Abstract

This paper describes techniques behind the construction of a framework for detection of cognates in different languages. The techniques are developed using eleven similarity measures and transliteration rules, each one used with the purpose of gathering information of particular aspects of words. Two phases are described in the framework: preprocessing and classification. The former phase discards words that can hardly be considered as cognates. The classification step labels the filtered words as cognates resulting from the previous phase using a trained model based on the scores of the similarity measures. This framework produced a $F$-measure rate of 66.93%.

Once the framework is built, it is used for detection of translations of named entities across languages. Together with some specific techniques, this detector uses three named entity recognizer systems to acquire named entities: Stanford NER for English names, XIP NER and an Adaptive Method to acquire Portuguese named entities. Two approaches were used: first Stanford NER was used plus the XIP NER. The second approach consisted in the use of the Stanford NER against the Adaptive Method. The first approach had $F$-measure rate of 62.65%, whilst the second one was more efficient, 73.91% of $F$-measure rate.

1 Introduction

The objective of this paper is to present a framework that will be used for cognate extraction and translation of named entities on texts written in different languages. Parallel corpora is an expensive resource. Since [16] proved that even for unrelated texts of different languages patterns of words co-occurring with each other are preserved, non-parallel texts became part of these systems as well. Comparable corpora is a specific type of non-parallel texts with high level of comparability that point to the same subject, have similar time window and size [17]. This type of texts is seen as an alternative to parallel corpora. News websites like euronews.net, www.bbc.co.uk or intranews.intra.net provide this kind of data.

First we designed techniques for detection of cognate words based on different techniques and similarity measures. Cognates are words that have a common etymological origin. There are three types of cognates: real cognates that are words that have the same etymological origin and have the same meaning, and false cognates which are words that descend from the same etymological origin, but somehow turned out to have different meanings. As real cognates, the words international in English and internacional in Portuguese share the same meaning and ancestors. Another example of real cognates are the words night for English, nuit for French, nacht for German, nacht for Dutch, nicht for Scottish, natt for Swedish and Norwegian, nat for Danish and noite for Portuguese. As false cognates, the word adept in English means specialist and adepto in Portuguese means a supporter. The third type of cognates are loanwords, which are words that have been brought into one language from another. As example, the English word email is extended to many languages across the world. Based on techniques that depend on the specificities of the pair of languages [7], which in the present experiments, are English and Portuguese, the framework will combine English-to-Portuguese rules with the use of similarity measures. For instance, in English, the suffix tion corresponds to the suffix ção in Portuguese, as shown, for instance, in the pair intuition, intuição. For prefixes, sp in English and esp in Portuguese match for the same words quite often, as stated by the pair special, especial. These particular features, together with the use of similarity measures regarding spelling and phonetics of words, may form a platform capable of
responding to the cognate extraction problem.

After this framework is built, it can be used for detecting translation of named entities. Hence, this kind of detector can be built using this framework, as it is a problem that can be also treated by comparing spelling of words. Translation of named entities is not a trivial problem. It deals in many cases with vague and ambiguous named entities (NEs), which is an intrinsic characteristic of natural human language. Systems that provide translation of named entities have been increasing its scope. With the growth of the information in society, entropy increases in terms of organization of information data. News normally contain a great number of NEs. They are varied, and always changing, much more than regular words, and that is a fact that makes the task of creating dictionaries for NEs of translated news even harder. Moreover, translating systems do not have enough trained data to face this problem. Parallel corpora would enhance our translation process of named entities, but aligning this kind of corpora is not a trivial task. So, the alternative is to choose comparable corpora. Therefore, for the tasks of cognate detection and translation of named entities, our experiments will focus on comparable corpora.

The paper is organized as follows: In Section 2 we discuss related work. The overall architecture is presented in Section 3. Next, in Section 4, the experiments and results of cognate detection are described. In Section 5, the results and the experiments of translation of named entities are shown. In Section 6, a discussion is performed over the results obtained. Finally, in Section ?? conclusions are given, as well as future work.

2 Background

In the context of lexicon extraction, since the mid 90’s, many strategies have been developed to perform translation based on non-parallel corpora. For instance, [16] developed methods to identify translations between non-parallel texts considering there is a correlation between patterns of the co-occurrences of the words in texts of different languages. [17] proposed the use of additional resources like an initial lexicon and introduced windowed methods to determine context similarity which permitted to improve the former proposal. [3] presented a similar approach, but the author considered context as the whole sentence where the word was embedded, instead of a constant length window. [7] developed clues on lexicon extraction, namely cognates detection, similar context, preservation of word similarity, and word frequency. For cognate detection, the author used spelling similarity measures like identification of identical words and Longest Common Subsequence Ratio (LCSR) [11]. [19] built a similarity measure that learned which letter changes occurred more likely between cognates of two languages. [13] created a method based on edit distance from a set of known cognates. The method captured orthographic mutations of words when put into the rules of another language. The rules are then applied as a preprocessing step before measuring the orthographic similarity between candidate cognates.

Generally, cognate words are detected by two main methods: the first is based on manual determination of rules describing how spelling of a given word should change if translated into another language: [7] created a list of English-German cognate words by applying well-established mapping rules like the substitution of the letters k or z in German words by c in English. The second method is to compute similarity measures between strings: [10] used edit distance for cognate extraction, and tried to use a third language so the source and target languages could be more familiar with that one. Both are compatible once the latter can work with words that already have the transliteration transformations mentioned on the first method.

Nevertheless, there is a third method that uses context of words. It analyzes a word local context using the web as corpus. For instance, [15] used it in order to distinguish true cognates from false cognates. The context was composed by the words surrounding the target word. It was assumed those words were semantically related to the target word. If two words were semantically related, both should appear in the local contexts of each other. Therefore, a vector was computed containing the frequencies of the context words. A seed glossary was used for filling information in the context vectors. Furthermore, Cosine of both vectors was calculated in order to check context similarity. If both words had the same context, they were true cognates.

A large quantity of new named entities are entered into newspaper agencies every day. That is a problem. They have to be translated into many languages quite often. Unfortunately, normally translation of named entities cannot be found in dictionaries. Accordingly to [5], for translating named entities there are two paths: translation of named entities systems that use a rule-based approach, or statistics-based approach. Both strategies point to transliteration across languages. The former uses linguistic rules for deterministic generation of translation at a character level. It has the disadvantage of, if two languages have completely different character sets, it becomes hard to systematically select the correct characters that have the
The latter is supervised, in the way that for the transliteration process, this method utilizes trained data. Thus, the most probable translations are selected. However, sometimes, the same fragment of characters may have different transliteration, whereby there is a great flexibility on the generation of translated named entities. Hence, the author proposed an improved strategy that combined both referenced techniques. First, a transliteration model is built for helping a web mining process for candidates generation. Afterwards, a Maximum Entropy model [18] is used with different features in order to rank the candidates. This combination enhances low-frequency word translation, enlarge coverage of candidates and potentiates accuracy on the order of the candidate ranking. [4] built a system for translating NEs that is language independent. It performed an alignment of comparable corpora based on semantics. The alignment of documents was supported by generation of candidates of possible aligned documents, and the identification of the correct ones. First, documents of the target language were roughly translated into the source language. Next, a query was performed based on extracted keywords of the documents. Finally, according to some criteria, the corresponding document was retrieved. Once they were aligned, an extractor of named entities was performed which generated candidates for each source language named entity. Finally, based on transliteration mapping rules, the best translation of named entity was retrieved. [6] created a system of translation of NEs from possible non-temporarily aligned corpus. The author used a scoring metric to match the time frames. Moreover, an algorithm was developed for identification of transliteration equivalent classes. [14] developed a NE translation system based on two approaches: combination of bilingual dictionaries with a phonologic/spelling information on the elements of the named entities, providing candidates for every elements of the named entities, and a language-independent grammar based on edition distance. The author did not regard the possibility that the same entity elements might not be at the same order, a feature that occurs quite often on inter-language systems. Our system predicts that possibility, once it compares every element of the named entities from both sides, independently from the order they appear.

### 3 General Architecture

Figure 1: General Architecture

Figure 1 shows that the central part of our architecture is formed by a set of similarity measures and transliteration rules. They are both used for cognate detection and for translation of named entities. The similarity measures are applied to the corpus together with transliteration rules. Then, ranked word pairs result from that application, whereby a training file is computed with the scores of every ranked word pairs generated by the similarity measures. Therefore, for classification of each pair of words, we use Support Vector Machines \(^3\). This method receives the training file as input, returning a training model (\textit{svm\_train}). Then, this model is further used on the prediction (\textit{svm\_predict}) of whether a user-defined pair of words is composed by cognates or not. Taking advantage of the framework already built, a module for translation of named entities was engaged to the system. First, named entity recognizers were used to retrieve named entities (NEs). Stanford Named Entity Recognizer [2] was used in order to detect English language named entities. For the Portuguese language, Xip Named Entity Recognizer [9] and an Adaptive Method were used for named entity recognition. Then, we used the generated model from the Cognate Detection module to verify whether the elements of the retrieved NEs were cognates. Hence, each named entity is checked against the NEs from the other language, and if the majority of the elements of the named entities are considered cognates by the model, then those NEs are translation of each other, even if they appear in different order.

---

\(^3\)en.wikipedia.org/wiki/Support_vector_machine
3.1 Similarity measures

Eleven similarity measures were set for the framework construction. Some of them were based on spelling, and some others on phonetics. Each one is a procedure that receives a pair of words and calculates its resemblance, given by a value in [0,1]. The following shows the specifications of each similarity measure.

3.1.1 Dice

\[
Dice(x, y) = \frac{2|x \cap y|}{|x| + |y|}
\]

(1)

Given two words, Dice can be calculated by Equation 1. This function gives a rate of similarity in [0,1], so does not need normalization. This similarity measure focuses on the amount of common letters between both words, rather than their positions, which means that this measure makes sense if applied together with other similarity measures.

3.1.2 Jaccard

\[
Jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|}
\]

(2)

Another common method for comparing strings, is the Jaccard distance. This similarity measures is very similar to Dice distance. The score is comprehended in [0,1] as well, so it does not need normalization either.

3.1.3 Jaro Winkler

\[
d_j = \frac{1}{3} \left( \frac{m}{|x|} + \frac{m}{|y|} + \frac{m - t}{m} \right)
\]

(3)

\[
\left\lceil \frac{\max(|x|, |y|)}{2} \right\rceil - 1
\]

(4)

\[
d_w = d_j + (lp(1 - d_j))
\]

(5)

The score is comprehended in [0,1]. It is composed by the Jaro distance and a prefix function \(p\). Jaro distance is given by Equation 3 where \(m\) is the number of matching characters. Two characters from \(x\) and \(y\) are considered matching only if they are not farther than the coefficient shown by Equation 4 \(^4\). \(t\) is the number of transpositions, that is, if the matching characters are in the same order, no transpositions are needed, so \(t = 0\). For instance, the words race and cry have 2 matching characters, \(r\) and \(c\), but they are not in the same order, so, in this case, one transposition is needed, \(t = 1\). On the other hand, words like crazy and cry have no transpositions needed as the matching characters cry are in the proper order. Jaro Winkler adds an item to the Jaro distance: a prefix function \(p\). This gives better scores to strings that match from the beginning for a prefix with length \(l\). Equation 5 formalizes the Jaro Winkler measure, where \(d_j\) is the Jaro distance for strings \(x\) and \(y\), \(l\) is the length of common prefix of the strings up to a maximum of four characters and \(p\) is a constant scaling factor which leads the prefix factor greater importance. The standard value for this constant is \(p = 0.1\). Given \(x = \text{night}\) and \(y = \text{nacht}\), \(m = 3\). There are no mismatched characters, so \(t = 0\). Computing the Jaro distance, it is given by \(d_j = \frac{1}{3}(\frac{2}{5} + \frac{3}{5} + \frac{3}{5}) = 0.73(3)\).

The Jaro Winkler score \(d_w\) uses the standard weight \(p = 0.1\) and \(l = 1\), once the initial common prefix between \(x\) and \(y\) has length 1. Thus, the score is given by \(d_w = 0.73(3) + (1 \times 0.1(1 - 0.73(3))) = 0.7597\).

3.1.4 Lcsr

\[
lcsr(x, y) = \frac{|lcs(\text{cons}(x),\text{cons}(y))|}{\max(|\text{cons}(x)|, |\text{cons}(y)|)}
\]

(6)

The longest common subsequence ratio (LCSR) is a character-based measure. \[^1\] formalized it as quotient between the number of sequence letters that are common in both words and the length of the longest word. However, we implemented with a slight difference. Only consonants were considered. This is because, giving two cognate words, they tend to vary vowels more frequently than consonants, so consonants seem to be more relevant in cognate detection. The score is comprehended in [0,1]. For instance, taking into account the words night and nacht, the longest common subsequence is the fragment nht of size 3. The longest word considering only consonants is night and nacht, as they have both the same length. So, the final score is \(\frac{3}{3} = 0.75\).

3.1.5 Lcsm

\[
lcsm(x, y) = \frac{|lcs(x, y)|}{\max(|x|, |y|)}
\]

(7)

Lcsm uses the procedure longest common substring (LCS) \(^5\) in order to recognize the longest common pattern between both strings. The biggest the pattern, the greater the possibility of both words to be equivalents written in different languages. Equation 7 demonstrates how to normalize LCSM, in order to fit in the interval [0,1]. Taking the words night and nacht into account, \(LCS(\text{night}, \text{nacht}) = \text{nht}\), which

\[^4\]en.wikipedia.org/wiki/Jaro-Winkler_distance

\[^5\]en.wikipedia.org/wiki/Longest_common_subsequence_problem
means $|LCS(night, nacht)| = 3$. Normalizing it, the result is $lcsm = \frac{2}{3} = 0.6$.

### 3.1.6 Identical words

\[ ident\_words(x, y) = \begin{cases} 1, & \text{case} \ x = y \\ 0, & \text{otherwise} \end{cases} \quad (8) \]

This similarity measure has the goal to identify words that are spelled exactly the same way. The score is 1 for equally spelled words, or 0 otherwise.

### 3.1.7 Word Length

\[ wL(x, y) = |x| - |y| \quad (9) \]

\[ 1 - \frac{\max(|x|, |y|) - \min(|x|, |y|)}{\max(|x|, |y|)} \quad (10) \]

Word length is the difference between the length of each word of the considered pair. This similarity measure can be used above all for a preprocessing phase, based on the principle that normally cognates do not have great difference concerning length. Since the result is not in $[0, 1]$, it has to be normalized, in order to fit in that interval. Therefore, Equation 10 is applied so that for words with the similar length, the result is approximately 1, and 0 otherwise.

### 3.1.8 Sequence letters

\[ s_j(x, y) = \max(|\text{commonSequenceLetters}(x, y)|) \quad (11) \]

\[ sL(x, y) \quad \min(|x|, |y|) \quad (12) \]

This similarity measure focuses on the length of the longest common sequence of both words that is not interrupted by other letters. This measure had to be normalized, in order to fit in the interval of $[0, 1]$. The normalization is featured by Equation 12. By using this similarity measure, we are enhancing words that have big common fragments of characters, which is a characteristic found in cognate type words. As example, giving words like *night* and *nacht*, this measure identifies the subsequence *ht* as the longest one between both words. Applying normalization, the result is $\frac{2}{5} = 0.4$.

### 3.1.9 Levenshtein

\[ \frac{\max(|x|, |y|) - lev(x, y)}{\max(|x|, |y|)} \quad (13) \]

The *Levenshtein distance* [8] returns the minimal number of operations like substitution, insertion and deletion of characters in order to convert one word into another. Due to the fact that cognates have common ancestral roots, they tend to have few operations for transformation of one word into the other. That means that cognates have small Levenshtein distances quite often. Also this measure had to be normalized, so that the score fits in $[0, 1]$. Equation 13 performs the normalization.

### 3.1.10 Soundex

*Soundex* measure makes use of the phonetics of the words. Unlike the other similarity measures, this has the ability of looking beyond spelling, in a way that if applying Soundex on different words, it may produce the same code for both, which means they sound alike. Normally, this happens to words that have similar orthography, which is a characteristic of cognate words. This is the principle that makes this similarity measure very useful. However, due to the limitations of the algorithm such as the fact that sounds are coded only for single letters or that the letters are only coded for English, the results depend hardly on the considered languages. For equally sounded words, the score is 1, and 0 otherwise.

### 3.1.11 Cmpg

\[ cmpg(x, y) = \text{soundex}(x, y) \ast \text{jaroWinkler}(x, y) \quad (14) \]

This is a similarity measure [12] that was created in order to check words simultaneously in terms of their sound and spelling. It used the *Jaro Winkler* and *Soundex* measures. The fact that Jaro Winkler assign more favorable ratings to strings that match from the beginning for a set prefix length was considered for the process of cognate recognition, in the way that normally cognates match characters from the beginning. It is assumed that if two words are similar, then their phonetic and spelling should not differ a lot.

### 3.2 Transliteration rules

Transliteration rules between English and Portuguese were used. They are segmented into three types that correspond to different parts of words: prefixes, middle parts and suffixes. They were acquired by consultation of on-line resources, and by inspection of the file that contains the manually added cognates. This set of rules can be edited by the user. This fact makes the platform language independent, as any pair of languages can be added to these rules. The following explains transliteration rules with more detail.

---

6[www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm](www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm)

7[www.learnenglish.de/grammar/suffixtext.htm](www.learnenglish.de/grammar/suffixtext.htm)
3.3 Prefixes

A prefix rule is a pair whereby it is possible to match a prefix of a given language into the prefix of another language. For instance, giving the words *special* and *especial*, we observe that the English prefix *sp* can be frequently matched by the Portuguese prefix *esp*. So, for each pair, we replace those prefixes by a single character [char] in both words. In the present example, the result would be, for both English and Portuguese words something like [char]*cial*. This makes the spelling of both words closer to each other.

3.4 Middle parts

Middle rules have the same principle. The only difference is that the rules are applied in the middle of the words. Due to that reason, a mechanism of control had to be created. For instance, suppose we have the rule that replaces letters *k* in the English word and *c* in the Portuguese word for a single character [char]. A pair like *desktop* and *encenar* would be transformed into des[char]top and en[char]enar. These words are completely unrelated, for that makes no sense applying that rule. Instead, the strategy is to control the neighbors of each fragment. For instance, for the words *immigration* and *imigração*, applying the rule mm->[char], it would transform each word into i[char]igration and i[char]igração. By controlling the letters in the left and right of the replaced fragments, we observe that both words have the letters *i* and *i* as neighbors, respectively. Hence, the neighbors have to be the same, in order to apply this kind of rule, for that makes sense to apply this rule to the pair *immigration*, *imigração*.

3.5 Suffixes

Suffix rules are built exactly from the same process as the prefix ones, only with suffixes.

4 Cognate Detection

4.1 Experimental Setup

4.1.1 Corpora and Tools

For the experiments, a corpus was created containing 19 news from politics plus 19 news from the economics sections of the website *euronews.net* translated into English and Portuguese. From that corpus, 15 news of each section were chosen for training, and the remaining 4 for testing. At the same time, each pair of texts was manually annotated with its existent cognates. Those annotations were used for supervising the training process. The training corpus is composed by 444 cognates, whilst the testing corpus has 131 cognates. Only words with more than 2 letters are considered, once most of the words with less than 3 letters are stop words with little meaning. Accents are removed and there are no numerical words. Table 1, Table 2 and Table 3 show a detailed data distribution for each thematic section regarding the number of pairs of words that were compared. External tools like Support Vector Machines *Libsvm* [1] software package is used for classification of cognates.

<table>
<thead>
<tr>
<th>Theme</th>
<th>Economics</th>
<th>Politics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>15 documents</td>
<td>15 documents</td>
</tr>
<tr>
<td>Testing</td>
<td>4 documents</td>
<td>4 documents</td>
</tr>
</tbody>
</table>

Table 1. Number of documents used

<table>
<thead>
<tr>
<th>Theme</th>
<th>Economics</th>
<th>Politics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>107809 word pairs</td>
<td>156453 word pairs</td>
</tr>
<tr>
<td>Testing</td>
<td>35688 word pairs</td>
<td>41300 word pairs</td>
</tr>
</tbody>
</table>

Table 2. Number of compared word pairs

<table>
<thead>
<tr>
<th>Theme</th>
<th>Economics</th>
<th>Politics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avrg words</td>
<td>136 words</td>
<td>168 words</td>
</tr>
<tr>
<td>Avrg cognates</td>
<td>13 cognates</td>
<td>17 cognates</td>
</tr>
<tr>
<td>Avrg sentences</td>
<td>11 sentences</td>
<td>13 sentences</td>
</tr>
</tbody>
</table>

Table 3. Statistics per document

4.1.2 Algorithm

First, there is a preprocessing phase. This step is accomplished using the set of cognates that belong to each pair of news. The objective of this step is to narrow the corpus by discarding words that are completely unrelated. Hence, for each similarity measure, the worst score is extracted from within the cognates present on the whole corpus, defining thresholds, so that only above those, words would worth to be considered. However, those thresholds are not formed *ipsis verbis* from the cognates. The number of cognates would have to be much bigger so that thresholds extracted can be unaltered. Instead, based on a trial and error process, we tried to reach the best configuration as possible. Hence, from the whole set of similarity measures, only *word length*, lcsm, lcsr, jaro winkler and sequence letters are used for preprocessing. These used measures do not need normalization at this stage. Only words that begin with the same letter are considered. Next, classification of cognates takes place. This process only uses those filtered pairs of words resulting
from the early preprocessing step. This time, instead of raw data, we normalize every similarity measures.

In order to verify the contribution and importance of each similarity measure, we first use 11 training files, each one with the scores of only one similarity measure. Each model generated from these files is tested against the testing corpus.

Finally, using the most important similarity measures, we generated the model that is going to be used for the detection of cognates, and tested it against the testing corpora.

4.2 Results

First, we applied the first preprocessing configuration based strictly on the scores of the cognates present in the training corpus. This configuration only used a subset of the whole similarity measures. As mentioned before, these values are the worst coefficients, as from them it is worth to consider words as potential cognates.

<table>
<thead>
<tr>
<th>Dic</th>
<th>Jacc</th>
<th>WLen</th>
<th>Lcsm</th>
<th>Lear</th>
<th>Leav</th>
<th>JWin</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0.25</td>
<td>&gt;0.142</td>
<td>&lt;5</td>
<td>&gt;0.2</td>
<td>&gt;0.33</td>
<td>&lt;5</td>
<td>&gt;0.559</td>
<td>&gt;1</td>
</tr>
</tbody>
</table>

Table 4. Baseline configuration

The results of the application of the baseline configuration in the preprocessing process are shown by Table 5.

<table>
<thead>
<tr>
<th>#Cog found</th>
<th>#Cog missed</th>
<th>#NCog found</th>
<th>Compress ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>432</td>
<td>11</td>
<td>4410</td>
<td>0.018</td>
</tr>
</tbody>
</table>

Table 5. Results of baseline configuration

The baseline configuration permitted to narrow our corpus on a proportion of $\frac{4410+432}{264262} = 0.0183$, which is the ratio between the number of pairs of words that resulted from preprocessing and the number of pairs of words in the training corpus. Nevertheless, the number of cognates retrieved was still quite low, approximately in a ratio of 1 to 10 compared to the number of non-cognates retrieved. Hence, the configuration was altered, in order to lower the number of non-cognates. Given the great variability of cognates in the way they relate to each other, we advanced to a trial and error method by iteratively changing configurations, improving sensitiveness for which thresholds and similarity measures to use.

After many iterations of definition, checking and evaluating the scores of the similarity measures, the final configuration was achieved, which is shown by Table 6.

<table>
<thead>
<tr>
<th>WLen</th>
<th>Lcsm</th>
<th>Lear</th>
<th>JWin</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;4</td>
<td>&gt;0.4</td>
<td>&gt;0.33</td>
<td>&gt;0.55</td>
<td>&gt;2</td>
</tr>
</tbody>
</table>

Table 6. Final configuration

This configuration permitted to achieve better results in terms of filtered non-cognates, but the number of missed cognates increased. However, the trade-off is worthed. A smaller subset of similarity measures was used, which proves that some similarity measures were more important than others for this stage.

<table>
<thead>
<tr>
<th>#Cog found</th>
<th>#Cog missed</th>
<th>#NCog found</th>
<th>Compress ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>410</td>
<td>32</td>
<td>2197</td>
<td>0.0098</td>
</tr>
</tbody>
</table>

Table 7. Results of final configuration

This configuration achieved a compressing ratio of $\frac{2197+410}{264262} = 0.0098$. Despite of loosing a few cognates, using the same corpora as baseline, the proportion of cognates retrieved versus non-cognates retrieved increased to approximately a ratio of 1 to 5.

Next, classification takes place. A trained model is generated with the eleven similarity measures and tested against the testing corpora. Table 8 shows the produced results.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>73.78 %</td>
<td>58.46 %</td>
<td>65.23 %</td>
</tr>
</tbody>
</table>

Table 8. Results of the used model

In order to improve the results shown by Table 8, we have done an experiment that permitted to infer about the most important similarity measures to be used by our model. We evaluated the individual predictive abilities of each similarity measure. For that, we trained the training corpus with only one measure at a time, and tested the resulting model on the testing corpus for every measures. Table 9 demonstrates the results for each similarity measure.

As shown by Table 9, Lcsm, Levenshtein and Dice are the best measures to use in a model regarding F-measure. By constructing a model with only those top 3 similarity measures, results can be improved relatively to the previous model. Table 10 shows its results.
<table>
<thead>
<tr>
<th>Similarity Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levenshtein</td>
<td>59.73 %</td>
<td>68.46 %</td>
<td>63.80 %</td>
</tr>
<tr>
<td>Dice</td>
<td>75.58 %</td>
<td>56.00 %</td>
<td>60.18 %</td>
</tr>
<tr>
<td>Leav</td>
<td>30.02 %</td>
<td>86.15 %</td>
<td>44.53 %</td>
</tr>
<tr>
<td>Soundex</td>
<td>37.59 %</td>
<td>76.61 %</td>
<td>49.99 %</td>
</tr>
<tr>
<td>Identical words</td>
<td>94.73 %</td>
<td>27.69 %</td>
<td>42.85 %</td>
</tr>
<tr>
<td>Jaccard</td>
<td>5.56 %</td>
<td>94.61 %</td>
<td>10.51 %</td>
</tr>
<tr>
<td>Comp</td>
<td>0.29 %</td>
<td>100.0 %</td>
<td>0.59 %</td>
</tr>
<tr>
<td>Jaro Winkler</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
<tr>
<td>Sequence Letters</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
<tr>
<td>Word Length</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
</tbody>
</table>

Table 9. Individual capabilities of the similarity measures

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>69.42 %</td>
<td>64.61 %</td>
<td>66.93 %</td>
</tr>
</tbody>
</table>

Table 10. Results of the used model

5 Translation of Named Entities

5.1 Experimental setup

5.1.1 Corpora and tools

For testing, the corpora used was the already referenced euronews.net corpus from the economics section translated into English and Portuguese. This time, instead of only 4 news, for testing, the whole 19 news were used. 56 translated named entities can be found among the corpus. Table 11 and Table 12 describe the corpus with more detail.

Table 11. Number of documents used

<table>
<thead>
<tr>
<th>Theme</th>
<th>Economic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>19 documents</td>
</tr>
</tbody>
</table>

Table 12. Statistics per document

5.2 Individual Skills of the NER Systems

Using the testing corpora, three methods for performing recognition of named entities were tested individually: Stanford NER for English names and XIP NER for Portuguese ones. This experiment tested also the adaptive method created for acquiring Portuguese names. The result is shown in Table 13.

<table>
<thead>
<tr>
<th>Similarity Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levenshtein</td>
<td>59.73 %</td>
<td>68.46 %</td>
<td>63.80 %</td>
</tr>
<tr>
<td>Dice</td>
<td>75.58 %</td>
<td>56.00 %</td>
<td>60.18 %</td>
</tr>
<tr>
<td>Leav</td>
<td>30.02 %</td>
<td>86.15 %</td>
<td>44.53 %</td>
</tr>
<tr>
<td>Soundex</td>
<td>37.59 %</td>
<td>76.61 %</td>
<td>49.99 %</td>
</tr>
<tr>
<td>Identical words</td>
<td>94.73 %</td>
<td>27.69 %</td>
<td>42.85 %</td>
</tr>
<tr>
<td>Jaccard</td>
<td>5.56 %</td>
<td>94.61 %</td>
<td>10.51 %</td>
</tr>
<tr>
<td>Comp</td>
<td>0.29 %</td>
<td>100.0 %</td>
<td>0.59 %</td>
</tr>
<tr>
<td>Jaro Winkler</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
<tr>
<td>Sequence Letters</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
<tr>
<td>Word Length</td>
<td>0.0 %</td>
<td>0.0 %</td>
<td>0.0 %</td>
</tr>
</tbody>
</table>

Table 13. Results of the individual NER methods

Results show that the English NER system is the one that gets better scores. For Portuguese tagging of names, the XIP NER presents better results against the Adaptive Algorithm.

5.3 Algorithm

We utilize the model generated from the Cognate Detection module. Then, as mentioned in subsection 5.1.1, two approaches take place: first we use the Stanford NER for the English part plus the Xip NER for the Portuguese part. The second approach consists in the use of the Stanford NER against possible NEs from the Portuguese side, which are retrieved by the following adaptive method: first, only words starting with a capital letter are considered to be part of a named entity. Then, we compose named entities by grouping words, one after the other. At the same time, we define function words that can be part of the named entities. For instance, the location Estados Unidos da América, (United States of America) contains the function word da. According to our method, these kind of words can be part of named entities as long as they are not isolated, in the beginning or at the end of a named entity. Finally, we apply the cognate model into the different elements of each named entity for the verification of whether they are cognates. When two named entities from both languages are composed by more than 50% cognates, they are translation of each other.

5.4 Results

Table 14 shows that the method Stanford NER plus Adaptive Method has better results.

Table 14. Results of Translation of Named Entities module

6 Discussion

Some clues can be taken by analyzing the results demonstrated by both modules. The Recall of the
Cognate Detection module is low. That can be explained by the used discipline for considering words as cognates: only words that are pure cognates are considered cognates, in the way that even words that have the same meaning and lemma but do not match in gender or number are discarded. The feature selection procedure revealed that Jaro Winkler has no individual predictive abilities as shown by 0.0 % of F-measure. Sequence Letters and Word Length also reached 0.0 % of F-measure. Nevertheless, these last two are expected to have poor predictive abilities, as they are useful for discarding words that are not cognates much more than proving the opposite. Soundex is surprisingly good concerning its individual capabilities, as it was considered the fifth best similarity measure. Giving the limitations of the algorithm, namely the fact that it only can be used for the English language, and the disadvantage of combination of letters are disregarded, it still gets better performance than Identical words, Jaccard, Cmpg, Jaro Winkler, Word Length and Sequence Letters. That can be explained by its relatively high Recall.

By constructing a model with only Lesm, Levenshtein and Dice, results improved in terms of F-measure when comparing with the model generated by every measures in more than 1.5 %. This percentage is relevant, once it means that, in theory, the system now is able to recognize ≈ 2.1 more cognates than the version before feature selection.

According to the results of the translation of named entities, the combination of the Stanford NER plus the Adaptive Method for English and Portuguese is better when compared to the Stanford NER plus XIP NER. That was the result of the combination of a high Precision tool like Stanford and a relatively high Recall by the Adaptive Method. That is the reason why that combination revealed to be the best among the two.

7 Conclusion and Future Work

7.1 Resume

This framework attempted to learn a model that provides the best accuracy on the cognate detection problem giving comparable corpora, using eleven similarity measures together with transliteration rules. Some of them did not work as it supposed to, as only Lesm, Levenshtein and Dice turned out to be effective on the cognate prediction task. Transliteration rules turned the Cognate Detection module language independent, as it can be extended for any language pair. Cognates are difficult to be detected, once they suffer from many variations, and it is hard to get a good level of resemblance between words that permits to say that words are cognates. The manually compiled scores of the cognates present in the training corpus proved to be a valid starting point to gather sensitiveness of what similarity measures could enhance the trained model. The radial kernel used in Support Vector Machines was the best choice among the four types of kernel provided by libsvm. The model can be improved in future work with some new similarity measures, or even by altering parameters on the used Support Vector Machines.

After the framework was built, it was used for detecting translations of named entities. Three types of named entity recognizers were individually measured: Stanford NER for the English language revealed to be the best NER tool, whilst XIP NER and the Adaptive Method were the second and third respectively. Nevertheless, the best combination is Stanford NER plus Adaptive Method. The idea that the three mentioned name recognizers only served for recognizing names, regardless of the type, states that the system is not demanding concerning these tools, which indicates low dependency of the system regarding NER systems.

By creating an adaptive method for acquiring entities, the work makes that both modules can be run on any Unix platform. This fact adds portability to the system, as a simple installation environment may by enough to accomplish the tasks defined on this application.

7.2 Future work

The built architecture can be easily used by systems like lexicon extractors by inducing a seed composed by cognates. Given comparable corpora, the Cognate Detection module can produce pairs of words that can be used as seed to bootstrap a more complex context module. So, each context vector of each word from the target language would initially contain the counts over the cognate words. Once the seed words are translated, as they are captured by the cognate detection module, the context vectors are translated. Finally, using context similarity measures, it is possible to retrieve the equivalents of both languages.

There are some changes to the system that can be experimented. In order to improve the generated model, instead of using the top 5 measures ordered by F-measure, perhaps the system should be the top 5 measures in terms of Precision. Hence, giving the fact that each similarity measure works on its particular skills, like, for instance, the consonant version of Lcsr,
combination of those different skills may result on a more accurate model.

In order to enhance the system on the cognate detection task, POS tagging can be suggested in order to lower the retrieved false positives, that is, the detected cognates that the model considered as cognates and were not. For instance, the system erroneously considered as cognates words like *does* and *dos*. Once a POS tagger was enabled, it would recognize that the English word *does* is a verb, and the Portuguese word *dos* is a preposition. Since they do not share the same morphological category, they could not be considered as cognates.

Lemmatization also may be a good feature to add to this work. For instance, the pair *president*, *presidential* contains words that belong to the same lemma in their languages. Unlike a system that uses POS tags, if one system only requires a cognate module able to detect cognates regarding just the concept of the word instead of its whole form, lemmatization may be a good characteristic to include on this work.

References


