Unsupervised Learning of Paraphrases

João Cordeiro[1], Gaël Dias[1], and Pavel Brazdil[2]

[1] University of Beira Interior, HULTIG***, 6200-001 Covilhã, Portugal,
{jpaulo,ddg}@hultig.di.ubi.pt,
http://hultig.di.ubi.pt/
[2] University of Porto, NIIAD†,
4150-190 Porto, Portugal,
http://www.niaad.liacc.up.pt/

Abstract. Paraphrasing constitutes a corner stone in many Natural Language Processing fields like monolingual text-to-text generation and automatic text summarization. Indeed, aligned monolingual corpora are likely to boost the learning process of text-to-text generation models. A Paraphrase learning strategy can be defined as a two-step process: (1) identifying and extracting related sentence pairs from on-line comparable corpora (for example sentences that convey the same information but yet are written in different forms) and (2) applying learning methodologies over the extracted material to induce text-to-text rewriting rules. In this paper, we compare different lexical distance metrics for the identification of related sentences, i.e. paraphrase candidates. In particular, we discuss how different metrics lead to the identification of different types of paraphrases. Finally, the comparisons and discussions give relevant insights towards automatic generation of paraphrase corpora.

1 Introduction

Monolingual text-to-text generation is an emerging research area in Natural Language Processing, as described by [1]. Unlike in traditional concept-to-text generation, text-to-text generation applications take a text as input and transform it into a new text satisfying specific constraints, such as length in summarization [8, 10, 11, 22, 1, 13] or style in text simplification [4, 3, 12, 17]

Such research fields usually rely on monolingual comparable text corpora, such as paraphrase corpora, which tend to be very difficult to be supplied by humans. Therefore, automatic processes for the construction of such corpora are a critical issue. In fact, text-to-text generation is a particularly promising research direction given that there are naturally occurring examples of comparable texts that convey the same information yet are written in different styles. Web news stories are an obvious example of these non-parallel comparable corpora. In our case, we aim at learning monolingual asymmetric text-to-text generation models, specially for the purpose of summarization, in particular sentence compression.

*** Centre for Human Language Technology and Bioinformatics
† Artificial Intelligence and Data Analysis Group

In this context, a collection of sentence pairs where one is a simplified version from the other will be a valuable resource. So, presented with such texts, we are interested in efficiently identifying and extracting sentence pairs that convey the same information, thereby building a training set of rewriting examples i.e. a paraphrase corpus. These pairs of sentences share almost the same meaning, but contain different lexical elements and possibly different syntactic structures. The issue of paraphrase detection from texts is complex, since we may have sentence pairs that almost do not share any lexical similarity but are in fact paraphrases, as described in [2]. So far, we are only interested on efficient surface methods for paraphrase detection.

However, the unsupervised methodologies proposed so far [1, 7] are not well tailored for the reality and special needs of paraphrase detection, showing a major drawback, by extracting quasi-exact or even exact match pairs of sentences. This is mainly due to the fact that they rely on classical string similarity measures such as the Edit Distance in the case of [7] and word overlap for [1]. Such pairs are obviously useless for us, since we are interested in learning sentence compression patterns.

In this paper, we compare a set of string similarity metrics for paraphrase extraction and also suggest a new metric category[3], to serve our particular needs. This metric family was thought to deal with the classical metrics limitations and perform efficiently. We show that it competes well and even outperforms all state-of-the-art metrics both in the general case where exact and quasi-exact pairs do not occur and in a more realistic scenario, where exact and quasi-exact pairs occur (like in web news stories). For convenience in writing, we denote this family as the *LogSim* family.

In fact, the *LogSim* family extracts a great deal of asymmetric entailed sentence pairs. In such a pair $\langle S_a, S_b \rangle$ sentence S_a entails sentence S_b ($S_a \trianglerighteq S_b$), but S_b does not entail S_a ($S_a \ntrianglerighteq S_b$), or vice-versa:

> S_a**:** *The control panel looks the same but responds more quickly to commands and menu choices.*
> S_b**:** *The control panel responds more quickly.*

This particular case is much more challenging for classical string similarity measures that have been conceived for exact match of string pairs.

2 Related Work

The issue of finding paraphrases in monolingual comparable corpora is recently becoming more and more relevant as researchers realize the importance of such resources for Information Retrieval, Information Extraction, Automatic Text Summarization and Automatic Text Generation [10, 11, 22, 1, 13, 17].

[3] It is a parameterized set of metrics, not just one.

In particular, three different approaches have been proposed for paraphrase detection: unsupervised methodologies based on lexical similarity [1, 7], supervised methodologies based on context similarity measures [2] and methodologies based on linguistic analysis of comparable corpora [9].

[7] endeavored a work to find and extract monolingual paraphrases from massive comparable news stories. They use the Edit Distance (also known as *Levenshtein Distance* [14]) and compare it with an heuristic derived from Press writing rules that considers initial sentences, from equivalent news stories, as paraphrases. The evaluation shows that the data produced by the Edit Distance is cleaner and more easily aligned than by using the heuristic. However, using word error alignment rate (AER), a metric borrowed from statistical machine translation [18], results show that both techniques perform similarly.

[1] used the simple word n-gram ($n = 1, 2, 3, 4$) overlap measure in the context of paraphrase lattices learning. In particular, this string similarity measure is used to produce clusters of paraphrases using hierarchical complete-link clustering. This metric is usually used for string comparison in Natural Language Processing applications [23, 16]. We will see in section 6 that simple word n-gram overlap also performs well for our purpose.

More deepening techniques rely on context similarity measures such as [2]. They find sentence alignments in comparable corpora by considering sentence contexts (local alignment) after semantically aligning equivalent paragraphs. To combine the lexical similarity[4] and the proximity feature, local alignments are computed on each paragraph pairs using dynamic programming. Although this methodology shows interesting results, it relies on supervised learning techniques, which need huge quantities of training data that may be scarce and difficult to obtain.

Others, such as [9], go further by exploring harvesting linguistic features combined with machine learning techniques to propose a new text similarity metric. Once again, it is a supervised approach and also heavily dependent on valuable linguistic resources which are usually not available for the vast majority of languages. We agree on the fact that linguistic resources may improve accuracy and accordance with human judges but they shorten the application of such systems to very few languages.

3 Metrics Overview

In the literature [14, 19, 1], we can find the *Levenshtein Distance* [14] and the *Word N-Gram Overlap Family* of similarity measures [19, 1]. Indeed in the latter case, some variations of word n-gram overlap measures are proposed but not clearly explained. In this section, we will review all the existing metrics that may be conveniently adapted and used for surface paraphrase pairs detection.

[4] With the cosine similarity measure.

3.1 The Levenshtein Distance

This metric, also known as *Edit Distance*, was created for string similarity computation. Considering two strings, the metric computes the number of character insertions, deletions and substitutions that would be needed to transform one string into the opposite [14]. It was adapted for sentence proximity calculation by [7] as a surface metric for paraphrase detection on monolingual parallel texts. Within sentences, words are considered as characters.

As a consequence, an evident problem arises from using the Edit Distance for sentence proximity calculation, since there are pairs that are true paraphrases, however, as they contain high lexical alternations or different syntactic structures they are likely to receive high distance values. This indicates low inter-pair relatedness and erroneous paraphrase classification. In practice, the pair will not be identified and extracted. For example, it is unlikely that sentences (1) and (2) would be extracted as paraphrases, since their *Edit Distance* shows high value.

(1) *Due to high energy prices, our GDP may continue to fall, said Prime Minister, early morning.*

(2) *Early morning, Prime Minister claim that our GDP will continue to fall, due to growing energy prices.*

3.2 The Word N-Gram Family

In fact, there exist not only one, but a family of text similarity measures based on word n-gram overlap that have been used to measure text proximity, like summary quality evaluation (BLEU and ROUGE). Sometimes it is not clear or unspecified which word n-gram version is used. In fact, two metrics are usually found: the Word Simple N-gram Overlap and the BLEU/ROUGE Metric.

Word Simple N-gram Overlap: This is the simplest metric that uses word n-gram overlap between sentences. For a given sentence pair, the metric counts how many 1-grams, 2-grams, 3-grams, ..., N-grams overlap. Usually N is chosen equal to 4 or less [1]. Let's name this counting function $Count_{match}$(n-gram). For a given $N \geqslant 1$, a normalized metric that equally weights any matching n-gram and evaluates similarity between 2 sentences S_a and S_b, is shown in Equation 1:

$$NGsim(S_a, S_b) = \frac{1}{N} * \sum_{n=1}^{N} \frac{Count_{match}(\text{n-gram})}{Count(\text{n-gram})} \tag{1}$$

where the function $Count$(n-gram) counts the maximum number of n-grams that exist in the shorter sentence as it rules the max number of overlapping n-grams.

The BLEU/ROUGE Metric: The BLEU metric was introduced by [19] for automatic evaluation of machine translation, and was after adapted to automatically evaluate summaries and subsequently renamed as ROUGE by [5]. It is clear that these metrics can easily be adapted to calculate similarity between two sentences as they are based on string overlaps. When this adaptation is taken into account, both BLEU and ROUGE are transformed into the same metric. In particular, they were used to evaluate precision (BLEU) and recall (ROUGE), by considering n-gram matches between a text (a translation or a summary), and a set of candidate texts. It is just a slight difference of sums that gives the difference. In our case, we just have two texts (the two sentences, not a set of candidate texts or reference summaries). Therefore the adapted two metrics are the same. We call it the $BLEU_{adapted}$ metric and it is shown in equation 2:

$$BLEU_{adapted} = \frac{1}{N} * exp[\sum_{n=1}^{N} log \sum_{\text{n-gram}} \frac{Count_{match}(\text{n-gram })}{Count(\text{n-gram })}] \quad (2)$$

The $Count_{match}$(n-gram) function counts the number of exclusive or no-exclusive n-grams co-occurring between the two sentences, and the function $Count$(n-gram) the maximum number of n-grams that exist in the shorter sentence[5].

4 The LogSim family

In our main research topic (Automatic Sentence Compression) paraphrase corpora are relevant to induce compression models, as in [11, 22, 1, 13]. As we already told, in the context of our work, we are especially interested in building asymmetric paraphrase corpora.

4.1 Main Motivation

We propose the automatic construction of a huge paraphrase corpus, a valuable resource for research topic. It is not the first work in automatic paraphrase corpus construction [1, 7] but it is the only one that clearly addresses the problems of existing string similarity metrics. Indeed, when applying existing metrics for paraphrase detection in real-world conditions like in Web news stories, most of the results are exact or quasi-exact match pairs of sentences. Such results are obviously useless, since we are interested on asymmetric paraphrases, where one sentence contains more information, possibly irrelevant, than the other.

For that purpose, we designed a new metric family to detect paraphrases that avoids the extraction of exact and quasi-exact matches and outperforms state-of-the-art metrics in most of the evaluations presented in 6. In fact, four main premises guided our research: (1) Achieve maximum automation in corpus construction - minimum or even no human intervention, with high reliability, (2)

[5] In our experiments, we will only show the results with no-exclusive n-grams as results were worst with exclusive n-grams.

Penalize equal and almost equal sentences - they are not useful for our research needs, but frequent in real-world situations, (3) Consider pairs having a high degree of lexical reordering, and different syntactic structure and (4) Define a computationally fast and well founded metric.

The basic idea of the *LogSim* family lays on the notion of exclusive lexical links between pairs of sentences, as shown in figure 1.

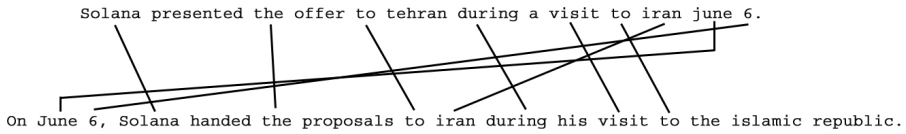

Fig. 1. Links between a pair of sentences extracted from *Web news stories*.

It is another form to think about 1-gram exclusive overlap. If a link is established between sentence S_a and sentence S_b, for the word **w**, then other occurrences of word **w** in sentence S_a will engage a new link to sentence S_b if there exists at least one more occurrence of **w** in S_b, besides the one which is already connected.

4.2 LogSim

We define the number of links between the two sentences as λ and the number of words in the longest sentence as x. The fraction $\frac{\lambda}{x}$ in $[0, 1]$ indicates a normalized lexical connectivity among sentences, by the longest sentence. As $\frac{\lambda}{x} \longrightarrow 1$, it is more likely that both sentences are equal, and with $\frac{\lambda}{x} = 1$, they are exactly equal. Remark that even if the shortest sentence is strictly contained in the longest one, we have $\frac{\lambda}{x} < 1$

To calculate the *LogSim* metric, $LogSim(.,.)$, we first evaluate the function $L(x, \lambda)$ as in Equation 3[6]

$$L(x, \lambda) = -\log_2(\frac{\lambda}{x}) \tag{3}$$

Then the $LogSim(.,.)$ is obtained as in Equation 4 ensuring that the function range lays in the interval $[0, 1]$. The logarithm is used as a mechanism to gradually penalize sentence pairs that are too similar, returning exactly zero when we have an exact match.

$$LogSim(S_a, S_b) = \begin{cases} L(x, \lambda) \; if \;\; L(x, \lambda) < 1.0 \\ \\ e^{-k*L(x,\lambda)} \; otherwise \end{cases} \tag{4}$$

[6] When $\lambda = 0$, $L(x, \lambda) = 0$.

The main objective of the second branch $e^{-k*L(x,\lambda)}$ is to penalize pairs with great dissimilarities, since in this case $\frac{\lambda}{x} \longrightarrow 0$ and naturally $L(x,\lambda) \longrightarrow +\infty$. For example, if $x = 30$, $y = 5$ and $\lambda = 4$ (y is the number of words in the shortest sentence), we get $L(x,\lambda) = 2.9068$ and the final $LogSim(.,.)$ value is repositioned in $[0,1]$ with a low value of 0.00039. The k value is a positive parameter used to boost the penalization branch. In our experiments, we used $k = 2.7$. In fact, the greater the k, the greater the penalization will be for dissimilar pairs.

4.3 Expanded LogSim

The $L(x,\lambda)$ function, defined in the previous subsection, seems to be independent from the number of words in the shorter sentence y. This is not completely true since $\lambda \leq y$. However, one may thing about considering a broader function that also depends explicitly on y. The natural idea that follows is to consider $\frac{\lambda}{y}$ and also applying the log_2 mechanism for high-similarity penalization, i.e $L(y,\lambda)$. So, a new function $L(.,.,.)$ depending on the tree parameters x, y and λ could be a linear interpolation of $L(x,\lambda)$ and $L(y,\lambda)$, as shown in Equation 5:

$$L(x,y,\lambda) = -\alpha * L(x,\lambda) - \beta * L(y,\lambda) \tag{5}$$

where α and β are such that $\alpha \in [0,1]$ and $\alpha = 1 - \beta$, i.e weighting factors. Remark that $L(x,y,\lambda) = L(x,\lambda)$ when $\alpha = 1$. By varying the α weights different values of the function $L(x,y,\lambda)$ can be obtained.

Similarly to the $LogSim(.,.)$, in order to ensure that the function range lays in the interval $[0,1]$, we define the $LogSimX(.,.)$ in equation 6.

$$LogSimX(S_a,S_b) = \begin{cases} L(x,y,\lambda)\ if\ \ L(x,y,\lambda) < 1.0 \\ \\ e^{-k*L(x,y,\lambda)}\ otherwise \end{cases} \tag{6}$$

In particular, we will show the results obtained for $\alpha = 0.25, 0.5, 0.75$, in section 6.

4.4 Complexity

The *LogSim* family was conceived to be as simple and efficient as possible, since no digram, trigram or n-gram ($n > 1$) is computed. Only unigrams (words) are taken into account to calculate the λ value (number of links between sentences). In the worst case, this computation is done in $\Theta(x*y)$ time - when the sentences are completely different, i.e. there is no link among them. In that case, we compute $x*y$ comparisons i.e. each word in one sentence is compared with each word in the other. In the best situation the computation will take only $\Theta(y)$ time. This is the case when the shortest sentence is a prefix of the longest one. However, these are extreme situations and the real complexity lays between these two - $\Theta(y) \leq \Theta(LogSim) \leq \Theta(x*y)$. Similarly, the word *Edit Distance* takes at least

$\Theta(x * y)$ time complexity by using the commonly-used bottom-up dynamic programming algorithm. On the opposite, any N-gram based metric requires more computations: $\Theta(N * x * y)$, where N is the maximum number of N-grams considered. For example, if $N = 3$ only unigrams, bigrams and trigrams are counted which takes $\Theta(x * y) + \Theta((x-1) * (y-1)) + \Theta((x-2) * (y-2)) = \Theta(3 * x * y)$ operations. Empirical computations, realized over huge text collections, support these statements, showing considerable time differences in real-world conditions. In conclusion, we may abusively state that $\Theta(LogSim) = \Theta(LogSimX) \leq \Theta(Edit) \leq \Theta(Ngram)$.

5 The Corpora Set

Two standard corpora were used for comparative tests between metrics: The Microsoft Research Paraphrase Corpus [7] and a corpus supplied by Daniel Marcu that has been used for research in the field of Sentence Compression [11, 13]. By adapting these corpora we created three new corpora to serve as a benchmark for our specific purpose.

5.1 The Microsoft Paraphrase Corpus

In 2005, Microsoft researchers Dolan, Brocket, and Quirck [7] published the first paraphrase corpus containing 5801 pairs of sentences with 3900 tagged as "semantically equivalent" or true paraphrases. Sentences were obtained from massive parallel news sources and tagged by 3 human raters according to guidelines described in [7]. We will refer to this corpus as the label $\{MSRPC\}$.

5.2 The Knight and Marcu Corpus.

The corpus used by Kevin Knight and Daniel Marcu in their Sentence Compression research work [11], contains 1087 sentence pairs, where one sentence is a compressed or summarized version of the other one. This corpus was produced completely manually from pairs of texts and respective summaries. We label $\{KMC\}$ this corpus.

5.3 The Corpora Used

One major limitation with the $\{KMC\}$ corpus is that it only contains positive pairs. Therefore it should not be taken as such to perform any evaluation. Indeed, we need an equal number of negative pairs of sentences to produce a fair evaluation for any paraphrase detection metric. Although the $\{MSRPC\}$ corpus already contains negative pairs, they are only 1901 against 3900 positive examples. To perform an equitable evaluation, we first expanded both corpora by adding negative sentence pairs selected from *Web news corpora* so that they have the same number of positive and negative examples and also created a new corpus based on the combination of the $\{MSRPC\}$ and the $\{KMC\}$.

The $\{MSRPC \cup X^{-}_{1999}\}$ Corpus: This new derived corpus contains the original $\{MSRPC\}$ collection of 5801 pairs (3900 positives and 1901 negatives) plus 1999 extra negative sentences (symbolized by X^{-}_{1999}), selected from Web news stories. So we end with 3900 positive pairs and 3900 negative ones.

The $\{KMC \cup X^{-}_{1087}\}$ Corpus: From the $\{KMC\}$, we derived a new corpus that contains its 1087 positive pairs plus a set of negative pairs, in equal number, selected from Web news stories. We named this new corpus $\{KMC \cup X^{-}_{1087}\}$, where the X^{-}_{1087} stands for extra negative paraphrase pairs (1087 in this case).

The $\{MSRPC^{+} \cup KMC \cup X^{-}_{4987}\}$ Corpus: Finally we decided to build a bigger corpus that gathers the positive $\{MSRPC\}$ part i.e. 3900 positive examples, and the 1087 positive pairs of sentences from the $\{KMC\}$ corpus, giving a total of 4987 positive pairs. To balance these positive pairs we added an equal number of negative pairs, selected in a same manner as described previously. We labeled this wider corpus the $\{MSRPC^{+} \cup KMC \cup X^{-}_{4987}\}$ corpus. In this corpus, we intentionally ignored the $\{MSRPC\}$ negative pairs as many pairs that are labeled negative, following the guidelines expressed in [7], are in fact useful paraphrases.

6 Results

This work makes a comparative study between metrics for paraphrase identification, including a new family of metrics (*LogSim*) for asymmetric paraphrase detection. A new benchmark of paraphrase test corpora was also proposed and used for metric testing.

In order to evaluate the results of each metric over each corpus, we computed *F-Measure* and *Accuracy*. The results were calculated by averaging the 10 *F-Measure* and *Accuracy* values obtained from a *10-fold cross validation* test. For every fold, the best threshold was found on the $\frac{9}{10}$ training data and then used on the $\frac{1}{10}$ test block to measure the correspondent *F-Measure* and *Accuracy*. The results for all measures except the $LogSimX$[7] are presented in Table 1.

Table 1. $F - Measure$ and *Accuracy* results obtained.

	A F_β	A *Accuracy*	B F_β	B *Accuracy*	C F_β	C *Accuracy*
Edit	74.41%	67.67%	70.65%	68.02%	80.98%	79.02%
NGsim	78.06%	73.15%	**94.66%**	**94.47%**	91.92%	91.79%
BLEU	70.77%	66.17%	82.39%	78.89%	76.79%	74.13%
LogSim	**80.43%**	**78.19%**	92.14%	92.58%	**97.12%**	**97.12%**

[7] A specific analysis is made in Table 2.

The *F-measure* and the *Accuracy* are respectively defined in Equation 7 and 9. In particular, the experiments with the *F-Measure* were made with $\beta = 1$.

$$F_\beta = \frac{(1+\beta^2) * precision * recall}{\beta^2 * precision + recall} \tag{7}$$

with

$$precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN} \tag{8}$$

where TP are True Positives, TN True Negatives, FP False Positives and FN False Negatives.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

The results evidenced in Table 1 show that the *LogSim* outperforms all state-of-the-art metrics over all corpora, except on B corpus where *NGsim* achieved better results. However, it is important to observe that this corpus is the smallest one (2174 pairs).

As described in subsection 4.3, an expanded version of *LogSim*, the *LogSimX* has also been experimented and the results are shown in Table 2 for $\alpha = 0.25$, $\alpha = 0.5$, and $\alpha = 0.75$. This means that in the first case more weight is given to $\frac{\lambda}{x}$, in the second $\frac{\lambda}{x}$ and $\frac{\lambda}{y}$ are equally weighted, and in the last case more relevance is given to factor $\frac{\lambda}{y}$.

Table 2. $F - Measure$ and *Accuracy* results obtained.

	A	A	B	B	C	C
	F_β	*Accuracy*	F_β	*Accuracy*	F_β	*Accuracy*
$LogSimX_{\alpha=0.25}$	80.74%	77.63%	98.67%	98.66%	98.31%	98.30%
$LogSimX_{\alpha=0.50}$	80.94%	78.19%	98.47%	98.47%	98.49%	98.48%
$LogSimX_{\alpha=0.75}$	80.68%	78.58%	96.48%	96.50%	97.98%	97.96%

The conclusion from Table 2 is that the *LogSimX* performs better than the *LogSim* and is best tailored for asymmetric pair detection, since better results are obtained even over the **B** corpus which contains the type of pairs we want to identify. This shows that the combination of both components $\frac{\lambda}{x}$ and $\frac{\lambda}{y}$ achieves better results than using only the first one. Moreover, the *LogSimX* shows improved results over all existing state-of-the-art metrics.

The BLEU[8] metric and the Edit Distance obtain the worst results over all corpora. The BLEU performed better only over the **B** corpus, where the positive pairs are all asymmetric, i.e one sentence is a compressed version of the other one. This is comprehensible, since such pairs generate great dissimilarity values when

[8] This is the $BLEU_{adapted}$ as described in subsection 3.2

the Edit Distance is applied. Remark that $LogSimX_{\alpha=0.5}$ performs at 98.47% in this corpus. The Edit Distance gives better results than the BLEU metric for the **A** corpus as it contains more near string matches as positive examples. Remember that this corpus was created with the Edit Distance guidance.

Finally, the **C** corpus is a more general and realistic corpus, containing pairs from different types: symmetric, asymmetric, quasi-equal, and completely dissimilar. We only remark that in this corpus $LogSimX_{\alpha=0.5}$ correctly classified, on average, 98.48% of all the 9974 sentence pairs. In fact, it shows systematically better F-Measure and Accuracy measures over all other metrics showing an improvement of (1) at least 2.88% in terms of F-Measure and 5.04% in terms of Accuracy and (2) at most 3.81% in terms of F-Measure and 4% in terms of Accuracy compared to the second best metric which is also systematically the $NGsim$ similarity measure.

7 Conclusion and Future Work

In this paper, we made a comparative study among a set of similarity metrics for paraphrase identification and extraction in text corpora. Some of these were already used for the same task. However, in order to be complete, we proposed a new similarity metric family for asymmetric paraphrase detection - the *LogSim* family. We also proposed a new benchmark of paraphrase test corpora and tested all state-of-the-art metrics on these corpora. One main and general conclusion is that the *LogSimX* performs better than any other metric, over all corpora either in terms of F-Measure and Accuracy. The *Levenshtein Distance* [14] performs poorly over corpora with high lexical and syntactic diversity unlike the BLEU measure. However, when paraphrases are almost string matches, the Edit Distance outperforms the BLEU measure.

In the future we will try to insert tf.idf [21] information in our metric, as we believe that word links between sentences should have distinct weights. Indeed, it is different to have a match between determinants (with low tf.idf) or between verbs or nouns/names (with high tf.idf). Verbs and nouns/names obviously convey relevant information about the sentence while it is not the case for determinants. We may also integrate the notion of content character n-grams that can be extracted from monolingual corpora as in [6].

8 Acknowledgement

We are grateful to Daniel Marcu for providing us with his corpus and we would also thank the Portuguese *Fundação para a Ciência e a Tecnologia* agency for funding this research (Reference: POSC/PLP/57438/2004).

References

1. R. Barzilay and L. Lee: Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In Proceedings of Human Language Technology and

North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), Edmonton, Canada, 2003.
2. R. Barzilay and N. Elhadad: Sentence Alignment for Monolingual Comparable Corpora. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003), pages:25-33, Sapporo, Japan, 2003.
3. J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin and J. Tait: Simplifying Text for Language-Impaired Readers, In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999), Bergen, Norway, 1999.
4. R. Chandrasekar and S. Bangalore: Automatic Induction of Rules for Text Simplification, Knowledge-Based Systems, 10(3):183190, 1997.
5. Chin-Yew Lin: ROUGE: A Package for Automatic Evaluation of Summaries, In Proceedings of Workshop on Text Summarization Branches Out (ACL 2004), Barcelona, Spain 2004
6. G. Dias, S. Guilloré and J.G.P. Lopes: Extraction Automatique d'Associations Textuelles à Partir de Corpora Non Traités, In Proceedings of 5th International Conference on the Statistical Analysis of Textual Data, pages:213-221, 2000.
7. W.B Dolan, C. Quirck and C. Brockett: Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources, In Proceedings of 20th International Conference on Computational Linguistics (COLING 2004), Geneva, Switzerland, 2004.
8. G. Grefenstette: Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind, In Proceedings of AAAI spring Workshop on Intelligent Text Summarization, Palo Alto, USA, 1998.
9. V. Hatzivassiloglou, J.L. Klavans and E. Eskin: Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning, In Proceedings of Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP 1999), University of Maryland, USA, 1999.
10. H. Jing and K. McKeown: Cut and Paste based Text Summarization, In Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics, pages:178-185, Seattle, USA, 2000.
11. K. Knight and D. Marcu: Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. Artificial Intelligence, 139(1):91-107, 2002.
12. M. Lapata: Probabilistic Text Structuring: Experiments with Sentence Ordering, In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan, 2003.
13. M. Le Nguyen, S. Horiguchi, A. Shimazu and B. Tu Ho: Example-based Sentence Reduction using the Hidden Markov Model, ACM Transactions on Asian Language Information Processing (TALIP), 3(2):146-158, 2004.
14. V. Levenshtein: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals., Soviet Physice-Doklady, 10:707-710, 1966.
15. C.Y. Lin and E.H. Hovy: Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics, In Proceedings of Human Language Technology and North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), Edmonton, Canada, 2003.
16. L.V. Lita, M. Rogati, A. Lavie: BLANC: Learning Evaluation Metrics for MT, In Proceedings of Human Language Technology and Empirical Methods in Natural Language Processing Joint Conference (HLT-EMNLP 2005), Vancouver, Canada, 2005.

17. Marsi, E. and E. Krahmer: Explorations in Sentence Fusion, In Proceedings of the 10th European Workshop on Natural Language Generation, Aberdeen, Scotland, 2005.
18. F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models, Computational Linguistics, 29(1):19-51, 2003.
19. K. Papineni, S. Roukos, T. Ward, W.-J. Zhu: BLEU: a Method for Automatic Evaluation of Machine Translation, IBM Research Report RC22176, 2001.
20. E. Polak: Computational Methods in Optimization, New York Academic Press, 1971.
21. G. Salton and C. Buckley: Term Weighting Approaches in Automatic Text Retrieval, Information Processing and Management, 24(5):513-523, 1988.
22. Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman: Automatic Paraphrase Acquisition from News Articles, In Proceedings of Human Language Technology (HLT 2002), Sao Diego, USA, 2002.
23. J. Sjöbergh and Kenji Araki: Extraction based summarization using a shortest path algorithm, In Proceedings of 12th Annual Language Processing Conference (NLP 2006), Yokohama, Japan, 2006.
24. M. Yamamoto and Church, K.: Using Suffix Arrays to Compute Term Frequency and Document Frequency for all Substrings in a Corpus, Computational Linguistics, 27(1):1-30, 2001.