequently execute the model to benchmark our system on the given test data. The baseline features comprise of word unigrams and bigrams, and Jaro-Winkler string similarity features. The performance, as well as the impact of each feature (as observed by feature ablation tests) of our system, are provided in Table 2.

Some example cases have been illustrated in Table 3, including cases of correct and incorrect detections by our system. One glance at the examples make it obvious that, in most cases, it is challenging to label the text pairs as paraphrases versus otherwise. Thus, these examples not only illustrate instances of successes and failures of our system, but also provides an instinctive view of the magnitude of challenge, that a solution to the current problem needs to overcome.

Features Used in Model	Semantic Similarity			Paraphrase		
	F1	P	R	F1	P	R
Baseline	0.641	0.667	0.617	0.523	0.67	0.429
+Trigrams	0.663	0.701	0.629	0.533	0.675	0.44
+Topic Removal	0.708	0.739	0.68	0.705	0.707	0.703
+WordNet Synonym Replacement	0.718	0.747	0.691	0.709	0.683	0.737
+POS features	0.721	0.734	0.709	0.709	0.683	0.737
+Phrase Overlap features	0.735	0.758	0.714	0.717	0.697	0.737
+NE features (Final Model)	**0.741**	0.756	0.726	**0.717**	0.697	0.737

Table 2: Performance of our system and impact of features (based upon feature ablation tests). F1 ← F1-score. P ← Precision. R ← Recall.

4.3 Comparing with existing systems

We compare the performance of our system with the existing systems. (Xu et al., 2014) summarize the performances of prior existing paraphrase and semantic similarity detection systems, trained on clean-text corpus, when tested on noisy user-generated short-text data on Twitter. We further compare our results with the existing systems in the literature that are specifically trained on the Twitter paraphrase corpus, and benchmark our performance. Table 4 provides a summary of the results that are attained by the existing systems, trained and tested on the same (SemEval 2015) dataset as ours, as well as our by our methodology. Clearly, our system outperforms all the others in terms of the F1-score it achieves. Although (Zhao and Lan, 2015) obtain a higher precision and (Zarrella et al., 2015) obtain a higher recall compared to ours, our system ranks #2 in terms of precision as well as in terms of recall, and #1 when the precision and recall are combined to obtain an F1-score.

4.4 Applying our feature set on clean-corpus paraphrase detection text

In order to obtain empirical insights about the goodness of the proposed feature set, for the task of paraphrase detection on clean-corpus data, we port the features to the Microsoft Paraphrase dataset (Dolan et al., 2004). Performing training on the Microsoft Paraphrase dataset with our features, and testing on the corresponding test dataset, we obtain an F1-score of 0.824. We further note that, including the subjectivity feature described in Section 3.2.4, lifts the score to 0.825 on this dataset. According to the ACL wiki for paraphrase identification [3], and also factoring for (Eyecioglu and Keller, 2015) that delivers a higher performance on the Microsoft Paraphrase database[4] compared to our system, this gives us the 10^{th} rank overall for the clean-corpus text, using no additional feature. We note the following.

- The best known system, designed by (Ji and Eisenstein, 2013), currently delivers an F1-score of 0.8596 on the Microsoft Paraphrase dataset, while ours delivers 0.825, without any adaptation. On the other hand, our system at an F1-score of 0.741, delivers a significantly better performance compared to their system with an F1-score of 0.641 on the Twitter data, also without any adaptation. Clearly, our system performs (adapts) "better" when observed in a cross-data-type (clean and noisy text) scenario, compared to theirs. This argument holds for all the other existing systems as well, including (Eyecioglu and Keller, 2015), that delivers an F1-score of 0.674 on the noisy Twitter data and an F1-score of 0.828 on the clean Microsoft Paraphrase data.

- Topic removal, which plays a major in helping our system deliver its high performance (as clear from Table 2), cannot be performed on the Microsoft Paraphrase corpus, since, unlike the Twitter corpus, the topics are not explicitly given, and topic-detection is not a focus of the current work. Detecting and removing topics from the Microsoft Paraphrase dataset, would completely unify the

[3] $http://aclweb.org/aclwiki/index.php?title = Paraphrase\_Identification\_(State\_of\_the\_art)$, crawled on July 10, 2016

[4] Yet to be added to the ACL wiki, as found at the time of writing this paper

Sentence Pair from Twitter	Gold Label	System Label	Confidence	Remark
And the Mets and Marlins go 20 Miami Marlins beat the NY Mets 21 in 20 innings	Paraphrase	Paraphrase	0.8971	Correct
Thank you very much Rafa Benitez Why do liverpool fans love benitez so much	Not Paraphrase	Paraphrase	0.6192	Incorrect
chris davis is 44 with two bombs Chris Davis has 2 home runs tonight	Paraphrase	Not Paraphrase	0.2318	Incorrect
Which Star Wars episode should I watch I have so much of a life that Im at home watching Star Wars	Not Paraphrase	Not Paraphrase	0.2317	Correct
So Roberto Mancini has been officially sacked as Man City s manager UK football manager Roberto Mancini sacked by Manchester City	Paraphrase	Paraphrase	0.9791	Correct
I wanna see the movie after earth NOW YOU SEE ME and AFTER EARTH Cant Outpace FAST FURIOUS 6	Not Paraphrase	Paraphrase	0.6180	Incorrect
Classy gesture by the Mets for Mariano real class shown by The Mets Mo Rivera is a legend	Paraphrase	Not Paraphrase	0.2316	Incorrect
Anyone wanting to go to the JT concert 722 in Chicago just watched season finale of chicago fire and cried	Not Paraphrase	Not Paraphrase	0.2312	Correct

Table 3: Examples of outputs of our system, and its correctness with respect to gold labels. "Confidence" denotes the confidence score, between 0 and 1, of a given pair of tweets being paraphrases.

feature set across the two types of data (clean and noisy), and also might potentially improve the system performance (open for exploration). We propose to explore this in future.

5 Discussion

Choice of Approach: Why not Deep Learning?
We use the traditional methodology of identifying features, and a traditional SVM classifier. The choice of modeling in the given manner is, in the known state of the art, traditional feature-based systems, such as (Xu et al., 2014) and (Eyecioglu and Keller, 2015), have so far outperformed deep learning systems, such as the recurrent neural network based approaches by (Zarrella et al., 2015) and (Zhao and Lan, 2015), for paraphrase detection. A deeper subsequent study by (Sanborn and Skryzalin, 2015) also models deep neural learning systems, and observes a similar shortcoming of such systems. A likely cause of this anomaly is the simple absence of a sufficiently large paraphrase dataset in the domain of user-generated noisy texts on microblogs such as Twitter. As and when a much-larger noisy short-text paraphrase corpus for user generated content on microblogs, such as Twitter, becomes available, a deep-learning model will be likely to deliver stronger performances, and will require a revisit.

The Human Upper Bound and the Practical Improvement Delivered by Our System
We further note that, (Xu et al., 2014) observe the human upper bound for detecting paraphrases (computed as semantic similarity, as per the numerical values they report and the corresponding program code they make available) on noisy short-text of Twitter to be limited to 0.823. Given this human upper bound, improving the F1-score for semantic similarity from the reported 0.724 to the current

Method	F1	P	R	Rank		
				F1-Based	P-Based	R-Based
Random	0.294	0.208	0.500	10	10	10
WTMF (Guo and Diab, 2012)	0.583	0.525	0.655	9	9	5
LR (Das and Smith, 2009)	0.630	0.629	0.632	8	7	6
LEXLATENT (Xu et al., 2014)	0.641	0.663	0.621	7	6	8
LEXDISCRIM (Ji and Eisenstein, 2013)	0.645	0.664	0.628	6	5	7
ENCU (Zhao and Lan, 2015)	0.662	**0.767**	0.583	5	1	9
MITRE (Zarrella et al., 2015)	0.667	0.569	**0.806**	4	8	1
ASOBEK (Eyecioglu and Keller, 2015)	0.674	0.680	0.669	3	3	4
MultiP (Xu et al., 2014)	0.724	0.722	0.726	2	3	2
Our Methodology	**0.741**	0.756	0.726	**1**	2	2

Table 4: A comparison of the performance of different systems on the given Twitter data. F1 ← F1-score. P ← Precision. R ← Recall. Note: a part of the table has been reprinted from (Xu et al., 2014).

0.741 delivers an effective improvement of not $(0.741 - 0.724) * 100\% = 1.7\%$, but a much-higher $((0.741 - 0.724)/0.823) * 100\% = 2.066\%$. This enhances the significance of our work.

6 Conclusion

In this work, we proposed a rich feature set, and performed simplistic SVM-based learning, for performing paraphrase and semantic similarity detection on user generated noisy short-text on social microblogs such as Twitter. We demonstrated the goodness of our system on the benchmark SemEval 2015 database, obtaining an F1-score of 0.717 for paraphrase detection where the known state of the art attains 0.696, and an F1-score of 0.741 for semantic similarity detection where the known state of the art attains 0.724, thus outperforming all the known systems. We show our system to also work well on the benchmark Microsoft Paraphrase database for clean text, and thereby empirically establish our approach as the most ubiquitous system available today, across the different types of paraphrase detection datasets (noisy and clean-text). Our system can be used on applications that need to identify paraphrases and semantic similarities, such as social information spread modeling.

References

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Yee Seng Chan and Hwee Tou Ng. 2008. Maxsim: A maximum similarity metric for machine translation evaluation. In *ACL*, pages 55–62. Citeseer.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics.

Dipanjan Das and Noah A Smith. 2009. 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: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Asli Eyecioglu and Bill Keller. 2015. Asobek: Twitter paraphrase identification with simple overlap features and svms. *Proceedings of SemEval*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 864–872. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Sebastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. 2015. Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8(1):1–254.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 455–462. Association for Computational Linguistics.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 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*, pages 182–190. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2012. Using paraphrases for improving first story detection in news and twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–346. Association for Computational Linguistics.

Martin F Porter. 2001. Snowball: A language for stemming algorithms.

Adrian Sanborn and Jacek Skryzalin. 2015. Deep learning for semantic similarity. *CS224d: Deep Learning for Natural Language Processing. Stanford, CA, USA: Stanford University*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

William E Winkler. 1999. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 25–30. Association for Computational Linguistics.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of SemEval*.

Wei Xu. 2014. *Data-driven approaches for paraphrasing across language variations*. Ph.D. thesis, New York University.

Guido Zarrella, John Henderson, Elizabeth M Merkhofer, and Laura Strickhart. 2015. Mitre: Seven systems for semantic similarity in tweets. *Proceedings of SemEval*.

Jiang Zhao and Man Lan. 2015. Ecnu: Leveraging word embeddings to boost performance for paraphrase in twitter. *Proceedings of SemEval*, page 34.