Machine Learning Method for Paraphrase Identification

Oleksandr Marchenko[1(✉)], Anatoly Anisimov[1], Andrii Nykonenko[2], Tetiana Rossada[1], and Egor Melnikov[3]

[1] Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
omarchenko@univ.kiev.ua
[2] International Research and Training Center for IT and Systems, Kyiv, Ukraine
[3] PHASE ONE: KARMA LTD., London, UK

Abstract. A new effective algorithm and a system for paraphrase identification have been developed using a machine learning approach. The system architecture has the form of a multilayer classifier. According to their strategies, sub-classifiers of the lower level make decisions about the presence of paraphrase in sentences, while a super-classifier of the upper level makes the final decision. Conducted experiments demonstrated that the system has the accuracy of the paraphrase detection comparable with the best known analogous systems while being superior to all of them in implementation.

Keywords: Machine learning · SVM · Paraphrase identification

1 Introduction

The paraphrase identification task is one of the primary objectives in computational linguistics. This is due to the close relativeness of the paraphrase detection to the difficult problem of semantic analysis of natural language texts. To date, the development of algorithms for paraphrase detection has attained many significant achievements. Such algorithms use mainly a machine learning approach. As a rule, systems that demonstrate the most accurate results on standard paraphrase corpora use powerful and resource-consuming techniques such as Recursive Neural Networks, Convolutional Neural Networks, and Non-negative Matrix Factorization. In addition to non-triviality and ambiguity in the applications of neural networks, we should also emphasize the algorithmic complexity of Non-negative Matrix Factorization. These methods are too complex to be applied in practical systems operating in real time. This is the main obstacle limiting their practical use.

The authors of this paper set the goal to develop a full-fledged system operating online in real time that should recognize paraphrase and be capable of processing huge flows of information. Therefore, we choose the method of Support Vector Machine as the basic approach along with the development of the original multilevel classification system. The key idea behind this approach is the

H. Christiansen et al. (Eds.): FQAS 2017, LNAI 10333, pp. 164–173, 2017.
DOI: 10.1007/978-3-319-59692-1_14

development of a set of lower level sub-classifiers that are trained to accurately identify certain types of paraphrases. The super-classifier at the upper level analyzes the data provided by the sub-classifiers and decides about the presence of paraphrase.

The system has been developed and tested using the standard Microsoft Research Paraphrase Corpus (MSRP), and it demonstrates the accuracy of paraphrase identification that can be compared to the best up-to-date systems.

2 Related Work

Most of the previous works dedicated to paraphrase identification that use methods of machine learning aimed at creating an optimal set of features also known as effective feature space.

There were determined several types of features including:

- features based on strings and words including n-gram overlap of words and symbols [13], and features based on the assessment of machine translation quality [10];
- features based on the knowledge that use external lexical resources such as WordNet [5];
- features based on evaluating differences of syntax correlations between two sentences [2];
- features being evaluated in corpora based on the distribution models of similarity and Latent Semantic Analysis [6,7].

In modern studies, researchers refrain from the hand-crafted selection of features for distribution modeling and neural network solutions. Hua He, Kevin Gimpel and Jimmy Lin [8] implemented convoluted neural network for calculating multi-perspective similarity of sentences. Their system demonstrates precision at the state-of-the-art level (accuracy = 78.6%, F1 = 84.73% on the MSRP corpus). Cheng and Kartsaklis [1] implemented distributional models together with recursive neural network for syntax-aware multi-sense word embeddings for deep compositional models of meaning, thus completing the task with much better results than the ones demonstrated previously (accuracy = 78.6%, F1 = 85.3% on MSRP). At present, the best results are demonstrated by Ji and Eisenstein [9] who use Non-negative Matrix Factorization and Kullback-Leibler divergence to optimize features space (accuracy = 80.4%, F1 = 85.9% on MSRP).

Nitin Madnani, Joel Tetreault and Martin Chodorow [10] developed the algorithm that consists of 8 machine translation quality metrics for calculating the similarity of sentences and one upper level classifier that makes the final decision based on the assessments of the lower level metrics. Despite the absence of powerful and resource-consuming computations, this algorithm demonstrates the state-of-the-art level results (accuracy = 77.4%, F1 = 84.1% on MSRP), significantly exceeding in simplicity of implementation all the aforementioned methods.

The algorithm presented in this article could be considered as belonging to that class of methods for paraphrase identification.

3 Description of the Method

To detect paraphrases, a two-level classifier was built (Fig. 1). On the lower level, the input data are pairs of sentences. The task is to check whether these sentences constitute paraphrases of each other. It is achieved by selection of prime classifiers, each of them is trained to detect paraphrases of a certain type. These classifiers determine the presence or absence of paraphrase for each incoming pair. On the upper level, the received results are assessed by the main classifier that makes the final decision.

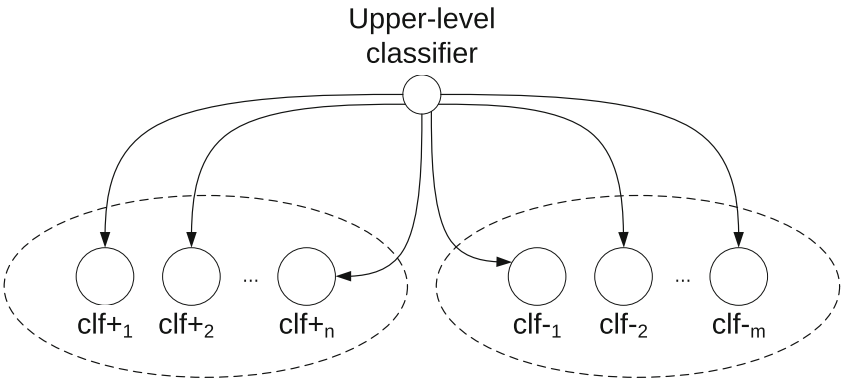

Fig. 1. Architecture of the two-level classifier

For training, a set of tagged pairs of sentences is required for the system (tag 1 – paraphrase/0 – absence of paraphrase).

We denote the first and the second sentences as r and c respectively, $|x|$ - is a count of tokens in a sentence x.

Features used for training are as follows:

1. Sentence Length Difference – comparison of lexeme quantities in sentences.

$$SLD(r,c) = \frac{|r| - |c|}{|r|} \tag{1}$$

$$SLD^*(r,c) = \frac{1}{d^{|r|-|c|}} \tag{2}$$

where d - is a constant number (we took $d = 0.8$).

2. N-Grams Comparing – comparison of unigrams, bigrams, and trigrams

$$NGC_N(r,c) = \frac{|NGrams_N(r) \bigcap NGrams_N(c)|}{|NGrams_N(r)|} \tag{3}$$

where $NGrams_N(x)$ – the set of word sequences with length N in a sentence x.

3. Dependencies Similarity – similarity of syntax dependencies

$$DS(r,c) = \frac{\sum\limits_{d \in DT_r} \sum\limits_{i \in r_{dep=d}} \max\limits_{j \in c_{dep=d}} sim(i,j) \cdot BP(|r_{dep=d}|, |c_{dep=d}|)}{|DT_r| \cdot BP(|DT_r|, |DT_c|)} \quad (4)$$

where DT_x – set of all existing syntax dependencies in a sentence x; $x_{dep=d}$ – all lexemes of a sentence x connected with dependency d; $sim(x,y)$ – indicator of similarity for lexemes x and y calculated on the basis of WordNet; $BP(x,y)$ – Brevity Penalty:

$$BP(x,y) = \begin{cases} 1, & if\ y > x \\ e^{1-\frac{x}{y}}, & if\ y \leq x \end{cases} \quad (5)$$

4. Dependencies Comparing – comparison of syntax dependencies

$$DC(r,c) = \frac{|dependencies(r) \bigcap dependencies(c)|}{|dependencies(r)|} \quad (6)$$

where $dependencies(x) = \{(i,j,d) : i$ and j are connected with the relation d in a sentence $x\ \}$

5. Syntactic N-Grams Comparing – comparison of syntax unigrams, bigrams, and trigrams. The calculation is done the same way as for usual N-grams, though here syntax N-grams are sequences of lexemes, which are subgraphs of sentence dependency trees.

Measures of semantic proximity developed in the models of machine translation are also used:

6. BLEU [11]

$$BLEU(r,c) = BP(r,c) \times exp\left[\sum_{n=1}^{L} \frac{1}{L} \times log(p_n(r,c))\right] \quad (7)$$

where L is the maximum n-gram size. The n-gram precision p_n is given as follows:

$$p_n(r,c) = \frac{\sum_{x \in NGrams_n(c)} count(x, NGrams_n(r) \cap NGrams_n(c))}{\sum_{x \in NGrams_n(c)} count(x, NGrams_n(c))} \quad (8)$$

where $count(x,X)$ is the count of an element x in a set X.

7. BLEU, where sequences of meaningful words are used for N-grams, that is:

$$IDF(x, docs) > K \quad (9)$$

where IDF is the abbreviation for the Inverse Document Frequency:

$$IDF(x, docs) = log(\frac{|docs|}{|docs|_{x \in docs|}}); \quad (10)$$

$docs$ – corpus of documents; K – a certain threshold depending on the peculiarities of a document corpus.

8. BLEU, where sequences of syntax N-grams are used.
9. NIST [4]

$$NIST(r,c) = \sum_{n=1}^{N} BP(r,c) \times \frac{\sum_{x \in NGrams_n(c)} Info(x)}{count(x, NGrams_n(c))} \tag{11}$$

where $Info(x)$ is the information weight, we use IDF metric to compute (11): $Info(x) = IDF(x, docs)$.
10. METEOR calculates a lexical similarity score based on a word-to-word alignment between two strings [3]. Words are matched if and only if their surface forms are identical; words are matched if they are both members of a synonym set according to the WordNet database.
11. BADGER is a method based on information theory and data compression. It implements efficient Normalized Compression Distance (NCD) utilizing the Burrows Wheeler Transformation (BWT) [12].

Each measure $M(x, y)$ has two variants of implementation: *precision* $M(x, y)$ is a function as is, and *recall* $M(y, x)$ – the same function is used but the arguments-sentences are given in inverse order. All measures using comparison of lexemes have two variants of implementation: total lexeme coincidence and lexemes synonyms coincidence.

To train the classifiers of the lower level, it is necessary to create training sets for each classifier (Fig. 2). Each of them should be able to determine paraphrases of a certain set of types. Besides, each type of paraphrases could be contained in training sets for several classifiers. Due to this, the coverage of all types of paraphrases and mutual insurance of classifiers are guaranteed while solving a task. Thus, questions arise – what a paraphrase type is, how to model it, and how to determine the type of paraphrases for each pair of sentences. As a working hypothesis the following assumption is used: two pairs of sentences belong to the set of paraphrases of the same type, if after processing these two pairs of sentences they have a certain subset of features with close values.

During the *first stage* of the algorithm training, for set of pairs of sentences from the training sub-corpus of the Microsoft Research Paraphrase Corpus (MSRP) a training matrix is being calculated for values of features f_i.

During the *second stage*, with the use of clustering algorithms the training matrix is partitioned into the set of matrices: $training_1^+$, $training_2^+$, ..., $training_N^+$ and $training_1^-$, $training_2^-$,..., $training_M^-$:

$$training^+ = \bigcup_{i=1}^{N} training_i^+, training^- = \bigcup_{i=1}^{M} training_i^- \tag{12}$$

$$training^+ \cup training^- = training \tag{13}$$

where $training^+$ and $training^-$ are the sets of vectors formed by feature values for all pairs of sentences tagged as paraphrase or non-paraphrase correspondingly.

To make classifiers be able to "support" each other when making decisions in each particular case, a training set should consist of several different types of paraphrases so that the sets can intensively overlap. For this, the following condition should be met for $training^+$ and $training^-$:

$$\forall i \exists j : training_i^+ \cap training_j^+ \neq \varnothing \tag{14}$$

$$\forall i \exists j : training_i^- \cap training_j^- \neq \varnothing. \tag{15}$$

The *third stage*. Based on the matrices received at the second stage, training sets are formed by creating $n = \binom{N}{k}$ combinations with sets of $\{training_i^+\}$ and adding to each combination the full $training^-$ set, where k is a number of merged sets. All the corresponding pairs of sentences with vectors of feature values being included into a certain obtained combination form a training set for a classifier of lower level clf_i^+. Thus, we obtain training sets for $n = \binom{N}{k}$ classifiers clf_i^+. Training sets for $m = \binom{M}{k}$ classifiers clf_i^- are formed the same way.

The *fourth stage*. Training the set of classifiers clf_1, clf_2, ..., clf_{n+m} is performed on the basis of the obtained sets. After the training, all classifiers process the whole MSRP training set. Matrix of classifiers decisions for all pairs of sentences that belong to the training corpus together with "paraphrase"/"non-paraphrase" tags serve as the training set for the super-classifier.

During the *fifth stage*, the training of the upper-level classifier is performed.

To build the classifiers for the upper and the lower levels, the method of Support Vector Machine is used.

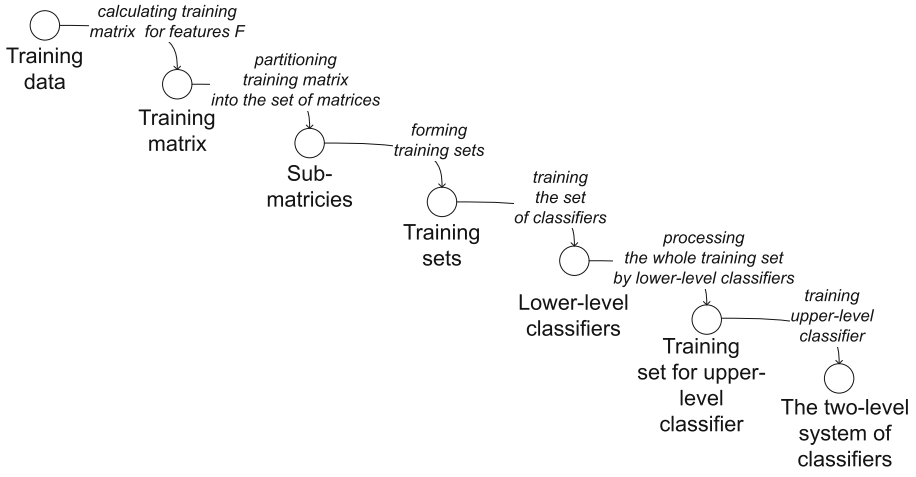

Fig. 2. Steps of algorithm for building system of classifiers

4 Algorithm for Building System of Classifiers

Step 1. Automated partitioning of $training^+$ into classes according to the types of paraphrases.

Pairs of paraphrase that belong to the same type should correlate according to the values of calculated features. Thus, the values should be *close*. Some of the features can be irrelevant for the paraphrase of a certain type. That is why it is necessary to determine C^+ set of such typical sub-sets of paraphrase $\{c\}$ that include any two elements with the similar set of values for a certain set of features. The following conditions should be met:

$$\bigcup_{c \subseteq C^+} c = training^+ \tag{16}$$

$$\forall c_1 \subseteq C^+ \; \exists c_2 \subseteq C^+ : c_1 \cap c_2 \neq \varnothing \tag{17}$$

$$\forall c \subseteq C^+ \; \forall x, y \in c : x \simeq^l y \tag{18}$$

where:

- C^+ is the set of the received typical sub-sets;
- $x \simeq^l y \Leftrightarrow \exists X \subseteq F : \|X\| = l \& \forall f \in X : |f(x) - f(y)| < \varepsilon$ for selected l and ε;
- F is the set of the implemented features;
- l is a number of features having close values for pairs of sentences that belong to one type of paraphrases; ε is the maximum value of differences for such features. Optimal values for l and ε were selected during conducted experiments.

To solve the problem with $training^+$ set of the vectors of the feature values for the pairs of sentences tagged as paraphrase, N centroids are calculated: C_1^+, C_2^+,..., C_N^+ – the most distant from each other elements in $training^+$. The following conditions should be met for the centroids:

$$C_1^+, C_2^+, ..., C_N^+ \in training^+ \tag{19}$$

$$\sum_{i,j \in 1..N \& i \neq j} dist(C_i^+, C_j^+) \rightarrow max \tag{20}$$

where *dist* is the Euclidean distance between two elements. We select such centroids C_i^+ that the sum of distances among them is maximum possible.

After the selection of centroids, each element from the $training^+$ set is added to one or several clusters, which are determined by C_1^+,C_2^+,..., C_N^+ centroids. Condition that element x gets to the cluster c_i^+ determined by the centroid C_i^+ if the $x \simeq^l C_i^+$ holds. This way, N clusters are built: c_1^+, c_2^+, ..., c_N^+.

For the elements that haven't been selected to any class, Step 1 is repeated recursively, though with modified N, l and ε parameters. As a result, N' clusters are built, each of them consists of pairs of paraphrase sentences taken from the training set, which together form the $training^+$ set. Besides, they should not have an empty intersection. The intensity of intersections is determined by parameters N, l and ε.

Step 2. From c_1^+, c_2^+, .., $c_{N'}^+$ we form $n = \binom{N'}{k}$ of all possible combinations of k clusters. To each of combinations, a $training^-$ set is added. After selecting the corresponding pairs of sentences from the **training** corpus, the training sets T_1^+, T_2^+, ..., T_n^+ are received.

Step 3. Using standard methods from library http://scikit-learn.org/stable/modules/feature_selection.html for each training set T_1^+, T_2^+, ..., T_n^+ for SVM method, the optimal set of features $\{f_1, f_2, ..., f_k\}$ is formed using the initial set of the implemented features F.

Step 4. Classifiers of the lower level clf_1^+, clf_2^+, ..., clf_n^+ train based on T_1^+, T_2^+, ...,T_n^+ using the corresponding optimal sets of features.

Step 5. The same way the classifiers clf_1^-, clf_2^-, ..., clf_m^- are generated. Together with clf_1^+, clf_2^+, ..., clf_n^+ they constitute the classifiers of the lower level clf_1, clf_2, ..., clf_{n+m}.

Step 6. When the training is completed, all the classifiers process the whole training set. The matrix of solutions made by classifiers clf_1, clf_2, ..., clf_{n+m} for all pairs of sentences from the training corpus together with the pairs tags serve as a training set for super-classifier of the upper level. The training of the classifier of the upper level is performed.

Step 7. The trained system processes the test corpus of the pairs of sentences.

5 Results of Experiments

Training and testing were performed using the Microsoft Research Paraphrase Corpus set. The corpus consists of 5,800 sentences taken from different sources with tags signaling whether a certain pair constitutes paraphrase or not. Besides, the corpus is divided into the training set with 4,076 pairs of sentences (2,753 positive: 67,5%) and a the testing set with 1,725 pairs of sentences (1,147 positive: 66,5%).

Tables 1 and 2 demonstrate the results of experiments on the Microsoft Research Paraphrase Corpus set.

As seen, the chosen method has demonstrated the results that can be compared with the best existing algorithms (Table 1, www.aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art)) with–out additional implementation of powerful methods such as neural networks, Latent Semantic Analysis, and Non-negative Matrix Factorization. To some extent, the

Table 1. Comparisons of our system results (the last line) with the existing state-of-the-art systems

Algorithm	Description	Accuracy	F1
MTMETRICS [10]	Combination of eight machine translation metrics	77.4%	84.1%
Multi-Perspective CNN [8]	Multi-perspective Convolutional NNs and structured similarity layer	78.6%	84.7%
SAMS-RecNN [1]	Recursive NNs using syntax-aware multi-sense word embeddings	78.6%	85.3%
TF-KLD [9]	Matrix factorization with supervised reweighting	80.4%	85.9%
Two-level classifier	Multilayer classifier	77.86%	85.16%

Table 2. Experiment results

Algorithm	Precision	Recall	Accuracy	F1
Single classifier based on SVM	76.29%	89.19%	74.38%	82.23%
Two-level classifier ($N = 5$, $l = 7$, $\varepsilon = 1.3e - 3$, $k = 3$)	75.02%	95.03%	75.65%	83.84%
Two-level classifier ($N = 5$, $l = 8$, $\varepsilon = 1.3e - 3$, $k = 2$)	76.76%	95.64%	77.86%	85.16%

developed method could be considered as a generalization of the algorithm [10]. From this point of view our algorithm gives better precision results compared to its predecessor. The main improvement of implementation has been achieved by optimization of training sets for lower level classifiers in such a way that each of them should be able to determine paraphrases of a certain set of types and each type of paraphrases could be contained in training sets for several classifiers. Hence, the coverage of all types of paraphrases and mutual insurance of classifiers are guaranteed.

The proposed two-level system demonstrates accuracy of paraphrase identification and evaluation of F1 almost as good as the best existing state-of-the-art systems. At the same time it has advantages that are the ease of implementation and less required computing resources.

To evaluate the efficiency of the developed two-level architecture we have built a single system of paraphrase identification based on SVM algorithm implemented with the use of the abovementioned features 1–11. Standard methods of http://scikit-learn.org/stable/modules/feature_selection.html library have been used to optimize the set of features. As we can see from the results of Table 2, the two-level system developed in both implemented configurations (the second and the third lines) overcomes the single system based on the SVM method (the first line).

6 Conclusions

The research paper describes a new effective algorithm for the paraphrase identification task using machine learning. The experiments demonstrated high accuracy in paraphrase identification that can be compared to the results achieved by the existing state-of-the-art systems while being superior to all of them in implementation.

Acknowledgments. The authors of the article are grateful to PHASE ONE: KARMA LTD. company, especially to the Unplag team for the support in research and considerable assistance in the development, testing and implementation of the paraphrase identification method.

References

1. Cheng, J., Kartsaklis, D.: Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In: Proceedings of EMNLP 2015, pp. 1531–1542 (2015)
2. Das, D., Smith, N.A.: Paraphrase identification as probabilistic quasi-synchronous recognition. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics, pp. 468–476 (2009)
3. Denkowski, M., Lavie, A.: Extending the meteor machine translation metric to the phrase level. In: Proceedings of NAACL (2010)
4. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: Proceedings of HLT, pp. 138–145 (2002)
5. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
6. Guo, W., Diab, M.: Modeling sentences in the latent space. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 864–872 (2012)
7. Hassan, S.: Measuring semantic relatedness using salient encyclopedic concepts. Ph.D. thesis. University of North Texas (2011)
8. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Proceedings of EMNLP 2015, pp. 1576–1586 (2015)
9. Ji, Y., Eisenstein, J.: Discriminative improvements to distributional sentence similarity. In: Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2013), pp. 891–896 (2013)
10. Madnani, N., Tetreault, J., Chodorow, M.: 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, pp. 182–190 (2012)
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of ACL (2002)
12. Parker, S.: BADGER: a new machine translation metric. In: Proceedings of the Workshop on Metrics for Machine Translation at AMTA (2008)
13. Wan, S., Dras, M., Dale, R., Paris, C.: Using dependency-based features to take the "para-farce" out of paraphrase. In: Australasian Language Technology, Workshop, pp. 131–138 (2006)