ECNU: Leveraging Word Embeddings to Boost Performance for Paraphrase in Twitter

Jiang Zhao, Man Lan*
Shanghai Key Laboratory of Multidimensional Information Processing
Department of Computer Science and Technology,
East China Normal University Shanghai 200241, P. R. China
51121201042@ecnu.cn; mlan@cs.ecnu.edu.cn*

Abstract

This paper describes our approaches to paraphrase recognition in Twitter organized as task 1 in Semantic Evaluation 2015. Lots of approaches have been proposed to address the paraphrasing task on conventional texts (surveyed in (Madnani and Dorr, 2010)). In this work we examined the effectiveness of various linguistic features proposed in traditional paraphrasing task on informal texts, (i.e., Twitter), for example, string based, corpus based, and syntactic features, which served as input of a classification algorithm. Besides, we also proposed novel features based on distributed word representations, which were learned using deep learning paradigms. Results on test dataset show that our proposed features improve the performance by a margin of 1.9% in terms of F1-score and our team ranks third among 10 teams with 38 systems.

1 Introduction

Generally, a **paraphrase** is an alternative surface form in the same language expressing the same semantic content as the original form and it can appear at different levels, e.g., lexical, phrasal, sentential (Madnani and Dorr, 2010). Identifying paraphrase can improve the performance of several natural language processing (NLP) applications, such as query and pattern expansion (Metzler et al., 2007), machine translation (Mirkin et al., 2009), question answering (Duboue and Chu-Carroll, 2006), see survey (Androutsopoulos and Malakasiotis, 2010) for completion. Most of previous work of paraphrase are on formal text. Recently with the rapidly growth of microblogs and social media services, the computational linguistic community is moving its attention to informal genre of text (Java et al., 2007; Ritter et al., 2010). For example, (Zanzotto et al., 2011) defined the problem of redundancy detection in Twitter and proposed SVM models based on bag-of-word, syntactic content features to detect paraphrase.

To provide a benchmark so as to compare and develop different paraphrasing techniques in Twitter, the paraphrase and semantic similarity task in *SemEval* 2015 (Xu et al., 2015) requires the participants to determine whether two tweets express the same meaning or not and optionally a degree score between 0 and 1, which can be regarded as a binary classification problem. Paraphrasing task is very close to semantic textual similarity and textual entailment task (Marelli et al., 2014) since substantially these tasks all concentrated on modeling the underlying similarity between two sentences. The commonly-used features in these tasks can be categorized into several following groups: (1) string based which measures the sequence similarities of original strings with others, e.g., *n*-gram `Overlap`, cosine similarity; (2) corpus based which measures word or sentence similarities using word distributional vectors learned from large corpora using distributional models, like *Latent Semantic Analysis* (LSA), etc. (3) knowledge based which estimates similarities with the aid of external resources, such as WordNet; (4) syntactic based which utilizes syntax information to measure similarities; (5) other features such as using Named Entity similarity.

In this work, we built a supervised binary classifier for paraphrase judgment and adopted multiple features used in conventional texts to recognize paraphrase in Twitter, which includes string based features, corpus based features, etc. Besides, we also proposed a novel feature based on distributed word representations (i.e., word embeddings) learned over a large raw corpus using neural language models. The results on test dataset demonstrate that linguistic features are effective for paraphrase in Twitter task and proposed word embedding features further improve the performance.

The rest of this paper is organized as follows. Section 2 describes the features used in our systems. System setups and experimental results on training and test datasets are presented in Section 3. Finally, conclusions and future work are given in Section 4.

2 Feature Engineering

In this section, we describe the our preprocessing step and the traditional NLP linguistic features, as well as the word embedding features used in our systems.

2.1 Preprocessing

We conducted following text preprocessing operations before we extracted features: (1) we recovered the elongated words to their normal forms, e.g., "*goooooood*" to "*good*"; (2) about 5,000 slangs or abbreviations collected from Internet were used to convert these informal texts into their complete forms, e.g., "*1dering*" to "*wondering*", "*2g2b4g*" to "*to good to be forgotten*"; (3) the WordNet-based Lemmatizer implemented in Natural Language Toolkit[1] was used to lemmatize all words to their nearest base forms in WordNet, for example, *was* is lemmatized to *be*. (4) we replaced a word from one sentence with another word from the other sentence if the two words share the same meaning, where WordNet was used to look up synonyms. No word sense disambiguation was performed and all synsets for a particular lemma were considered.

2.2 String Based Features

We firstly recorded length information of given sentences pairs using following eight measure functions: $|A|, |B|, |A-B|, |B-A|, |A \cup B|, |A \cap B|, \frac{(|A|-|B|)}{|B|}, \frac{(|B|-|A|)}{|A|}$ where $|A|$ stands for the number of non-repeated words in sentence A , $|A-B|$ means the number of unmatched words found in A but not in B , $|A \cup B|$ stands for the set size of non-repeated words found in either A or B and $|A \cap B|$ means the set size of shared words found in both A and B .

[1]http://nltk.org/

Motivated by the hypothesis that two texts are considered to be more similar if they share more strings, we adopted the following five types of measurements: (1) longest common sequence similarity on the original and lemmatized sentences; (2) `Jaccard`, `Dice`, `Overlap` coefficient on original word sequences; (3) `Jaccard` similarity using *n*-grams, where *n*-grams were obtained at three different levels, i.e., the original word level (*n*=1,2,3), the lemmatized word level (*n*=1,2,3) and the character level (n=2,3,4); (4) weighted word overlap feature (Šarić et al., 2012) that takes the importance of words into consideration, where Web 1T 5-gram Corpus[2] was used to estimate the importance of words. (5) sentences were represented as vectors in *tf*idf* schema based on their lemmatized forms and then these vectors were used to calculate `cosine`, `Manhattan`, `Euclidean` distance and `Pearson`, `Spearmanr`, `Kendalltau` correlation coefficients based on different perspectives. Totally, we got thirty-one string based features.

2.3 Corpus Based Features

Corpus based features aim to capture the semantic similarities using distributional meanings of words and *Latent Semantic Analysis* (LSA) (Landauer and Dumais, 1997) is widely used to estimate the distributional vectors of words. Hence, we adopted two distributional sets released in TakeLab (Šarić et al., 2012), where LSA is performed over the New York Times Annotated Corpus (NYT)[3] and Wikipedia. Then two strategies were used to convert the distributional meanings of words to sentence level: (i) simply summing up the distributional vectors of words in the sentence, (ii) using the information content (Šarić et al., 2012) to weigh the LSA vector of each word w and summing them up. At last we used cosine similarity to measure the similarity of two sentences based on these vectors. Besides, we used the Co-occurrence Retrieval Model (CRM) (Weeds,

[2]https://catalog.ldc.upenn.edu/LDC2006T13
[3]https://catalog.ldc.upenn.edu/LDC2008T19

2003) as another type of corpus based feature. The CRM was calculated based on a notion of substitutability, that is, the more appropriate it was to substitute word w_1 in place of word w_2 in a suitable natural language task, the more semantically similar they were.

Besides, the extraction of aforementioned features rely on large external corpora, while (Guo and Diab, 2012) proposed a novel latent model, i.e., weighted textual matrix factorization (WTMF), to capture the contextual meanings of words in sentences based on internal term-sentence matrix. WTMF factorizes the original term-sentence matrix X into two matrices such that $X_{i,j} \approx P^T_{*,i} Q_{*,j}$, where $P_{*,i}$ is a latent semantics vector profile for word w_i and $Q_{*,j}$ is the vector profile that represents the sentence s_j. The weight matrix W is introduced in the optimization process in order to model the missing words at the right level of emphasis. Then, we used `cosine`, `Manhattan`, `Euclidean` functions and `Pearson`, `Spearmanr`, `Kendalltau` correlation coefficients to calculate the similarities based on sentence representations. At last, we obtained twelve corpus based features.

2.4 Syntactic Features

We estimated the similarities of sentence pairs at syntactic level. Stanford CoreNLP toolkit (Manning and Surdeanu, 2014) was used to obtain POS tag sequences. Afterwards, we performed eight measure functions described in Section 2.2 over these sequences, which resulted in eight syntactic based features.

2.5 Other Features

We built a binary feature to indicate whether two sentences in a pair have the same polarity (*affirmative* or *negative*) by looking up a manually-collected negation list with 29 negation words (e.g., *scarcely*, *no*, *little*). Also, we checked whether one sentence entails the other only using the named entity information which was provided in the dataset. Finally, we obtained nineteen other features.

2.6 Word Embedding Features

Recently, deep learning has achieved a great success in the fields of computer vision, automatic speech recognition and natural language processing. As a consequence of its application in NLP, word embeddings have been building blocks in many tasks, e.g., named entity recognition and chunking (Turian et al., 2010), semantic word similarities (Mikolov et al., 2013a), etc. Being distributed representation of words, word embeddings usually are learned using neural networks over a large raw corpus and has outperformed LSA for preserving linear regularities among words (Mikolov et al., 2013a). Due to its superior performance, we adopted word embeddings to estimate the similarities of sentence pairs. In our experiments, we used seven different word embeddings with different dimensions: *word2vec* (Mikolov et al., 2013b), *Collobert and Weston* embeddings (Collobert and Weston, 2008) and *HLBL* embeddings (Mnih and Hinton, 2007). Word2vec embeddings are distributed within the word2vec toolkit[4] and they are 300-dimensional vectors learned from Google News Corpus which consists of over a 100 billion words. *Collobert and Weston* and *HLBL* embeddings are learned over a part of RCV1 corpus which consists of 63 millions words, with 25, 50, 100, or 200 dimensions and 50, 100 dimensions over 5-gram windows respectively. To obtain sentence representations, we simply summed up embedding vectors corresponding to the non-stopwords tokens in bag of words (BOW) of sentences. After that, we used `cosine`, `Manhattan`, `Euclidean` functions and `Pearson`, `Spearmanr`, `Kendalltau` correlation coefficients to calculate the similarities based on these synthetic sentence representations. We got ninety word embedding features.

3 Experiments and Results

3.1 System Setups

The organizers provided 13,063 training pairs together with 4,727 development pairs in development phase and 972 test pairs in test phase. We removed the debatable instances (i.e., two annotators vote for *yes* and the other three for *no*) existing in the dataset, which resulted in 11,530 training pairs and 4,142 development pairs. We built two supervised classification systems over these datasets. One is `mlfeats` which only uses the traditional linguistic features

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

Algorithm	mlfeats			nnfeats		
	Precision	*Recall*	*F1*	*Precision*	*Recall*	*F1*
SVC(0.1)	0.756	0.942	0.839	0.756	0.942	0.839
GB(140)	0.756	0.939	0.838	0.754	0.940	0.837
GB(150)	0.755	0.939	0.837	0.753	0.939	0.836
RF(45)	0.754	0.937	0.835	0.749	0.936	0.832

Table 1: Top results of different classification algorithms in systems `mlfeats` and `nnfeats` on development dataset together with parameter values in brackets.

System	F1-Rank	*Precision*	*Recall*	*F1*
ECNU_`nnfeats`	4	0.767	0.583	0.662
ECNU_`mlfeats`	10	0.754	0.560	0.643
BASELINE_`logistic`	21	0.679	0.520	0.589
BASELINE_`WTMF`	28	0.450	0.663	0.536
BASELINE_`random`	38	0.192	0.434	0.266
ASOBEK_`svckernel`	1	0.680	0.669	0.674
ASOBEK_`linearsvm`	2	0.682	0.663	0.672
MITRE_`ikr`	3	0.569	0.806	0.667

Table 2: Performance and rankings of systems `mlfeats`, `nnfeats` and baseline systems on test dataset officially released by the organizers, as well as top ranking systems.

(i.e., features described in Section 2.2-2.5, 64 features in total) and the other is `nnfeats` which combines the traditional linguistic features with the word embedding features (148 features in total). Several classification algorithms were explored on development dataset including Support Vector Classification (SVC, *linear*), Random Forest (RF), Gradient Boosting (GB) implemented in the scikit-learn toolkit (Pedregosa et al., 2011) and a large scale of parameter values in these algorithms were tuned, i.e., the trade-off parameter c in SVR, the number of trees n in RF, the number of boosting stages n in GB. F-score was used to evaluate the performance of systems.

3.2 Results and Discussion

Table 1 presents the best four F1 results achieved by different algorithms together with their parameters in system `mlfeats` and `nnfeats` on development dataset. The results show that these two systems consistently yield comparable performance, which means that our proposed features based on word embeddings have little help to detect paraphrase on development set. And we also find that SVC performs slightly better than GB and RF algorithm. Therefore, we adopted a major voting schema based on SVC (*c*=0.1) and GB (*n*=140,150) in test period.

Table 2 summarizes the performance and ranks of our systems on test dataset, along with the baseline systems provided by the organizers and the top three systems. From this table, we observe following findings. Firstly, `nnfeats` using word embedding features outperforms the system `mlfeats` only using traditional linguistic features by 1.9%, which is inconsistent with the findings on development set. The possible reason may be that test data is collected from a different time period while train and development data is from the same time period while the word embedding features might more or less capture this differences. Secondly, our results are significantly better than the three baseline systems since our systems incorporate the features used in baseline systems and other effective features. Thirdly, the top 1 system (i.e., ASOBEK_`svckernel`) yields 3.1% and 1.2% improvement over our system `mlfeats` and `nnfeats` respectively, which indicates that word embedding features and traditional linguistic features are effective in resolving Twitter paraphrase problem.

To explore the influence of different feature types, we conducted feature ablation experiments where we removed one feature group from all feature set every time and then executed the same classification procedure. Table 3 shows the results of feature ablation experiments. From this table, we can see that the most influential features for recognizing tweet paraphrase is corpus based features and the second most important feature group is word embedding features, which are within our expectation since these two kinds of feature take advantage of the semantic meaning of words.

Feature	*Precision*	*Recall*	*F1*
All	0.767	0.583	0.662
-string	0.717	0.594	0.650 (-0.012)
-corpus	0.772	0.543	0.638 (-0.024)
-syntactic	0.797	0.560	0.658 (-0.004)
-other	0.784	0.560	0.653 (-0.009)
-embedding	0.823	0.531	0.646 (-0.016)

Table 3: The results of feature ablation experiments.

4 Conclusion

In this paper we address paraphrase in Twitter task by building a supervised classification model. Many linguistic features used in traditional paraphrase task and newly proposed features based on word embeddings were extracted. The results on test dataset demonstrate that (1) our proposed word embedding features improve the performance by a value of 1.9%; (2) the linguistic features used in paraphrase on conventional texts task are also useful and effective in Twitter domain.

Acknowledgements

This research is supported by grants from Science and Technology Commission of Shanghai Municipality under research grant no. (14DZ2260800 and 15ZR1410700) and Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213).

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, pages 135–187.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *NAACL, Companion Volume: Short Papers*, pages 33–36.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *ACL*, pages 864–872.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we Twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65.

Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, page 211.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Christopher D. Manning and Mihai et al. Surdeanu. 2014. The Stanford CoreNLP natural language processing toolkit. In *52nd ACL : System Demonstrations*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval*, pages 1–8.

Donald Metzler, Susan Dumais, and Christopher Meek. 2007. *Similarity measures for short segments of text*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *ACL*, pages 791–799.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. pages 172–180.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *the 48th ACL*, pages 384–394.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for measuring semantic text similarity. In **SEM 2012 and (SemEval 2012)*, pages 441–448, Montréal, Canada.

Julie Elizabeth Weeds. 2003. *Measures and applications of lexical distributional similarity*. Ph.D. thesis, University of Sussex.

Wei Xu, Chris Callison-Burch, and William B. Dolan. 2015. SemEval-2015 Task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*, Denver, CO.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Kostas Tsioutsiouliklis. 2011. Linguistic redundancy in Twitter. In *EMNLP*, pages 659–669.