# Paraphrase Detection Based on Identical Phrase and Similar Word Matching

**Hoang-Quoc Nguyen-Son[1], Yusuke Miyao[2], and Isao Echizen[2]**
[1]University of Science, VNU-HCM, Hochiminh, Vietnam
nshquoc@fit.hcmus.edu.vn
[2]National Institute of Informatics, Tokyo, Japan
{yusuke,iechizen}@nii.ac.jp

## Abstract

Paraphrase detection has numerous important applications in natural language processing (such as clustering, summarizing, and detecting plagiarism). One approach to detecting paraphrases is to use predicate argument tuples. Although this approach achieves high paraphrase recall, its accuracy is generally low. Other approaches focus on matching similar words, but word meaning is often contextual (e.g., 'get along with,' 'look forward to'). An effective approach to detecting plagiarism would take into account the fact that plagiarists frequently cut and paste whole phrases and/or replace several words with similar words. This generally results in the paraphrased text containing identical phrases and similar words. Moreover, plagiarists usually insert and/or remove various minor words (prepositions, conjunctions, etc.) to both improve the naturalness and disguise the paraphrasing. We have developed a similarity matching ($SimMat$) metric for detecting paraphrases that is based on matching identical phrases and similar words and quantifying the minor words. The metric achieved the highest paraphrase detection accuracy (77.6%) when it was combined with eight standard machine translation metrics. This accuracy is better than the 77.4% rate achieved with the state-of-the-art approach for paraphrase detection.

## 1 Introduction

Paraphrase detection is used to determine whether two texts (phrases, sentences, paragraphs, documents, etc.) of arbitrary lengths have the same meaning. Such detection is widely used to remove the tremendous amount of duplicate information on the Internet. It is also used to handle the overlap of semantic components in texts. Such components are used in various natural language applications such as word sense discrimination, summarization, automatic thesaurus extraction, question-and-answer generation, machine translation, and plagiarist or analogical relation identification.

Some researchers in the field of paraphrase detection have used vector-based similarity to identify the differences between two sentences (Mihalcea et al., 2006; Blacoe and Lapata, 2012). The two sentences are represented by two vectors based on the frequency of their words in text corpora. The vectors are compared to estimate sentence similarity. Plagiarists attempt to thwart this comparison by modifying the copied sentence by inserting or removing a few minor words, replacing words with similar words that have different usage frequencies, etc. Such modification reduces the effectiveness of vector-based similarity analysis.

Other researchers have analyzed the difference in meaning between two sentences on the basis of their syntactic parsing trees (Socher et al., 2011; Qiu et al., 2006; Das and Smith, 2009). The structure of the trees is a major factor used various sophisticated algorithms such as recursive autoencoders (Socher et al., 2011), heuristic similarity (Qiu et al., 2006), and probabilistic inference (Das and Smith, 2009). However, these algorithms are affected by manipulation (deleting, inserting, reordering, etc.) of the words in the sentences. Such manipulations can significantly change the structures of the parsing trees.



*29th Pacific Asia Conference on Language, Information and Computation pages 504 - 512*
*Shanghai, China, October 30 - November 1, 2015*

Other researchers (Mihalcea et al., 2006; Chan and Ng, 2008) have used matching algorithms to determine the similarity of two sentences. Mihalcea et al. (2006), for example, proposed a method for finding the best matching of a word in a sentence with the nearest word in the other sentence. However, word meaning is often contextual (e.g., 'make sure of,' 'take care of').

Machine translation (MT) metrics, which are generally used to evaluate the quality of translated text, can also be used to judge two texts in the same language. Due to the similarity of machine translation and paraphrase detection, many MT metrics have been applied to paraphrase detection (Finch et al., 2005; Madnani et al., 2012). For example, eight standard MT metrics have been combined to create a state-of-the-art paraphrase detection approach (Madnani et al., 2012). However, the objectives of machine translation and paraphrase detection differ: machine translation tries to effectively translate text from one language to another while paraphrase detection tries to identify paraphrased text. This difference affects the application of MT metrics to paraphrase detection.

A paraphrase is a restatement of the meaning of a text using other words. It is a specific type of plagiarisms. We identify common practices plagiarizers who try to paraphrase a text. The Microsoft Research Paraphrase (MSRP) corpus (Dolan et al., 2004) is commonly used to identify the common practices. An example paraphrase pair extracted from this corpus is shown in Figure 1.

Plagiarists frequently cut and paste several phrases of different lengths. This can result in a sentence pair containing *identical phrases*. The two sentences in Figure 1 have two identical phrases: "Intelligence officials" and "a week ago to expect a terrorist attack in Saudi Arabia." Plagiarists also add and delete *minor words* to improve the naturalness of the text. In the example pair, the preposition "in" (in bold) in the second sentence is considered a minor word.

Moreover, plagiarists can replace several words with *similar words* without changing the sentence meaning to avoid paraphrase detection. The words connected by dashed lines with arrows in the example are most likely such replacements. The remaining words are probably the combination of a few manipulations (reorganization, deletion, insertion, replacement, etc.). Such *modifications* are typically intended to ensure that the paraphrased sentence has the same meaning as the original sentence.

We make several contributions based on an analysis of related work and the common practices of plagiarists in this paper.

- We present a heuristic algorithm for finding an optimal matching of *identical phrases* with maximum lengths.
- We suggest removing the *minor words* from the words remaining in the sentences. These minor words include prepositions, subordinating conjunctions ('at,' 'in,' etc.), modal verbs, possessive pronouns ('its,' 'their,' etc.), and periods ('.').
- We present an algorithm for determining the perfect matching of *similar words* by using the matching algorithm proposed by Kuhn and Munkres (Kuhn, 1955; Munkres, 1957). The degree of similarity between two similar words is identified using WordNet (Pedersen et al., 2004). These similarities are used as weights for the matching algorithm.
- We present a related matching ($RelMat$) metric for quantifying the relationship between two sentences on the basis of matching identical phrases and similar words.
- We present a brevity penalty metric to reduce the effect of paraphrased sentence *modification*. This metric is combined with the $RelMat$ metric into a similarity matching $SimMat$ metric for effectively detecting paraphrases.

We used the MSRP corpus to evaluate the $SimMat$ metric. Our method using the $SimMat$ metric outperformed many previous methods. The $SimMat$ metric had the highest accuracy (77.6%) when used in combination with eight standard MT metrics (MAXSIM, SEPIA, TER, TERp, METEOR, BADGER, BLEU, and NIST). The accuracy was higher than with the state-of-the-art approach (accuracy=77.4%). The result shows that our method effectively uses the paraphrasing practices commonly used by plagiarists to detect them.

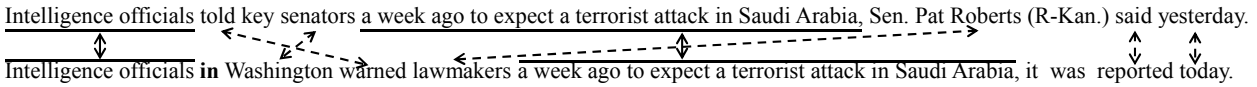


Figure 1: Example paraphrase pair taken from MSRP corpus.

## 2 Related work

### 2.1 Paraphrase detection

The baseline for paraphrase detection is based on vector-based similarity. Each source message and target message is represented as a vector using the frequencies of its words (such as term frequency (Mihalcea et al., 2006) and co-occurrence (Blacoe and Lapata, 2012)). The similarity of the two vectors is quantified using various measures (e.g., cosine (Mihalcea et al., 2006), addition and point-wise multiplication (Blacoe and Lapata, 2012)). The problem with vector-based methods is to focus on the frequency of separate words or phrases. However, plagiarists can paraphrase by replacing words with similar words that have a very different frequency. Moreover, they can delete and/or insert minor words that do not change the meaning of the original sentences. Such manipulations change the quality of the representation vector, which reduces paraphrase detection performance.

Several methods have been proposed for overcoming the manipulation problem that use syntactic parsing trees of messages. The replacement of similar words and the use of minor words do not change the basic structure of the trees. Qiu et al. (2006) reported a method that detects the similarity of two sentences by heuristically comparing their predicate argument tuples, which are a type of syntactic parsing tree. The high paraphrase recall (93%) it attained shows that most paraphrases have the same predicate argument tuples. However, the accuracy was very low (72%). Parsing trees were used for probabilistic inference of paraphrases by Das and Smith (2009).

Another method considers these trees as input for a paraphrase detection system based on recursive autoencoders (Socher et al., 2011). The drawback of the parsing tree approach is that parsing trees are affected by the reordering words in a sentence such as the conversion of a sentence from passive voice to active voice. Another method finds the maximum matching for each word in two sentences (Mihalcea et al., 2006). The similarity of matching two words is based on WordNet. However, the weakness of this method is that a word in a first sentence is probably matched to more than one word in the second sentence. This means that a very short sentence can be detected as a paraphrase of a long sentence in some cases. Another problem with word matching is that the meaning of some words depends on the context. For example, the basic meaning of 'get' changes when used in the phrasal verb 'get along with.'

Commonly used techniques for detecting paraphrases are based on MT metrics. This is because the translation task is very similar to the paraphrase detection task for text in the same language. For example, Finch et al. (2005) extended a MT metric (PER) and combined it with three other standard metrics (BLEU, NIST, and WER) into a method for detecting paraphrases. Another method developed by Madnani et al. (2012) is based on the integration of eight metrics (TER, TERp, BADGER, SEPIA, BLEU, NIST, METEOR, and MAXSIM). However, the main purpose of these metrics is for translating, and their integration is unsuitable for detecting paraphrases. To overcome these weaknesses, we developed a similarity metric and combined it with eight standard metrics, as described below.

### 2.2 Standard MT metrics

Two basic MT metrics for measuring the similarity of two text segments are based on finding the minimum number of operators needed to change one segment so that it matches the other one. The translation edit rate (TER) metric (Snover et al., 2006) supports standard operators, including shift, substitution, deletion, and insertion. The TER-Plus (TERp) metric (Snover et al., 2009) supports even more operators, including stemming and synonymizing.

The BADGER MT metric (Parker, 2008) uses compression and information theory. It is used to calculate the compression distance of two text segments by using Burrows-Wheeler transformation.

This distance represents for probability that one segment is a paraphrase of the other.

The SEPIA MT metric (Habash and Elkholy, 2008) is based on the dependence tree and is used to calculate the similarity of two text segments. It extends the tree to obtain the surface span, which is used as the main component of the similarity score. After the components of the tree are matched, a brevity penalty factor is suggested for deciding the difference in tree lengths for the two text segments.

Two other MT metrics commonly used in machine translation are the bilingual evaluation understudy (BLEU) metric (Papineni et al., 2002) and the NIST metric (Doddington, 2002) (an extension of the BLEU metric). Both also quantify similarity on the basis of matching words in the original text segment with words in the translated segment. Whereas the BLEU metric simply calculates the number of matching words, the NIST metric takes into account the importance of matching with different levels. The main drawback of these word matching metrics is that a word in a segment can match more than one word in the other segment.

Two MT metrics based on non-duplicate matching have been devised to overcome this problem. The METEOR metric (Denkowski and Lavie, 2010) uses explicit ordering to identify matching tuples with minimized cross edges. However, it simply performs word-by-word matching. The maximum similarity (MAXSIM) metric (Chan and Ng, 2008) finds the maximum matching of unigram, bigram, and trigram words by using the Kuhn-Munkres algorithm. However, the maximum length of the phrase is a trigram. Moreover, the similarities of the phrases (unigram, bigram, and trigram) are disjointly combined. To overcome these drawbacks with the standard MT metrics, we have developed a heuristic method for finding the maximum of matching tuples up to the length of the text segments being compared. We also developed a metric for sophisticatedly quantifying the similarity on the basis of the matching tuples.

## 3 Similarity matching ($SimMat$) metric

Our proposed similarity metric ($SimMat$) for quantifying the similarity of input text comprises four steps, as illustrated in Figure 2. The following is a step-by-step description of our method using two sentences, which is an actual paraphrase pair from the MSRP corpus.

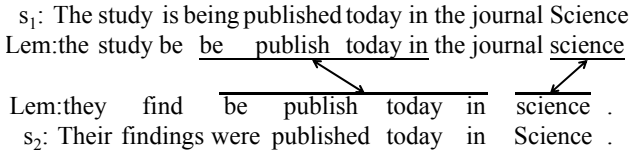


Figure 3: Matching identical phrases with their maximum lengths (Step 1).

$s_1$: "*The study is being published today in the journal Science*"

$s_2$: "*Their findings were published today in Science.*"

### 3.1 Match identical phrases (Step 1)

The individual words in the two input sentences are normalized using lemmas. The Natural Language Processing (NLP) library of Stanford University (Manning et al., 2014) is used to identify the lemmas. The lemmas for the two example sentences are shown in Figure 3.

The heuristic algorithm we developed for matching the lemmas in the two sentences repeatedly finds a new matching pair in each round. In each round, a new pair with the maximum phrase length is established. The pseudo code of the algorithm is illustrated in Algorithm 1. The stop condition is when there is no new matching pair. For example, two identical lemma of phrases, "be publish today in" and "science," are matched (as shown as Figure 3).

In algorithm 1, the function $getLemmas(s)$ extracts the lemmas of sentence $s$ using the NLP library. The function $len_L$ gets the number of elements in set $L$. The function $match(L_1[i], L_2[j])$ finds the maximum length matching of phrase $L_1$, which starts at the $i$-th position in the first sentence, and that of phrase $L_2$, which starts at the $j$-th position in the second sentence.

### 3.2 Remove minor words (Step 2)

The words remaining after phrase matching in Step 1 are used for removing minor words. First, the part of speech (POS) for each word is identified. The Stanford library tool (Manning et al., 2014) is used for this purpose. The POSs for the words in two the example sentences are shown in Figure 4.

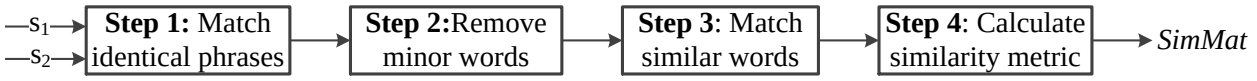


Figure 2: Four steps in calculation of similarity matching ($SimMat$) metric.

**Algorithm 1** Match identical phrases.

```
1:  function MATCHIDENTICALPHRASES(s1, s2)
2:      L1 ← getLemmas(s1);
3:      L2 ← getLemmas(s2);
4:      P ← ∅;
5:      repeat
6:          new ← ∅;
7:          for i = 0 to len_L1 − 1 do
8:              for j = 0 to len_L2 − 1 do
9:                  if {L1[i], L2[j]} ∉ P then
10:                     tmp ← match(L1[i],L2[j]);
11:                     if len_tmp > len_new then
12:                         new ← tmp;
13:                     end if
14:                 end if
15:             end for
16:         end for
17:         if new is not null then
18:             P = P ⋃ new;
19:         end if
20:     until (new = ∅);
21:     return P;
22: end function
```

Our analysis of the common practices of plagiarists showed that four types of minor words should be removed: prepositions and subordinating conjunctions (IN), modal verbs (MD), possessive pronouns (PRP$), and periods (“.”). These minor POSs generally do not change the meaning of the paraphrased text as they are often used to simply improve the naturalness of the paraphrased text. For example, the two minor POSs (PRP$ and “.”) were deleted from sentence $s_2$ in Figure 4. An example of preposition deletion is illustrated in Figure 1. Detection of remaining type of minor words (modal verbs) is illustrated for an actual paraphrase pair in Figure 5.

### 3.3 Match similar words (Step 3)

After minor word deletion in Step 2, the perfect matching of similar words is done using the algorithm we developed on the basis of the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957). The weights of each pair in the algorithm are calculated from the similarity of the two lemmas of the words using the $path$ metric (Pedersen et al., 2004). The $path(w_1, w_2)$ metric computes the shortest path ($pathLength$) between two words $w_1$ and $w_2$ in the ‘is-a’ hierarchies of WordNet, as shown in Eq. 1. The $pathLength$ is constrained to be a positive integer to ensure that $0 <= path <= 1$. For example, the $path$ metric for the “study” and “find” pair is 0.33. The perfect matching found for the two example sentences is shown in Figure 6. The word “study” in sentence $s_1$ is matched with a similar word, “findings,” in sentence $s_2$.

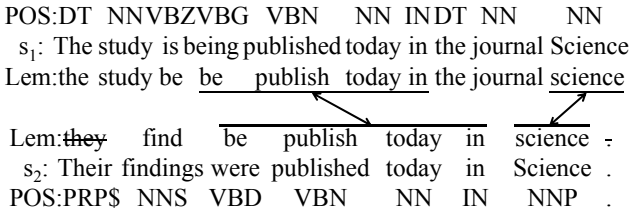


Figure 4: Remove minor words (Step 2).

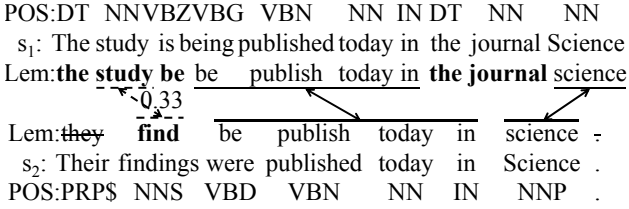


Figure 6: Find perfect matching of similar words using Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957) (Step 3).

$$path(w_1, w_2) = \frac{1}{pathLength(w_1, w_2)} \quad (1)$$

### 3.4 Calculate similarity metric (Step 4)

Finally, the $RelMat$ metric is calculated using the results of identical phrase matching in Step 1 and similar word matching in Step 3:

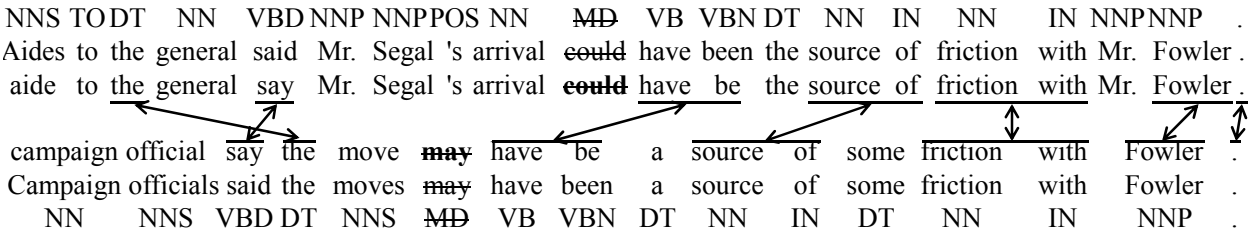


Figure 5: Example of removing minor words (modal verbs).

$$RelMat(s_1, s_2) = \\ = \frac{\#Np + \sum_{i=0}^{N-1} len(p_i)^\alpha + \sum_{j=0}^{M-1} path(w_j)^\alpha}{\#Np + \#Nw + \sum_{i=0}^{N-1} len(p_i)^\alpha + \sum_{j=0}^{M-1} 1^\alpha}, \quad (2)$$

where $\#Np$ is the total number of words in the matched identical phrases, $\#Nw$ is the number of matched similar words, $N$ and $M$ are the corresponding numbers of matched identical phrases and similar words, $p_i$ is the $i$-th matched phrase in Step 1, $len(p_i)$ is the number of words in the phrase $p_i$, and $path(w_j)$ is the $path$ metric of the $j$-th matched word in Step 3.

Eq. 2 ensures that $0 <= RelMat <= 1$. The $RelMat$ metric equals 1 only if the two sentences are identical. Using $\#Np$ only in the numerator means that the matching of identical phrases is more important than the matching of similar words. The $len(p_i)^\alpha$ and $path(w_j)^\alpha$ with $\alpha >= 0$ indicate the respective contributions of matched phrase $p_i$ and matched word $w_j$ to the $RelMat$ metric. The greater the value of $\alpha$, the greater the contribution of the identical phrases and the lesser the contribution of the similar words. Because $0 <= path(w_j) <= 1$, we use $1^\alpha$ to normalize the contributions of the matched words.

Threshold $\alpha$ is set to an optimal value of 0.2, as described in more detail in Section 5. The $RelMat$ metric for the two example sentences is calculated using

$$RelMat = \frac{5 + (4^{0.2} + 1^{0.2}) + 0.33^{0.2}}{5 + 1 + (4^{0.2} + 1^{0.2}) + 1^{0.2}} = 0.87.$$

The remaining words are probably modified by few manipulations (e.g., insertion, deletion). Such modification is typically intended to improve the naturalness of text. Therefore, the two sentences being compared frequently have different lengths. To reduce this effect, we developed a brevity penalty metric $p$ based on the METEOR metric (Denkowski and Lavie, 2010). It is calculated as shown in Eq. 3, where $\#ReW(s)$ is the number of words remaining in sentence $s$ after phrase matching and minor word removal. Penalty $p$ is combined with $RelMat$ into the similarity matching $SimMat$ metric, as shown in Eq. 4.

$$p(s_1, s_2) = \\ = 0.5 \times (\frac{|\#ReW(s_1) - \#ReW(s_2)|}{max(\#ReW(s_1), \#ReW(s_2))})^3 \quad (3)$$

$$SimMat = RelMat \times (1 - p) \quad (4)$$

Penalty metric $p$ and the $SimMat$ metric are respectively calculated for the example sentences using Eq. 5 and Eq. 6. To calculate the $\#ReW$, the remaining words (in bold) are shown in Figure 6.

$$p(s_1, s_2) = 0.5 \times (\frac{|5 - 1|}{max(5, 1)})^3 = 0.26 \quad (5)$$

$$SimMat = 0.87 \times (1 - 0.26) = 0.64 \quad (6)$$

## 4 Combination of $SimMat$ metric and MT metrics

We proposed paraphrase detection method by combining the $SimMat$ metric with the eight standard MT metrics described above, as shown in Figure 7. The last two steps are described in detail below.

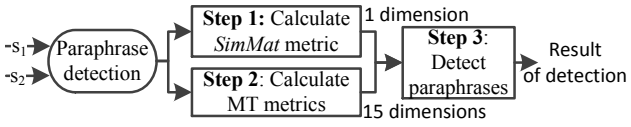


Figure 7: Combination of $SimMat$ metric with eight MT metrics.

### 4.1 Calculate MT metrics (Step 2)

The eight standard MT metrics are calculated for the two sentences. Eight libraries are used to quantify them. These libraries are suggested by NIST and the state-of-the-art approach for paraphrase detection (Madnani et al., 2012). The libraries are described in more detail in the evaluation section. The first six MT metrics (MAXSIM, SEPIA, TER, TERp, METEOR, and BADGER) create six dimensions in total. The two remaining metrics (BLEU and NIST) using the $n$-gram model create four ($n$=1..4) and five ($n$=1..5) dimensions, respectively. These 15 dimensions metrics are combined with that of our proposed metric ($SimMat$) for detecting paraphrases in the last step.

### 4.2 Detecting paraphrases (Step 3)

The 16 dimensions, 15 from the MT metrics and 1 from our proposed metric ($SimMat$) are combined for detecting paraphrases using a machine learning approach. Several commonly used machine learning algorithms (including support vector machine, logistic regression, etc.) were evaluated with these dimensions. Such algorithms are run with 10-fold cross validation in the training set of the MRPS corpus for choosing the best classifier. Logistic regression had the best performance and was used for detection.

## 5 Evaluation

### 5.1 MSRP corpus

We used the MSRP corpus to evaluate our method. It contains 5801 sentences pairs including 4076 for training and the remaining 1705 for testing.

The corpus has 2753 (67.5%) and 1147 (66.5%) paraphrase cases corresponding to training and testing datasets. The corpus was annotated by two native speakers. Disagreements in annotation were resolved by a third native speaker. Agreement between the two annotators was moderate to high (averaging 83%). This means that a perfect algorithm for detecting paraphrases would have 83% accuracy.

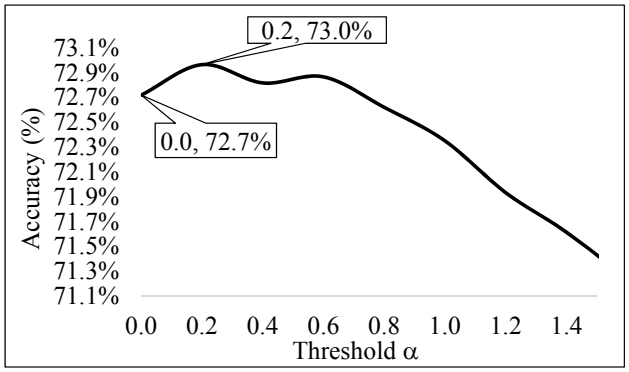


Figure 8: Estimated threshold $\alpha$.

### 5.2 Estimating threshold $\alpha$ for $SimMat$ metric

A threshold $\alpha$ is used to adjust the contributions of matched identical phrases and matched similar words. It was estimated using Eq. 2 and the training dataset of the MRPS corpus. The $SimMat$ metric was used as the single dimension for the logistic regression algorithm with 10-fold cross validation, as shown in Figure 8. Using only the training dataset ensured that the results did not overfit the test data.

The higher the threshold $\alpha$, the greater the contribution of the matched identical phrases and the lesser the contribution of the matched similar words. If $\alpha$ is small, the contributions of identical phrases are low and the contributions of similar words are high, resulting in lower accuracy. However, the $SimMat$ metric is over-estimated if the value of $\alpha$ is too large, resulting in lower accuracy. The highest accuracy (73.0%) was achieved for $\alpha = 0.2$. Therefore, $\alpha$ was set to 0.2 for the subsequent experiments.

### 5.3 MT metrics result

In our approach, the proposed metric ($SimMat$) is combined with eight MT metrics (MAXSIM, SEPIA, TER, TERp, METEOR, BADGER, BLEU, and NIST). These metrics are integrated to create what we call the MTMETRICS algorithm, which is state of the art for paraphrase detection. The eight metrics are re-implemented on the basis of standard libraries suggested by both of the state of the art and a well-known organization – NIST. The details of

| MT metric | Re-implementation | | | MTMETRICS | |
|---|---|---|---|---|---|
| | Ver. | Acc. | F1 | Acc. | F1 |
| MAXSIM | 1.01 | 67.5% | 79.4% | 67.2% | 79.4% |
| SEPIA | 0.2 | 68.3% | 79.8% | 68.1% | 79.8% |
| TER | 1.01 | 70.1% | 81.0% | 69.9% | 80.9% |
| TERP | 1.0 | 70.7% | 81.0% | 74.3% | 81.8% |
| BADGER | 2.0 | 67.2% | 79.9% | 67.6% | 79.9% |
| METEOR | 1.5 | 71.7% | 80.0% | 73.1% | 81.0% |
| BLEU | 13a | 72.1% | 80.8% | 72.3% | 80.9% |
| NIST | 13a | 71.8% | 80.4% | 72.8% | 81.2% |
| **Integration** | | 76.6% | 83.1% | 77.4% | 84.1% |

Table 1: Results for re-implemented MT metrics and MTMETRICS algorithm (Madnani et al., 2012).

| Method | Accuracy | F-score |
|---|---|---|
| Vector Based Similarity (baseline) | 65.4% | 75.3% |
| Mihalcea et al. (2006) | 70.3% | 81.3% |
| Qiu et al. (2006) | 72.0% | 81.6% |
| $SimMat$ | 72.7% | 81.3% |
| Blacoe and Lapata (2012) | 73.0% | 82.3% |
| Finch et al. (2005) | 75.0% | 82.7% |
| Das and Smith (2009) | 76.1% | 82.7% |
| *Madnani et al. (2012) (re-implemented)* | 76.6% | 83.1% |
| Socher et al. (2011) | 76.8% | 83.6% |
| Madnani et al. (2012) | 77.4% | **84.1%** |
| **Combination** | **77.6%** | 83.9% |

Table 2: Accuracy and F-score of our method ($SimMat$), previous methods, and combination of $SimMat$ with eight MT metrics.

the re-implementation are shown in Table 1.

The versions of the eight libraries for the re-implemented metrics are shown in column 2. They were the latest for each library, for which we used the default settings. Since the versions and settings are not shown for MTMETRICS, there is little difference between the re-implemented metric results and the MTMETRICS results. The results for the integration of the eight re-implemented metrics (accuracy=76.6%, F1=83.1%) also differ from the MTMETRICS results (accuracy=77.4%, F1=84.1%).

### 5.4 Comparison with previous methods

The results of our comparison with previous methods are summarized in Table 2. These methods were also evaluated using the MRPS corpus. Our proposed metric ($SimMat$) was evaluated using a threshold $\alpha$ of 0.2. This single metric outperformed many previous methods. The combination of $SimMat$ with the eight MT metrics had the highest accuracy (77.6%).

## 6 Conclusion

Our proposed similarity matching ($SimMat$) metric quantifies the similarity between two sentences and can be used to detect whether one is a paraphrase of the other. It is calculated using the matching of identical phrases and similar words. Phrase-by-phrase matching is done using a heuristic algorithm that determines the longest duplicate phrase in each iteration. Word matching is done using the Kuhn-Munkres algorithm. WordNet is used for determining the similarity of two words. This similarity is used as the weights for the word-matching algorithm. Minor words, which are often added or removed from paraphrased text to improve naturalness, can create noise when detecting paraphrases. They are thus removed as doing so generally does not change the meaning. A brevity penalty metric is combined with the $SimMat$ metric to quantify the effect of inserting and/or deleting words.

Evaluation using the MSRP corpus showed that the $SimMat$ metric detects paraphrases more effectively than previous methods. The $SimMat$ metric was combined with eight machine translation metrics. Although the accuracy of the eight re-implemented metrics (accuracy=76.6%, F-score=83.1%) was lower than the published result (accuracy=77.4%, F-score=84.1%), their combination with the $SimMat$ metric achieved the best accuracy (77.6%), which was higher than with the state-of-the-art approach (77.4%). Moreover, the F-score of the combination (83.9%) is nearly similar with the-state-of-the-art approach (84.1%). These results show that our method is promising approach to detecting paraphrasing.

Future work includes quantifying the weights of words in matched phrases, determining the effect of a word's position in a sentence, and analyzing misclassified pairs to improve performance.

## References


William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *EMNLP*, pages 546–556.

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

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *ACL*, pages 468–476.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *NAACL*, pages 250–253.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. of the 2nd International Conference on Human Language Technology Research*, pages 138–145.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*, pages 350–356.

Andrew Finch, Young-Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proc. of the 3rd International Workshop on Paraphrasing*, pages 17–24.

Nizar Habash and Ahmed Elkholy. 2008. Sepia: surface span extension to syntactic dependency precision-based mt evaluation. In *Proc. of Association for Machine Translation in the Americas*.

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1):83–97.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *NAACL*, pages 182–190.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60.

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.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Steven Parker. 2008. Badger: A new machine translation metric. In *Proc. of Association for Machine Translation in the Americas*.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *NAACL: Demonstration*, pages 38–41.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *EMNLP*, pages 18–26.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of Association for Machine Translation in the Americas*, pages 223–231.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

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 *NIPS*, pages 801–809.