New Functions for Unsupervised Asymmetrical Paraphrase Detection

Cordeiro João, Dias Gaël and Brazdil Pavel(†)
HULTIG, University of Beira Intetrior, Covilhã, Portugal
(†) LIACC, University of Porto, Porto, Portugal
Email: {jpaulo, ddg}@hultig.di.ubi.pt, pbrazdil@liacc.up.pt

***Abstract*— Monolingual text-to-text generation is an emerging research area in Natural Language Processing. One reason for the interest in such generation systems is the possibility to automatically learn text-to-text generation strategies from aligned monolingual corpora. In this context, paraphrase detection can be seen as the task of aligning sentences that convey the same information but yet are written in different forms, thereby building a training set of rewriting examples. In this paper, we propose a new type of mathematical functions for unsupervised detection of paraphrases, and test it over a set of standard paraphrase corpora. The results are promising as they outperform state-of-the-art functions developed for similar tasks. We consider two types of paraphrases - symmetrical and asymmetrical entailed - and show that although our proposed functions were conceived and oriented toward the asymmetrical detection, they perform rather well for symmetrical sentence pairs identification.**

***Index Terms*— Paraphrasing, Paraphrase Identification, Sentence compression, Text Summarization, Text Generation, Textual Entailment, Text Mining.**

I. Introduction

In generally, a paraphrase could be defined as a statement or remark explained in other words or another way, so as to simplify or clarify its meaning. Therefore a minimum of two monolingual "text entities" are involved, whether they are texts, paragraphs, sentences or simply phrases. On this article we refer to a paraphrase as a pair of sentences that expresses the same meaning or that that coincide in almost the same semantical items yet are usually written in different styles. We also identify two types of paraphrases, which we designate as symmetrical and asymmetrical. A symmetrical paraphrase complains with the general notion of a paraphrase:

Definition 1: A symmetrical paraphrase is a pair of sentences $\langle S_a, S_b \rangle$, where both sentence contains the same information, or in terms of entailment, each sentence entails the other one, i.e: $S_a \vDash S_b \wedge S_b \vDash S_a$.

We also remark a special type of paraphrases - the asymmetrical case, defined as follows:

Definition 2: An asymmetrical paraphrase is a pair of sentences where at least one sentence is more general or contains more information than the other one, i.e either $S_a \nvDash S_b$ or $S_b \nvDash S_a$.

An example of an asymmetrical paraphrase is shown next:

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.*

Paraphrase corpora are golden resources for learning monolingual text-to-text rewritten patterns[1], satisfying specific constraints, such as length in summarization [1]–[5] or style in text simplification [6]. However, such corpora are very costly to be constructed manually and will always be an imperfect and biased representation of the language paraphrase phenomena. Therefore reliable and efficient automatic methodologies capable of paraphrase extraction from text, and subsequently corpus construction, are crucial. In particular, we are mainly interested in asymmetrical paraphrase corpora construction[2].

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 but yet are written in different styles. *Web News Stories* are obviously a natural space for searching this type of texts. So, presented with such texts, one can pair sentences that convey the same information, thereby building a training set of rewriting examples i.e. a paraphrase corpus. A few unsupervised methodologies have been applied to automatic paraphrase identification and extraction [4], [7]. However, these unsupervised methodologies show a major drawback by extracting quasi-exact[3] or even exact match pairs of sentences, as they rely on classical string similarity measures such as the *Edit Distance* in the case of [7] and word n-gram overlap for [4]. Such pairs are clearly useless for us, since we aim for asymmetrical paraphrase examples, as explained.

As a consequence, we first propose a new function - named the *LogSimX* - that presents a solution to these

This paper is based on "A Metric for Paraphrase Detection," by J. Cordeiro, G. Dias, and P. Brazdil, which appeared in the Proceedings of the The Second International Multi-Conference on Computing in the Global Information Technology (ICCGI07), Guadeloupe, French Caribbean, March 2007. © 2007 IEEE.

This work was supported in part by the Portuguese Science Foundation (FCT) - "The Sumo Project".

[1]For example by applying machine learning techniques

[2]Sice our main research task is sentence compression or summarization.

[3]Almost equal strings, for example: *Bush said America is addicted to oil.* and *Mr. Bush said America is addicted to oil.*

limitations and outperforms all state-of-the-art metrics both in the general case where exact and quasi-exact pairs do not occur and in the real-world case where exact and quasi-exact pairs occur like in *Web News Stories*. Second we investigate well defined mathematical functions that comply with the main characteristics of *LogSimX*, and experimentally show that these functions performed as well as *LogSimX*.

Finally we make a comparative study between already existing functions for paraphrase extraction and our new proposed functions. We named this two types of functions the Asymmetrical Paraphrase functions (AP functions) and Symmetrical Paraphrase functions (SP functions). Results show that SP functions are preferable in any situation, for identifying symmetrical or asymmetrical pairs.

II. 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 [1]–[6], [8]–[11].

In particular, three different approaches have been proposed for paraphrase detection: unsupervised methodologies based on lexical similarity [4], [7], supervised methodologies based on context similarity measures [12] and methodologies based on linguistic analysis of comparable corpora [13].

Microsoft researchers [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, where initial sentences from equivalent news stories are considered 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, evaluating by using *word error alignment rate* (AER), a function borrowed from statistical machine translation [15], evidences that both techniques perform similarly.

In [4] they use the simple word n-gram ($n = 1, 2, 3, 4$) overlap function 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 also often employed for string comparison, in *Natural Language Processing* applications [16], [17]. We will see in section VII that simple word n-gram overlap also performs well, for symmetrical paraphrase identification.

More deepening techniques rely on context similarity measures such as [12]. 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 needs huge quantities of training data that may be scarce and difficult to obtain. Unlike unsupervised methodologies, this kind of applications is limited to the existence of such data sets.

[4]With the cosine similarity measure.

Others [13] go even further by exploring heavy linguistic features combined with machine learning techniques to propose a new text similarity function. Once again it is a supervised approach and also heavily dependent on valuable linguistic resources which is 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.

Finally we address the work done by [18] that compared a set of evaluation metrics for the task of text-to-text generation. They compare the NIST simple string accuracy (SSA), the BLEU and NIST n-gram co-occurrence metrics, Melameds F measure, and latent semantic analysis (LSA). The comparison was done for fluency and adequacy of generated candidate sentences by comparison with one or more reference sentences. Conclusions claim that these automatic evaluation metrics are not adequate for the task of evaluating fluency, and are barely adequate for evaluating adequacy in the context of variation generation. These results imply that specific similarity metrics must be defined for specific tasks as general metrics are not reliable and may produce a great variety of results depending on the task they tackle.

III. Symmetrical Paraphrase Detection Functions (SP-functions) - Overviwe

In the literature [4], [14], [19], we can find the *Levenshtein Distance* [14] and what we call the *Word N-Gram Overlap Family* [4], [19]. Indeed in the latter case, some variations of word n-gram overlap functions are proposed but not clearly explained. In this section, we will review all the existing functions and also propose an enhanced n-gram overlap metric based on LCP (Longest Common Prefix) [20].

A. The Levenshtein Distance

This function, also known as *Edit Distance*, was created for string similarity computation. Considering two strings, the function computes the number of character insertions, deletions and substitutions that would be needed to transform one string into the opposite [14]. The function may be adapted for calculating *Sentence Edit Distance* - upon words instead of characters [7]. Considering two strings, it computes the number of words insertions, deletions and substitutions that would be needed to transform one sentence into the other one.

A problem that we observed, while using this function for paraphrases detection on text, was its failure on certain types of true paraphrases like the ones where there exists high lexical alternations or different syntactical structures. For example, it is unlikely that sentences (1) and (2) would be identified as paraphrases, since *Edit Distance* outputs an high value, erroneously indicating high dissimilarity among sentences.

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

(2) *Early morning, Prime Minister said that our GDP may continuing to fall, due to growing energy prices.*

This type of possible reordering is very unlikely to happen among words, with some lexical proximity, and *Edit Distant* was precisely conceived to be used with such raw material and not sentences, where more complex linguistic phenomena exists. Therefore it will fail in many positive examples like the sentence pair previously shown.

B. The Word N-Gram Family

In fact, we found not only one, but a set of text similarity measures based on word n-gram overlap in the literature. Sometimes it is not clear or unspecified which word n-gram version is used. In fact, two metrics are usually found in the literature (the Word Simple N-gram Overlap and the BLEU Metric). But, in order to be complete, we propose a third metric based on the LCP paradigm.

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

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

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

2) Exclusive LCP N-gram Overlap: In most work in Natural Language Processing, the longest a string is, the more meaningful it should be [21]. Based on this idea, we propose an extension of the word simple n-gram overlap function. The difference between simple and exclusive n-gram overlap lays on the fact that the exclusive form counts prefix overlapping 1-grams, 2-grams, 3-grams, ..., N-grams, regarding the Longest Common Prefix (LCP) paradigm proposed by [20]. For example, if some maximum overlapping 4-gram is found then its 3-grams, 2-grams and 1-grams prefixes will not be counted. Only the 4-gram and its suffixes will be taken into account. This is based on the idea that the longer the match the more significant the match will be. Therefore smaller matches are discarded.

In particular, we compute exclusive n-grams co-occurring in a sentence pair by using a suffix-array algorithm proposed by Yamamoto and Church [20]. They proposed this method to efficiently compute n-grams in a long corpus and calculate term frequencies and document frequencies.

So far, we never observed this n-gram overlap computing method. However, we decided to endeavor some experiments with this function as it is based on a sound hypothesis for Natural Language Processing applications. However, we will see in section **??** that the simple word n-gram overlap function gives overall better results although this enhanced n-gram overlap function shows interesting results to classify false paraphrases.

In order to clarify how do this word Longest Common Prefix be computed, we give the following example, with sentences (3) and (4) having some n-grams in common:

(3) *The President ordered the final strike over terrorists camp.*

(4) *President ordered the assault.*

Between these two sentences we have the LCP n-gram overlap given by: "President ordered the" which is a 3-gram. So the complete set of overlapping n-grams, besides the 3-gram, is: "ordered the" (2-gram) and "the" (1-gram), i.e all its suffixes.

If one wants to normalize the n-gram overlap then a particular difficulty rises, due to the LCP n-gram considerations, i.e. the maximum number of overlapping n-grams depends on the number of (n+1)-gram overlaps that exist. For example, in the previous case and for 1-grams, we only have one overlapping 1-gram ("the") between the two sentences and not 3 as it could be computed with the word simple n-gram overlap metric i.e. "the", "President" and "ordered". Thus, with this process of considering exclusive n-grams, it is unlikely to compute similarity based on a weighted sum like in formula 1. Another method, more suitable, is used and it is expressed by Equation 2

$$sim_{exo}(S_a, S_b) = \max_{\text{n}} \left\{ \frac{Count_{match}(\text{n-gram})}{Count(\text{n-gram})} \right\} \quad (2)$$

where S_a and S_b are two sentences and the following functions $Count_{match}(\text{n-gram})$ and $Count(\text{n-gram})$ are the same as above with this new matching strategy i.e. we first calculate $sim_{exo}(S_a, S_b)$ for 4-grams and then for the remaining 3-grams and so on and so forth, and then choose the maximum ratio.

3) The BLEU Function: The BLEU [19] metric were introduced as a function to automatically evaluate the performance achieved by a translation system and was employed in some conference contests to judge the quality of their competing systems. In such a competition any system must generate the translation to a given text supplied by the evaluator. Afterwards the evaluator evaluate the quality of the produced translation, by comparing it with a set of reference translations for that text, previously created by humans. This comparison is executed through the BLEU function, which are shown next in equation 7. First it is calculated the p_n value as follows:

$$p_n = \frac{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{ngram \in \mathcal{C}} Count_{clip}(ngram)}{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{ngram \in \mathcal{C}} Count(ngram)} \tag{3}$$

and then

$$BLEU = BP * \exp \sum_{n=1}^{N} w_n \log p_n \tag{4}$$

where BP is a brevity penalty factor and w_n are weighting factors ($w_n \in [0,1]$ and $\sum_{i=1}^{N} w_n = 1.0$). In equation 3 $\{Candidates\}$ is the reference translations set and the functions $Count_{clip}(ngram)$ counts the number of n-gram clippings (like matches) between the automatic generated translation and a given reference text. This means that this function count n-gram relation among two texts and as a consequence BLEU could be adapted to compute proximity among sentences. By doing this adaptation for sentence comparison, BLEU become $BLEU_{adapted}$, which is shown next:

$$BLEU_{adapted} = \exp \frac{1}{N} \sum_{n=1}^{N} log\ C_n \tag{5}$$

where C_n is computed as follows:

$$C_n = \sum_{\text{ngram}} \frac{Count_{match}(\text{ngram })}{Count(\text{ngram })} \tag{6}$$

For the brevity penalty factor we chose $BP = exp(1 - \frac{r}{c})$, where r and s are respectively the greatest and smallest sentence lengths, in terms of word counting. For the weighting factors, we took $w_n = \frac{1}{N}$ and end up with nearly the geometrical mean of the C_n ratios:

$$BLEU_{adapted} = BP * \left[\prod_{n=1}^{N} C_n \right]^{\frac{1}{N}} \tag{7}$$

The $Count_{match}(\text{ngram})$ function counts the number of exclusive or no-exclusive n-grams co-occurring between the two sentences, and the function $Count(\text{ngram})$ the maximum number of n-grams that exists in the shorter sentence. The maximum number of n-grams (N) was chosen equal to 4 in our experiments, like in many other domains where BLEU is employed.

IV. Asymmetrical Paraphrase Detection Functions (AP-functions)

Paraphrase corpora are gold resources in many research fields [2]–[5] and for us it will be used for *sentence compression* rule induction, using machine learning techniques. Our main research area lays in the field of *Automatic Sentence Compression* and we see paraphrase clusters as nice raw material to discover Sentence Compression patterns as other authors pointed out [2]–[5], [10]. Therefore, we intend to automatically construct a huge paraphrase corpus.

Automatic Paraphrase corpus construction has already be addressed by some authors [4], [7], by using some of the functions presented previously in section III. However, the application of the existing functions for paraphrase detection in real-world conditions, like in Web news stories, reveal a particular difficulty: most of the results are exact or quasi-exact match pairs of sentences. Such results are obviously useless for us, since we are specially interested on asymmetric paraphrases, where one sentence contains more information, possibly irrelevant, than the other one. Therefore we started to think and design about a new function capable to avoid this difficulty and to maintain some desirable properties like those pointed next:

(1) Achieve maximum automation in corpus construction - minimum or even no human intervention, with high reliability.

(2) Penalize equal and almost equal sentences - they are not useful for our research needs, but frequent in real-world news stories situations.

(3) Ability to identify pairs having a high degree of lexical reordering, and different syntactic structure

(4) Define a computationally fast and well founded metric, to extract a great amount of paraphrases within relative low computation cost and time.

A. The LogSim family

This was our first attempt to create a metric to answer to our special needs of asymmetrical paraphrase identification, having the desired properties we have just described. The basic idea of the *LogSim* family lays on the notion of exclusive lexical links between pairs of sentences, as shown in figure 1. This metric may be "expanded" to integrate more complex features, like syntactic or semantic labels, but at a first step we preferred to maintain it at a lexical level, and so preserving language independency. In particular, we will show in section VI that the results obtained with this level of complexity will be difficult to improve by the integration of any other syntactic or semantic features, at least for this kind of corpora.

An exclusive lexical-link defines an 1-gram, or unigram, exclusive overlap: if a link is established between sentence S_a and sentence S_b, for the word $\mathbf{w}$, then other occurrences of word $\mathbf{w}$ in sentence S_a will engage a new link to sentence S_b if there exists at least one more occurrence of $\mathbf{w}$ in S_b, besides the one which is already connected.

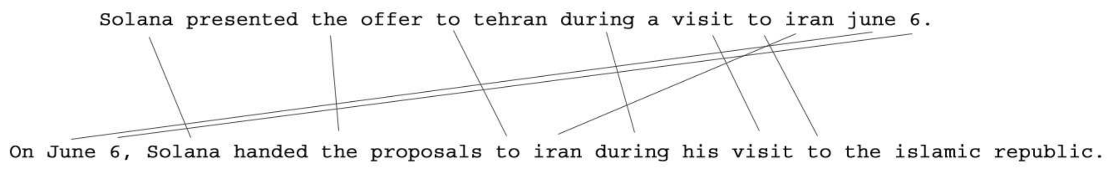
Solana presented the offer to tehran during a visit to iran june 6.
On June 6, Solana handed the proposals to iran during his visit to the islamic republic.

Figure 1. Links between a sentence pair.

B. LogSim

We define the number of links between 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 connection among sentences. As $\frac{\lambda}{x} \longrightarrow 1$, it becomes more likely that both sentences are equal, and with $\frac{\lambda}{x} = 1$, they are in fact exactly equal. Remark that even if the shortest sentence is strictly contained inside the longest one, we have $\frac{\lambda}{x} < 1$

To calculate the *LogSim* function, $LogSim(.,.)$, we first evaluate the function $L(x,\lambda)$ as in Equation 8[5]

$$L(x,\lambda) = -\log_2(\frac{\lambda}{x}) \tag{8}$$

Then the $LogSim(.,.)$ is obtained as in Equation 9 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 on those situations where we have an exact match.

$$LogSim(S_a,S_b) = \begin{cases} L(x,\lambda) & if \;\; L(x,\lambda) < 1.0 \\ e^{-k*L(x,\lambda)} & otherwise \end{cases} \tag{9}$$

The main objective of the second branch $e^{-k*L(x,\lambda)}$ is to dramatically 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 $LogSim(.,.)$ output value is repositioned in $[0,1]$ as a small value (0.00039). The positive k parameter is used to boost the penalization branch. In our experiments, we decided to fixed $k = 2.7$. In fact, the greater the k, the greater the penalization will be for dissimilar pairs.

C. Expanded LogSim

The $L(x,\lambda)$ function, defined in the previous subsection is independent from the number of words in the shorter sentence y. This is not absolutely 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 10:

$$L(x,y,\lambda) = -\alpha * L(x,\lambda) - \beta * L(y,\lambda) \tag{10}$$

where α and β are weighting factors, such that $\alpha \in [0,1]$ and $\alpha = 1 - \beta$. Remark that by varying the α weight we obtain different $L(x,y,\lambda)$ functions, in particular for $\alpha = 1$ we have $L(x,y,\lambda) = L(x,\lambda)$.

In a same thought as for the $LogSim(.,.)$ function, and in order to ensure that the function range lays in the interval $[0,1]$, we define the $LogSimX(.,.)$ as shown next in equation 11.

$$LogSimX(S_a,S_b) = \begin{cases} L(x,y,\lambda) & if \; L(x,y,\lambda) < 1.0 \\ e^{-k*L(x,y,\lambda)} & otherwise \end{cases} \tag{11}$$

D. 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, like in "n-gram overlap" or BLEU metrics. 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]When $\lambda = 0$, $L(x,\lambda) = 0$.

E. Beyond LogSim - A Set of Convenient Mathematical Functions

We are specially interested in asymmetrical paraphrases as previously mentioned, since they complain better with the key idea of compactness which is rather relevant in *Automatic Sentence Compression*. Such a collection enables, for instance, a *machine learning* based system to automatically induce sentence simplification rules. However, as expressed previously in section I, the known functions used so far for paraphrases identification fail to correctly identify asymmetrical pairs. Therefore, at a first step a new function was specially designed ($LogSim$ and $LogSimX$) to overcome the difficulties revealed by the existing functions.

However, despite the superior performance achieved by LogSimX (see the VI section), both in symmetrical and asymmetrical corpora, the function seems a bit tricky and difficult to explain and even not very mathematically sound, having an inelegant discontinuity point that is well noticed in the graph of figure 2.

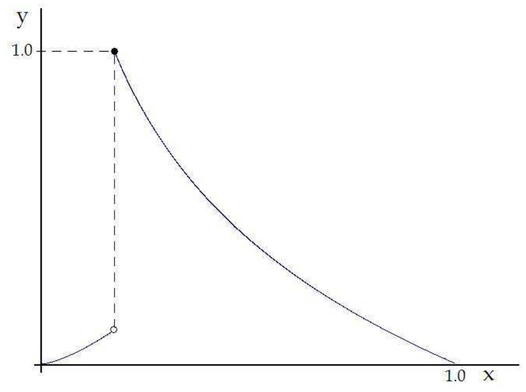

Figure 2. The LogSim(x) function in the $[0, 1]$ interval.

Therefore we thought about other functions that could well model our problem yet being more mathematically sound. We started by looking at the main characteristics that are encoded inside $LogSimX$ and try to find out other, and more simple, mathematical functions which contains these characteristics, hoping that such functions will behave as well as $LogSimX$. As a result we found a set of well known functions that models very well (see the obtained results, in section VI) this phenomena.

Such a function, lets say $paraph(S_a, S_b)$, should take two sentences as input and return a number in the $[0, 1]$ interval, meaning paraphrase relatedness among these sentences, or in a more probabilistic interpretation the probability for those sentences to form an asymmetrical paraphrase pair. The $paraph(.,.)$ function should return low values, whether the two sentences are too different or almost equal, like in the following example:

S_a**:** *The stock has gained 9.6 percent this year..*

S_b**:** *The stock has gained 9.6% this year.*

Having this example in mind, it seems clear that a much greater value should be returned (possibly near 1.0) for the "control panel" example shown in section I. Given these insights, we devised for $paraph(.,.)$ a mathematical function with a curve similar to one of those shown in figure 5.

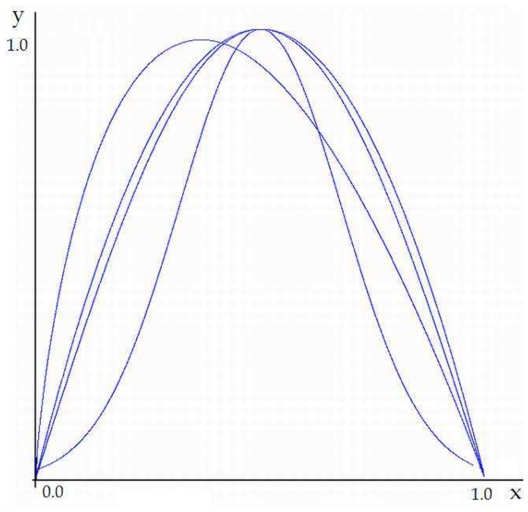

Figure 3. Candidate models for $paraph(.,.)$ function.

These kind of "hill curves" share all the same property of zero approximation, on their boundaries, i.e: $\lim_{x\to 0} f(x) = 0$ and $\lim_{x\to 1} f(x) = 0$, and also there exist an $x_{max} \in [0, 1]$ such that the function reach its maximum. Ideally we seek functions satisfying the condition $f(x_{max}) = 1.0$. In our problem, the x value will be a ratio representing some combinations of features, among sentences, for example the number of common words divided by the number of words in each sentence.

We experimented at least 4 different types of mathematical functions that satisfy the previous conditions, for convenience we named this four types as: *parabolic*, *trigonometric*, *gaussian* and *entropic* functions. The remainder of this section will explain the details of each one and what particular form[6], from each type, do we use. The comparative results shown in section VI evidences high performance achievement for almost all these new functions, when compared with the already existing SP functions, specially for asymmetrical paraphrase identification task.

F. Parabolic and Trigonometric Functions

A parabolic function is defined by a second order polynom, i.e: $f(x) = ax^2 + bx + c$, which depends upon parameters a, b and c. These parameters determines the function zeros and the concavity direction - upward or downward. For our problem, we took $a = -4$, $b = 4$ and $c = 0$: $f(x) = 4x - 4x^2$, where p was taken equal to $\frac{\lambda}{|S_a|} * \frac{\lambda}{|S_b|}$, and $|S_a|$ and $|S_b|$ are the number of words in the sentences.

Among the trigonometric functions we experimented a transformation of the $\sin(x)$ function which have a

[6]The form depends on a set of parameters chosen and bounded to some values.

maximum value equal to 1 in the interval $[0, \pi]$ for exactly $x = \frac{\pi}{2}$. Since our domain lays in the $[0, 1]$ interval, a linear transformation of the function domain was taken and our trigonometric function become equal to $\sin(\pi x)$. Once again we calculate x by multiplying the two ratios: $\frac{\lambda}{|S_a|} * \frac{\lambda}{|S_b|}$.

G. Gaussian Functions

Gaussian functions are widely known and used in many scientific and engineering domains, for instance the *Normal Distribution* used in *Probability* and *Statistic* is a particular case of a Gaussian function. Such a function is defined as follows:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{12}$$

The parameters a, b and c shapes different functions but with the same symmetric "bell shape" aspect, as it can be seen in equation 12, where three different Gaussian functions are drawn for three different c values ($c = 0.16, 0.20, 0.25$). In order to have $f(x_{max}) = 1.0$, we must take $a = 1$, and also another property inherent to gaussian functions is that the maximum value is reached at point b ($x_{max} = b$), in our experiments we fixed $b = 0.5$.

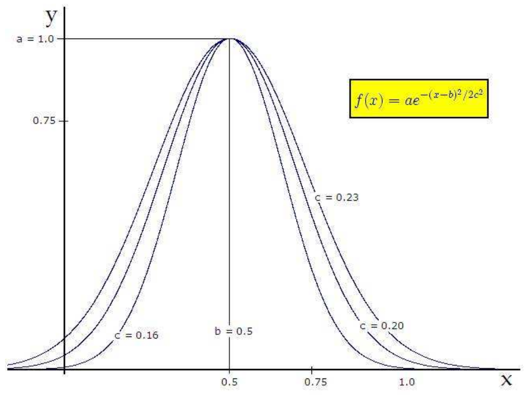

Figure 4. Three Gaussian functions for three different c parameters.

The c parameter governs the function values on the boundaries (0, 1), greater values outputs bigger boundary values, meaning less penalization for the dissimilarity (0) or too high-similarity (1.0) between the sentence pair. For example, by drawing the graphic with $c = 0.23$, in figure 4, we obtain $f(0) = 0.1$, meaning that the function will output a relatedness of 0.1, even for a sentence pair without any overlapping feature.

H. Entropic Functions

The *entropy* is a key concept in *Thermodynamics*, *Quantum Physics* and in particular in *Information Theory* it is known as the *Shannon Entropy*. This, also called information entropy, is a measure of the uncertainty associated with a random variable and it also quantifies information in a piece of data. Entropy is also the shortest average message length, in bits, that can be sent to communicate the true value of the random variable to a recipient. Considering our sentence pair (S_a, S_b) and the number of exclusive links in between λ, we have the two already mentioned ratios $\frac{\lambda}{|S_a|}$ and $\frac{\lambda}{|S_b|}$ and combine them like: $p = \frac{\lambda}{|S_a|} * \frac{\lambda}{|S_b|}$. We obtain p as the value of a random variable in the $[0, 1]$ interval, which may be interpreted as two random variables. For a given binary random variable with probability p, for a given outcome[7], the entropy function computes as follows:

$$E(X) = -p * log_2(p) - (1-p) * log_2(1-p) \tag{13}$$

This function reach the maximum value of 1.0 for $p = 0.5$ meaning that the uncertainty is maximum and if p is near 0 or 1 then $E(X) \rightarrow 0$, meaning that we are almost certain about the outcome.

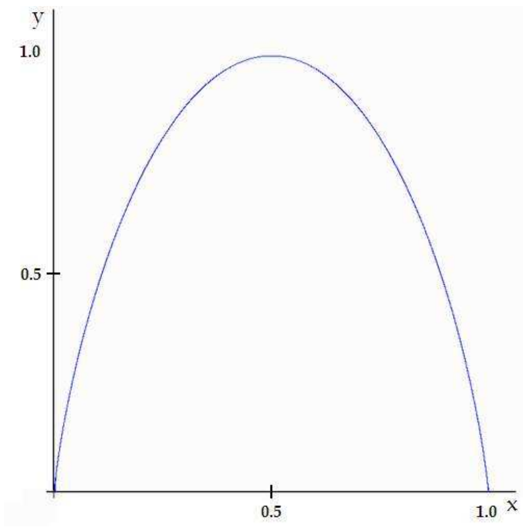

Figure 5. The Entropy Function.

This function contains all the desired properties, given in subsection IV-E and in our sentence comparison task the extreme values contains different meanings. In our problem, "maximum uncertainty" at 0.5 mean "desired relatedness" among sentences and the low values for 0 and 1, mean precisely "undesirable relatedness" - sentences almost equal ($p = 1.0$) or sentences too dissimilar ($p = 0.0$).

V. 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 in the Sentence Compression research field, like [2], [5]. By adapting these corpora we created three new corpora to serve as a benchmark for our specific purpose. We will explain the details of each corpus in the next subsections, so that the reader may have a more clear compression about the comparative experimentations that we have done and the subsequent results obtained and reported in section VI

[7]Which is our case, since the sentence pair is classified as paraphrase or not.

A. The Microsoft Paraphrase Corpus

In 2005, Microsoft researchers [7] published the first freely available paraphrase corpus containing a total of 5801 pairs of sentences, where 3900 was annotated as "semantically equivalent" or true paraphrases and the remaining 1901 with the contrary. In this article, we will refer to this corpus as $\{MSRPC\}$.

To identify and extract paraphrases from news texts, two techniques were used: the *String Edit Distance* (see section III-A) and an heuristic that pairs initial sentences from different news stories as paraphrases. Thus for this corpus construction, sentences were extracted from massive parallel news sources and tagged by 3 human raters according to certain guidelines described in [7]. The tagging guidelines established a set of definitions to help human judges to identify which pairs should be considered "equivalent" (paraphrases) or "not equivalent" pairs. To identify sentence pairs with "different" contents, these guidelines tackled issues like:

1) Different content: prototypical example,
2) Shared content of the same event, but lacking details,
3) Cannot determine if sentences refer to the same event,
4) Shared content but different rhetorical structure,
5) Same event but details different emphasis.

These definitions and guidelines where oriented toward the extraction of symmetrical paraphrase pairs and therefore does not absolutely comply with our broader view of the kinds of paraphrases that may exist. We are specially interested in asymmetrical paraphrases and some of the guidelines, for example the number 5 from the previous list, followed for the $\{MSRPC\}$ corpus construction, conflict out special paraphrase search. In fact, sentence pairs satisfying this condition, are generally very interesting in *Sentence Compression*, because they carry some sentence reduction information. For example, sentences (5) and (6) are clearly interesting paraphrases to our research, although they were not classified as such by regarding the guidelines, but as a negative example. A negative example is a pair of sentences, from an annotated paraphrase corpus, that is marked with some label meaning: "These two sentences are not paraphrases from each other".

> (5) *Researchers have identified a genetic pilot light for puberty in both mice and humans.*
>
> (6) *The discovery of a gene that appears to be a key regulator of puberty in humans and mice could lead to new infertility treatments and contraceptives.*

Hence, we expect that our proposed AP functions, reach a certain amount of disagreement over some $\{MSRPC\}$ negative examples, by identifying these examples as positive (asymmetrical paraphrases). In terms of evaluation, this will increase the number of false positives, classified by our AP functions, as it is experimentally confirmed (see the results, especially table II in section VI-B.

B. The Knight and Marcu Corpus.

The corpus used by [2] in their *sentence compression* research work, contains 1087 sentence pairs, where one sentence is a compressed or summarized version of the other one. Here, we labeled this corpus as $\{KMC\}$. It was created in a completely manual way, from pairs of texts and respective summaries and it is well tailored for *sentence compression* research, for example to train *supervised learning algorithms*. However, constructing such a corpus could be very laborious and time consuming and may even be insufficient in terms of linguistic diversity. In fact, these were the main reasons that lead us to propose new methodologies for automatically paraphrase corpus construction.

C. The Corpora Used for Evaluation

One major limitation with the $\{KMC\}$ corpus is that it only contains positive examples and therefore it should not be taken as such to perform any evaluation. Indeed, it is necessary to add an equal number of negative examples to obtain fair evaluations, among the paraphrase detection functions. Even the $\{MSRPC\}$ corpus is fairly unbalanced, having only 1901 negative examples against the 3900 positives. To perform an equitable evaluation, we decided to expand both corpora by adding negative examples, randomly selected from *Web News Texts*, in a sufficient number to balance the corpora, so that they have the same number of positive and negative examples. Finally we also decided to create a third corpus, which is a mixture of $\{MSRPC\}$ and $\{KMC\}$ corpora. These three adapted corpora are briefly described next but before we show an illustrations of what a "random selected negative pair" could be, to clarify a bit more the reader:

> *(7) Running back Julius Jones is expected to return after missing the last three games with a sprained left ankle.*
>
> *(8) Economists closely watch the performance of consumer spending since it accounts for two-thirds of total economic activity.*

As we can see, the two sentences are complectly different from each other. However, as the sentence repetition rate is high among related news texts, it is also probable to insert as a negative example a sentence pair equal or almost equal, like the one shown in the next example:

> *(9) "I don't think my age matters during competitions", said Nadal.*
>
> *(10) I don't think my age matters during the competitions, said Nadal.*

In subsection VI-D, we present a discussion and some evaluation about this issue of adding random pairs as negative examples, where some of them are quasi-equal sentences, yet they are inserted as negative pairs. We may see there that this do not compromise the evaluation method and that the difference among SP and AP type functions is absolutely independent from these negative random pairs.

1) 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 examples (symbolized by X^{-}_{1999}), selected from web news stories. So we end with 3900 positive pairs and 3900 negative ones.

2) 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 examples (1087 in this case).

3) The $\{MSRPC^{+} \cup KMC \cup X^{-}_{4987}\}$ *Corpus:* Finally we decided to build a bigger corpus that gathers the positive $\{MSRPC\}$ part, with its 3900 positive examples, and the 1087 positive pairs of sentences from the $\{KMC\}$ corpus, giving a total of 4987 positive pairs. To balance the corpus an equal number of negative pairs were added, obtained in a same fashion as described previously for the other corpora. We labeled this wider corpus with the label $\{MSRPC^{+} \cup KMC \cup X^{-}_{4987}\}$. In this corpus we exclude the $\{MSRPC\}$ negative pairs, due to what were discussed in the previous subsection V-A.

These corpora are available on our web site[8], without the $\{MSRPC\}$ corpus neither the $\{KMC\}$, since the first one is already available online[9], and the second one is not publicly available, but was directly provided to us by the [2] authorsh. Thus, we only provide the negative examples that we used: the $\{X^{-}_{1999}\}$, the $\{X^{-}_{1087}\}$, and the $\{X^{-}_{4987}\}$, enabling every one to exactly reconstruct each corpus and make comparative experiments.

VI. Results

In this work we made a meticulous investigation about paraphrase identification functions. A set of already known functions were tested and new functions are proposed. Subsequently an experimental comparative study was accomplished, between all the 9 functions, on the 3 corpora described in section V. The results obtained will are presented here, in the next subsections and main commentaries and conclusions are kept for section VII.

A. How to Classify a Paraphrase?

Before presenting the results, it is is necessary to talk about a classical difficulty that is inherent to every classification problem - *thresholds*. Usually, for a given classification problem, a system takes decisions upon some parameters, which are called thresholds and are somehow previously set by someone: the user, the programmer or others. In our paraphrase identification or classification problem, every function evaluation is dependent from a given predefined threshold θ - the "frontier value" upon which is taken the classification decision of "paraphrase" or "not paraphrase". For example, assuming that we are evaluating function $paraph(.,.)$, that returns the likelihood for any two sentences S_a and S_b, from then corpus, being paraphrases, and also presuming that we chose $\theta = 0.6$, then the pair $\langle S_a, S_b \rangle$ will be classified as "paraphrase" if $paraph(S_a, S_b) > 0.6$ or else as "not paraphrase".

Thresholds are parameters that unease the process of fairly evaluation. Indeed, the best parameter should be determined for each function, however this is not always the case and wrong evaluations are often proposed. In our evaluation process, we do not pre-define any threshold for any function but let the evaluator to automatically compute the best threshold for each function, i.e the threshold that maximizes the function performance in the corpora.

This computation is a classical problem of function maximization or optimization [22]. In particular, we use the bisection strategy as it computes fast, and well approximates the global maximum of the smooth function performance curve. As a result, we are optimizing the value of the threshold for each metric in the same way and do not introduce any subjectivity in the choice of the parameters.

In Table I, we present the obtained thresholds for the nine compared metrics using a *10-fold cross validation* scheme. The results show that the bisection strategy performs well for our task as the standard deviation for each function and corpus is almost negligible.

In this section we renamed $\{MSRPC \cup X^{-}_{1999}\}$ as **A**, $\{KMC \cup X^{-}_{1087}\}$ as **B** and $\{MSRPC^{+} \cup KMC \cup X^{-}_{4987}\}$ as **C** in order to ease the representation and reading, in the tables.

B. Experiments and Results

In order to evaluate and compare the results of each function over each corpus, we calculate the known *F-Measure* and *Accuracy* values. These performance values are calculated according equations 14 and 16. In particular, for *F-Measure* we took $\beta = 1$, meaning that precision and recall are equally weighted.

Every result shown in tables II and III was calculated by averaging the 10 *F-Measure* and *Accuracy* values obtained from a *10-fold cross validation* test, performed over the data. For every fold, the best threshold was found on the $\frac{9}{10}$ training data, as described previously, and then it is used on the $\frac{1}{10}$ test fold to calculate its *F-Measure* and *Accuracy* performance.

In the tables we relabeled some function names in order to have a better and compact representation, thus $Edit$, *Bleu*, *Trignom*, and *Entropy* are respectively the *Levenshtein Distance*, the $Bleu_{adapted}$, the trigonometric $sin(x\pi)$, and the entropy $E(p)$ function. For details see sections III and IV.

[8] `http://competence.di.ubi.pt/`
[9] `http://research.microsoft.com/nlp/`

TABLE I.
THRESHOLDS MEAN AND STANDARD DEVIATION

thresholds	A	B	C
Edit	17.222 ± 0.1109	20.167 ± 1.3751	17.312 ± 0.0000
Sim_o	0.2030 ± 0.0068	0.2604 ± 0.0026	0.2537 ± 0.0000
Sim_{exo}	0.5006 ± 0.0000	0.7250 ± 0.0130	0.5005 ± 0.0000
Bleu	0.5024 ± 0.0003	0.0083 ± 0.0071	0.5025 ± 0.0000
LogSimX	0.0765 ± 0.0035	0.0053 ± 0.0006	0.0069 ± 0.0000
Trignom	0.0610 ± 0.0200	0.2845 ± 0.0012	0.3470 ± 0.0000
Parabolic	0.4934 ± 0.0035	0.3000 ± 0.0012	0.3770 ± 0.0095
Entropy	0.4895 ± 0.0061	0.4060 ± 0.0070	0.4685 ± 0.0047
Gaussian	0.4781 ± 0.0155	0.3795 ± 0.0043	0.4102 ± 0.0000

TABLE II.
$F_{measure}$ OBTAINED.

type		A	B	C
AP	Edit	74.39%	71.04%	80.98%
	Sim_o	78.79%	94.45%	91.14%
	Sim_{exo}	77.87%	90.91%	87.17%
	Bleu	76.62%	69.32%	84.68%
SP	LogSimX	**80.94%**	**98.46%**	98.54%
	Trignom	79.69%	61.52%	88.86%
	Parabolic	80.24%	97.56%	98.49%
	Entropy	80.12%	97.46%	98.52%
	Gaussian	80.17%	97.51%	**98.56%**

TABLE III.
Accuracy OBTAINED.

type		A	B	C
AP	Edit	67.68%	68.61%	79.02%
	Sim_o	73.40%	95.20%	91.07%
	Sim_{exo}	72.37%	90.37%	86.00%
	Bleu	69.51%	57.05%	83.63%
SP	LogSimX	**78.20%**	**98.43%**	98.53%
	Trignom	66.36%	69.44%	89.54%
	Parabolic	75.78%	97.58%	98.50%
	Entropy	75.38%	97.51%	98.53%
	Gaussian	75.59%	97.51%	**98.57%**

$$F_\beta = \frac{(1+\beta^2) * precision * recall}{\beta^2 * precision + recall} \tag{14}$$

with

$$precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN} \tag{15}$$

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

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{16}$$

The results shown in Table II evidences that, in general, SP-functions outperforms the AP-functions over all corpora, except for the trigonometric $sin(x\pi)$ function, which do not behaves very well and even worse than AP-functions, on symmetrical corpora **B** and **C**. For instance, on the biggest corpus (**C**), the SF-functions (excluding the trigonometric case) correctly classified, on average, 98.53% from all (9974) sentence pairs, whether positive or negative, remark the accuracy table III, where correct positive and negative classifications are counted.

As we expected, the SP-functions performed better on corpora containing symmetrical pairs, like **B** and **C**, revealing an *Accuracy* difference of 13.05%, by considering the average performance of each type (AP and SP) on both corpora and a correspondent $F_{measure}$ difference of 9.84%. However, even for the symmetrical corpus **A**, the SP-functions performance are slightly better than the AP-functions, revealing an average differential of 3.31% in terms of $F_{measure}$ and about 3.52% in terms of *Accuracy*.

Another interesting result is the fact that all metrics behave the same way over all corpora, although the $Gaussian$ function achieved a better result in the **C** corpus than $LogSimX$, contrary to what happens for **A** and **B** corpora, however, the difference is negligible (about 0.04%).

Inside the AP functions set, the simple word n-gram overlap (sim_o) and the exclusive lcp n-gram overlap (sim_{exo}) metrics always get first and second places, respectively and the BLEU metric and the Edit Distance obtain the worst results over all corpora. For the SP functions, and in general, our earlier proposed LogSimX function obtains better results, however the $Gaussian$ function has a very close performance, surpassing the former in corpus **C**. Another remark for this SP type functions is that the performance difference among the functions is very small, except for the trigonometric function.

C. The (AP type) Bleu Functions - A Special Case

By looking at the BLEU function, on subsection III-B.3, it is clear that this function should indeed be interpreted as a set of functions that depends from the sum maximum limit (N) chosen, which were equal to 4 in our case, following some literature indications, as for example [4], which proposes to use $N = 4$ for overlap n-grams, i.e. they consider n-grams with length 1, 2, 3 and 4. By choosing different values for the N limit, one may questioning if there exist significant performance difference. We experienced that for values greater than 4, the performance starts to decay rapidly, and the reason for this is that, as we have a product of N factors (see equation 7), it is enough to have only one factor with low performance to affect the whole result. And when $N \to 1.0$ the performance tend to improve as shown in table IV.

TABLE IV.
THE $BLEU$ RESULTS, AS $N \rightarrow 1.0$

F_β	A	B	C
N=4	76.62%	69.32%	84.68%
N=3	77.82%	69.33%	87.86%
N=2	78.77%	68.88%	90.18%
N=1	79.39%	77.45%	91.15%

Remark that the function will become more inefficient (more computational complexity) as N grows, and the results evidences that this would be a computational worthless effort. The most efficient BLUE function will be the simplest one, when $N = 1$, i.e only simple unigram lexical links are counted, like in our proposed SP functions that relies only on the number of words from each sentence and the number of exclusive lexical links, as it was illustrated in figure 1.

D. The Influence of Random Negative Pairs

In section V-C we described a method to artificially add sentence pairs, randomly picked from *Web News Texts* as negative paraphrase pairs, to balance [10] the corpora. We also mentioned that it is likely that quasi-equal pairs could be inserted too, as negative examples. Certainly that this option may be criticized, at least due to these "quasi-equal" pairs, since they may be interpreted as asymmetrical negative pairs, precisely where the SP functions succeed and AP functions fail. This particular type of examples tend to be counted as *false positives* for AP functions and as TN for the SP functions. Therefore one may say that the evaluation is biased toward SP functions, however, this is not the case as we will see here. Indeed, to acknowledge this situation, we performed another experiment with a corpus similar to the **C** corpus (the biggest one) but without any quasi-equal or equal pair. We named this new corpus as **C'**. The performance obtained over the **C'** is illustrated in Table VI and clearly shows that our SP functions continue to outperforms AP-function in all evaluation situations.

TABLE V.
AP-FUNCTIONS ON A CORPUS WITHOUT QUASI-EQUAL OR EQUAL PAIRS

Accuracy %	edit	sim_o	sim_{exo}	bleu
C'	84.31	96.36	90.19	87.57

TABLE VI.
SP-FUNCTIONS ON A CORPUS WITHOUT QUASI-EQUAL OR EQUAL PAIRS (ACCURACY %)

LogsimX	Trignom	Parabolic	Entropy	Gauss
99.58	89.30	99.24	99.10	99.21

In this case, we only show the Accuracy measure as the F-measure evidences similar results. This give us at least 99% statistical confidence[11] (1% significance) that $Accuracy_{(SP-functions)} > Accuracy_{(AP-functions)}$.

VII. CONCLUSIONS

In this paper, we proposed a new set of functions, specially thought to identify asymmetrical[12] paraphrases in text - the AP functions. In terms of overlapping features, these functions reject all pairs that are whether too dissimilar or too similar. Beyond an earlier AP function proposition - the LogSimX - we also investigated a set of well defined mathematical functions, having the desired properties (see section IV) for symmetrical pairs identification, and performed a comparative study among all functions, on three different corpora. The first corpus (**A**) contains almost asymmetrical pairs, in the second one (**B**) only symmetrical pairs are present, as positive examples, and in the third corpus (**C**) we have a mix of symmetrical and asymmetrical pairs[13].

The experimental results obtained confirm our initial intuition that these new functions, oriented toward asymmetrical pair identification, achieves better results than previous already known and used functions for paraphrase identification, named here as SP functions, like the *Word n-gram Overlap*, and *BLEU* functions. It is even more interesting that the difference is also noticed when testing on a corpus with almost symmetrical examples, like corpus **A**, being the difference at least 3% on average. Therefore we conclude that our proposed SP functions are also suitable for symmetrical paraphrase identification. We would like also to remark here that the *Word Edit Distance* is the worst function for doing this job and we think that the reason for that is related with the many ways that a given information could be represented through a sentence, for example by choosing different syntactical structures, passive forms, alternations, etc. This phenomena is also related with the low performance achieved by the BLEU function for greater N (maximum number of n-grams considered) values, as discussed and demonstrated in subsection VI-C. It seems that considering as features long (≥ 2) n-grams tend to degrade the performance, and so it seems better to only consider lexical connections (by counting unigrams), which were counted exclusively, in our experiments. For instance this will be immune to the alternation possibilities among paraphrases, and even it will be more easily computed and so more efficient.

In the future we will try to include tf.idf [23] and *Part of Speech Tagging information* (POS), as input features for our functions, as we believe that word links between sentences should have distinct weights. Indeed, it is different to have a match between determinants or between verbs or names. Verbs and Determinants obviously convey relevant information about the sentence while it is not the case for determinants. We may also integrate the notion of content n-grams that can be extracted from monolingual corpora

[10]Equal number of positive and negative examples.

[11]By making a proportion statistical test for the accuracies: $H_0 : p_1 = p_2$ against $H_1 : p_1 > p_2$.

[12]When only one sentence entails the other one.

[13]This corpora labeling (A,B,C) is related to section VI

as in [21]. Finally, [4] propose a clustering methodology to group similar sentences (i.e. paraphrases) into clusters. We will use our metric to perform a similar task and compare the results.

ACKNOWLEDGMENT

We would like to thank Daniel Marcu for providing us with his corpus of paraphrases.

REFERENCES

[1] 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*, pp. 178–185, 2000.

[2] K. Knight and D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression." *Artificial Intelligence*, pp. 139(1):91–107, 2002.

[3] K. S. Y. Shinyama, S. Sekine and R. Grishman, "Automatic paraphrase acquisition from news articles," *In Proceedings of Human Language Technology (HLT 2002)*, 2002.

[4] R. Barzilay and L. Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment." *In Proceedings of HLT-NAACL.*, 2003.

[5] A. S. M. Le Nguyen, S. Horiguchi and B. T. Ho, "Example-based sentence reduction using the hidden markov model," *ACM Transactions on Asian Language Information Processing (TALIP)*, pp. 3(2):146–158, 2004.

[6] E. Marsi and E. Krahmer, "Explorations in sentence fusion," *In Proceedings of the 10th European Workshop on Natural Language Generation*, 2005.

[7] C. Q. W.B Dolan 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)*, 2004.

[8] R. Chandrasekar and B. Srinivas, "Automatic induction of rules for text simplification," *Knowledge-Based Systems*, pp. 10:183–190, 1997. [Online]. Available: citeseer.ist.psu.edu/article/chandrasekar97automatic.html

[9] D. P. Y. C. S. D. J. Carroll, G. Minnen 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)*, 1999.

[10] 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*, 1998.

[11] M. L. P. T. Structuring, "Experiments with sentence ordering," *In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2003.

[12] R. Barzilay and N. Elhadad, "Sentence alignment for monolingual comparable corpora." *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*, pp. 25–33, 2003.

[13] J. K. V. Hatzivassiloglou 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)*, 1999.

[14] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals." *Soviet Physice-Doklady*, pp. 10:707–710, 1966.

[15] F. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, pp. 29(1):19–51, 2003.

[16] J. Sjöbergh and K. Araki, "Extraction based summarization usinga shortest path algorithm," *In Proceedings of 12th Annual Language Processing Conference (NLP 2006)*, 2006.

[17] A. L. L.V. Lita, M. Rogati.

[18] M. M. A. Stent and M. Singhai.

[19] T. W. W.-J. Z. K. Papineni, S. Roukos, "Bleu: a method for automatic evaluation of machine translation," *IBM Research Report RC22176*, 2001.

[20] M. Yamamoto and K. Church, "Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus," *Computational Linguistics*, pp. 27(1):1–30, 2001.

[21] S. G. G. Dias and J. 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*, pp. 213–221, 2000.

[22] E. Polak, "Computational methods in optimization," *New York Academic Press*, 1971.

[23] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management*, pp. 24(5):513–523, 1988.

João Cordeiro is currently a Ph.D. candidate at the University of Beira Interior in Covilhã, Portugal. He received his MS degree in Artificial Intelligence and Computation from University of Porto, Portugal in 2003 and his BS degree in Matemathics/Informatics from University of Beira Interior, Covilhã, Portugal, in 1998. Since November of 1999, he is a lecturer at the University of Beira Interior, Covilhã, Portugal and he worked as a Software Engineering, in the Software Industry, at Lisbon, between 1997 and 1999. He is a member of the Association for Computational Linguistics (ACL) and a member of the Portuguese Association of Artificial Intelligence (APPIA). His research interests include Automatic Text Summarization, Text Mining, Information Extraction, Information Retrieval, Web Search and Machine Learning.

Geël Dias received his PhD in Computer Science in the specific area of Natural Language Processing from the New University of Lisbon (Portugal) in collaboration with the Laboratoire d'Informatique Fondamentale d'Orléans (France). He received a Master Degree in Computer Science in the specific field of Languages, Programming, and Translation (DEA Langages, Programmation et Traduction) from the Laboratoire d'Informatique Fondamentale d'Orléans (LIFO) in France. He graduated from the University of Orléans (France) in Computer Science and Management (Maîtrise d'Informatique Appliquée à la Gestion). He is Assistant Professor of the Department of Computer Science of the University of Beira Interior where he teaches all subjects related to Artificial Intelligence and a part of System Analysis. He is also the coordinator of the professional activities of the course on Software and Systems (sponsored by Microsoft) of the Department of Computer Science. He is a researcher at the Centre of Mathematics of the University of Beira Interior and Invited Researcher at the LF Group of INESC-ID in Lisbon (Portugal). He has the Erdo's Number 5 classification (Erdo's number project). He is also responsible for the Centre of Human Language Technology and Bioinformatics (HULTIG) of the University of Beira Interior which gathers twenty members. He is a member of the Association for Computational Linguistics (ACL), a member of the French Association for Natural Language Processing (ATALA) and a member of the Portuguese Association of Artificial Intelligence (APPIA). His research interests include Multiword Unit Extraction, Topic Segmentation, Text Summarization, Sentence Compression, Word Sense

Disambiguation, Ontology, Web Search, Medical Lexicons, and Text Mining.

Pavel Brazdil holds a Post-Doctoral habilitation at the Faculty of Economics of Univ. of Porto, Portugal, since 1996, and a Ph.D. in Artificial Intelligence at the University of Edinburgh, Great Britain, since 1981. He received his BS and MS degree in Electrical Engineering at VUT, Brno, Czech Republic, in 1968. He is currently a Full Professor at the Faculty of Economics (FEP) of University of Porto, Portugal and the Coordinator of Artificial Intelligence and Data Analysis Unit (NIAAD), which includes the Machine Learning and Data Mining group and which pertains to LIACC and FEP. He is member of the Coordinating Board of LIACC since 1988. In 2004, 2000 and 1996 scientific coordinator of LIACC. He is also hte President of the Center of Cognitive Science (C.Cog) of the University of Porto. His main research interests include Artificial Intelligence, and in particular the sub-areas Machine Learning, Knowledge Discovery in Databases and Data Mining, with particular focus on: Meta-learning, Pre-processing, Adaptive methods and multi-agent systems in Economics and Management, Extraction of information from text and Relational Data Mining.