INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Criação de Léxicos Bilingues para Tradução Automática Estatística

Luís Carlos Amado Magalhães Carvalho

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente:	Doutora Maria dos Remédios Vaz Pereira Lopes Cravo
Orientador:	Doutora Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-orientador:	Doutora Isabel Maria Martins Trancoso
Vogal:	Doutor Bruno Emanuel da Graça Martins

Novembro 2010

Acknowledgements

To my beloved girlfriend Ana. Without her, I would have never finished this course. Her precious advices made me carry on to this important goal in my life.

To my mother, who always tried to pass serenity on me. She always cook my favorite dish when I went to have dinner with her: HER chicken curry :).

To my father in Brazil, who never put extra pressure on me, only worried with my well fair.

To my TV star sister Rita in Netherlands and to my brother Marco, who always wished me luck on every single exam. I miss her.

To the mother of my girlfriend Rosarinha who never stopped to encourage me. She is also a great cook, and I'm looking forward to her next cooking meal :).

To her other dear daughter Joana, who never stopped believing me. She is also a great Poker player, mostly because every chips she earns she gives to me :).

To my professor Luísa who was always on top of everything. She is a great professor, fun to be with, very strict on time schedules, very demanding, but at the same time easygoing. Since the first interview, my impression was the best and I was not mistaken. She always pointed the right path to me, and by following it, I managed to be successful. Otherwise, I think I might not be able to finish this course ever. I learned a lot from her. Many people told me that I got very lucky on my orientator. I agree. If I were beginning my thesis, I definitely would like to have an orientator like Luísa.

To my good friend Ricardo and his wife Ana, who made this Summer, despite of hard work, one of the best. I think the sea in Costa de Caparica misses us even more than we miss it :).

To my friend Luis and his girlfriend Ines who in every single Saturday made me forget how hard was to accomplish this task by riding on his bike at 240 Km/h :). They planted me the bike syndrome, and now I have to get one of those fast babies too :). The sea also misses them a lot.

To my colleague Tiago Luís who helped me a lot in L2F. If he is not stopped immediately, with his selflessness, he may finish your whole work, and we do not want that :).

To the outstanding performance of my football club SLBenfica in the previous season. It made me spend many joyful moments, specially the one with 300 thousand people celebrating the title in Marquês de Pombal.

To James Hetfield and Metallica for performing live to me in the last 4 times in Portugal. It is never enough.

To Virgem Suta in the car CD.

To the Colonel who inexplicably failed me in my last flight as airplane pilot in AFA. I could not be more grateful to him.

Lisboa, November 22, 2010

Luís Carlos Carvalho

To my beloved girlfriend Ana and to my family and friends.

It is better to reign in Hell than to
be slave in Heaven.

Resumo

A pesquisa efectuada no contexto deste trabalho resultou no desenvolvimento de uma framework para detecção de palavras cognatas entre diferentes línguas. A framework centra-se em medidas de similaridade entre palavras e regras de transliteração. A detecção de cognatas foi feita em duas fases: preprocessamento e classificação. A fase de preprocessamento apenas usou um subconjunto das medidas de similaridade por forma a descartar pares de palavras que não partilhavam qualquer semelhança. As medidas foram *Word Length*, *Lcsm*, *Lcsr*, *Jaro Winkler* e *Sequence Letters*. Os pares resultantes foram então aproveitados para a primeira fase de classificação: o treino. Esta fase permitiu gerar um modelo baseado nas medidas de similaridade. Este modelo é utilizado para prever se as palavras são cognatas. De todas as medidas de similaridade, apenas três sao usadas: *Lcsm*, *Levenshtein* e *Dice*. A partir destas medidas, o módulo de cognatas atingiu uma *F-measure* de 66.93%. Após a construção da framework, esta foi usada para detecção de traduções de entidades mencionadas. Este segundo módulo usou três reconhecedores de entidades mencionadas: Stanford NER para nomes escritos na língua inglesa, XIP NER e um método adaptativo para nomes em português. Dois métodos foram utilizados: o primeiro usou o Stansford NER com o XIP NER. O segundo utilizou o Stanford NER mais o método adaptativo. O primeiro alcançou *F-measure* de 62.65%, enquanto que o segundo método revelou-se mais eficiente tendo atingido *F-measure* de 73.91%.

Abstract

The research performed in the context of this thesis resulted in the development of a framework for the detection of cognates across texts of different languages. The framework is centered in word similarity measures and transliteration rules. Cognate detection was accomplished in two phases: preprocessing and classification. The preprocessing phase used only a subset of the whole set of similarity measures in order to discard pairs of words that did not share any resemblance. The measures used were *Word Length*, *Lcsm*, *Lcsr*, *Jaro Winkler* and *Sequence Letters*. Furthermore, the resulting pairs were used in the first step of classification: training. Training permitted to generate a model based on similarity measures. This model is further used to predict whether words are cognates. From the whole set of similarity measures, the model used only three: *Lcsm*, *Dice* and *Levenshtein*. From these measures, the cognate module produced a *F-measure* rate of 66.93 %. After the framework was built, it was used to detect translations of named entities. This module used three named entity recognizers: 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.

Palavras Chave
Keywords

Palavras Chave

Língua natural

Corpora comparável

Cognatas

Tradução

Entidades mencionadas

Keywords

Natural language

Comparable corpora

Cognates

Translation

Named entities

Index

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Statistical Machine Translation (SMT) is the translation of text from one source language to a target language by using statistical methods. These systems normally combine two models: the translation model and the language model. The former is responsible for the *fidelity* of the translation, whilst the latter is responsible for the *fluency* applied to the translations resulting from the translation model. For instance, the sentence *Mary did not slap the green witch* can be translated into *A Maria não bateu à bruxa verde*, *A Maria não deu uma bofetada à bruxa verde* and *A Maria não deu uma bofetada à verde bruxa* by the translation model, and according to that model, the second and third translations have greater probability to be the correct translation. Then, the language model analyses the three translations and assigns a better ranking to the first and second in terms of fluency. Both models have different preferences concerning the best translations, so there must have a trade-off between fidelity and fluency, which points to the second hypothesis as the most correct one.

The parameters associated with both models are usually learned during a training based on the use of monolingual corpora for the language model and parallel aligned corpora for the translation model. Although SMT systems have well-established algorithms for defining probabilities associated to each statistical model, parallel corpora are an expensive resource. Given the expensiveness issue, the use of non-parallel comparable corpora is looked as an alternative source of information in machine translation. Comparable corpora consist of documents in several languages dealing with a given topic or domain. They are much easier to collect than parallel texts. For instance, *euronews.net* is a source of comparable corpora.

Considering the task of bilingual lexicon extraction, context information of words can be used. The association between a word and its context may be preserved through texts of different languages. The context is retrieved based on the principle that one word in one language occurring in a given context is likable to have a translation occurring in similar context. For instance, consider the sentences in Table 1.1.

Taking into account the sentences **e1**, **e2**, **p1** and **p2** that make part of a random corpora, suppose that a translation system is trying to translate the source word *bakery* into a Portuguese target word. By analyzing the context, it is possible to observe that from the English side, the word *bread* appears at both

English	Portuguese
e1 = *I went to the bakery to buy bread*	**p1** = *Fui à padaria comprar pão*
e2 = *That bakery cooks good bread*	**p2** = *A padaria cozeu a mais o pão*

Table 1.1: Example of context of words between languages

e1 and **e2** context sentences, as well as the word *pão* appears on the Portuguese side in **p1** and **p2**. At the same time, the word *cooks* from **e2** and the word *cozeu* from **p2** also occur in both contexts. Given the fact that those two mentioned pairs of words actually mean the same, it is possible to conclude that there is a certain degree of similarity on this context. So, looking for possible target words all over the corpus, the word *padaria* present in **p1** and **p2** is one which has contextual information that is quite similar to the context of *bakery*, for which is a good translation candidate. Nevertheless, the only problem is that the system cannot assume which words are translations of each other on the contexts. Therefore, for lexicon extraction, it is important to have an artifact containing the maximum number of already correctly translated words in order to bootstrap a system that is based on the context information of words. For a better perspective, Figure 1.1 shows the importance of having a seed artifact, which is a list of already translated words.

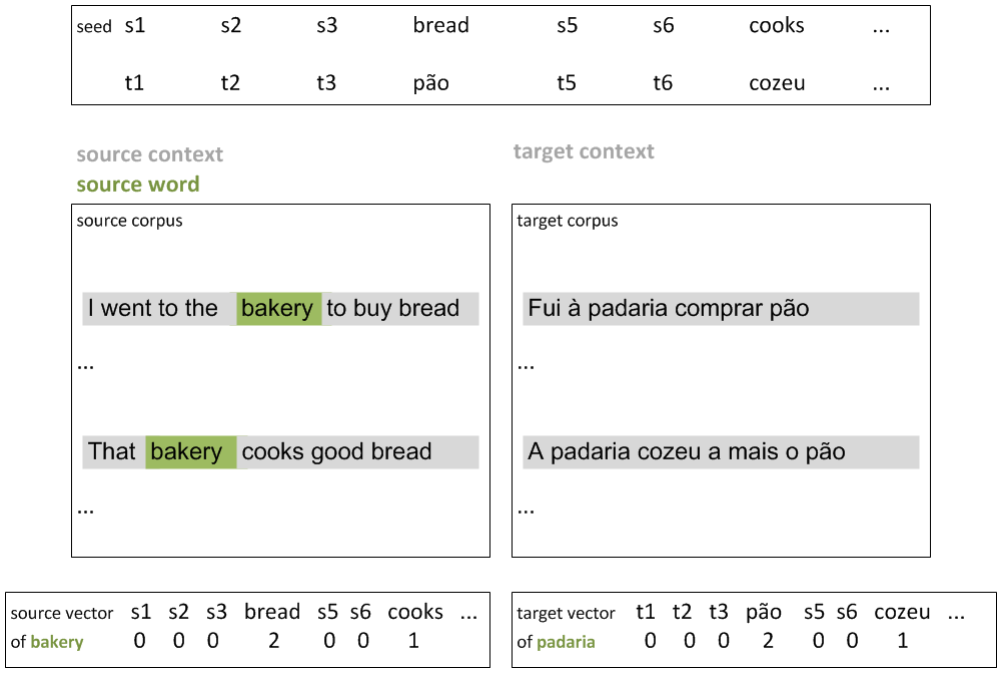

Figure 1.1: Seed usage

Taking into account the example mentioned before, the Figure 1.1 assumes that the seed already has the words *bread* and *cooks* and the correspondent translations *pão* and *cozeu*. After the seed is built, its words are used for composing the context. So, each word from the corpus is described by a vector structure. Each context vector of each word from the target language contains counts over seed words,

as well as the source word of the source language. This example presents the context vectors of the source word *bakery* and the target word *padaria*. They contain counts over the seed words that appear in the context. Once the context vectors are translated, they are compared by context similarity measures. This case shows that both source and target words have very similar context vectors, for which there is a great probability that *padaria* is the correct translation for the word *bakery*. It is an incremental process, as more words are added to the lexicon, more words are added to the seed, which permits to improve the description of each context vector.

By generating a seed resource from comparable corpora, a system like this becomes independent from external dictionaries. This is a feature that greatly enhances a lexicon extractor system, and makes the platform considerably cheaper. So, given comparable corpora, a cognate detection module can produce those seeds. Cognates are words that have the same root in different languages, or as it is said in linguistics, a common etymological origin. There are two types of cognates: real cognates are words that have the same etymological origin and have the same meaning; false cognates are words that descend from the same etymological origin, but during eras those words turned out to have different meanings. Cognates may also include loanwords, which are words that have been brought into one language from another. As an example, the word *email* is brought from the English language to a great number of countries around the world. Most of the times, for detecting cognates, the used techniques are based on spelling resemblance. This fact makes that also false cognates can be detected and added to a possible seed of a lexicon extractor. For instance, the English verb *to have* and the Portuguese verb *haver* are false cognates or friends, as they do not mean the same at all. Fortunately, this kind of words, when compared to real cognates, is a tiny fraction of the cognate universe.

After a cognate detection system is built, it can be used by other applications. For instance, detecting translation of names can be performed using a cognate detector system, as it can be a problem that may be treated with spelling as well. However, translation of named entities is not a trivial problem. It deals in many cases with vague and ambiguous named entities (NEs). That 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 the organization of information data. For instance, news agencies constantly request this feature in order to structure retrieved information. News normally contain a great number of names. They are varied, and always changing, much more than regular words, and that is a fact that makes the task of creating dictionaries for names even harder. Moreover, systems that provide translation of names do not have enough trained data to face this problem.

Hence, using a cognate detector, translations of names can be detected. Figure 1.2 exemplifies how using a cognate detection tool can benefit a translation of named entities framework.

If the correspondent colors of the entity written in different languages like English and Portuguese

Figure 1.2: Concept on named entities translation

can be detected as cognate words, then they can be possibly marked as translation of each other.

1.2 Objectives

The objectives of this work are the following:

– Study cognates and named entities translation.

– Implement a cognate detection module.

– Implement a named entities translation module that takes advantage of the cognate detection module.

– Evaluate both modules.

1.3 Document Structure

In Chapter 2, a survey on the state of the art regarding cognate detection systems and translation of named entities is done. Different similarity measures and techniques about cognate detection are showed. Several methods for translation of named entities are presented.

Chapter 3 shows the system architecture.

In Chapter 4, the approach for cognate detection and name translation is described.

In Chapter 5, an evaluation of the system is done with several experiments and examples.

Finally, in Chapter 6, conclusions are taken by performing a summary of the built framework, future work and the contributions of this work.

2 State of the art

2.1 Introduction

Section 2.2 will give an overview of applications that can benefit with cognate detection. Several similarity measures will be mentioned. Section 2.3 will explain used techniques for cognate acquisition in detail.

Section 2.4 will introduce different approaches on translation of named entities.

2.2 Cognate Applications

Cognate detection is an important problem. On natural language processing, several applications can benefit in a great deal with a built cognate detection framework. Areas like language reconstruction, sentence and word alignment in bilingual texts, machine translation or disambiguation of texts can improve results using a cognate tool.

(Kondrak, 2009) identified cognates and sound associations in word lists on the context of linguistics. Cognates can be a valid form for discovering a proto-language [1] of two languages. Languages always come from a proto-language. Given absence of proof documents that confirm the existence of a given proto-language, determining cognates and sound correspondence across languages may be a way for reconstructing proto-languages. For instance, Portuguese and Spanish come from Latin. However, there are many historical documents that prove the existence of Latin. On the contrary, Euro-Asian proto-languages lack in material evidence of their existence. For instance, the English word *hundred*, the French word *cent*, and the Polish word *sto* are all descendants from the Proto-Indo-European word *ḱmtom*. Nevertheless, all forms of the proto-language word are very different. This is the reason why a sound tool for detecting sound relations may help. An association between sounds of the different languages is established as a workaround to spelling misfortune like the *hundred* example. In order to detect cognates, the author used three methods: orthographic and phonetic measures, determination of recurrent sound associations and semantic information. For orthographic, Dice and Longest Common Subsequence Ratio (LCSR) (Melamed, 1995) were used. For phonetic measures, JAKARTA (Oakes, 2000)

[1] http://en.wikipedia.org/wiki/Proto-language

and ALINE (Kondrak, 2000) were used. Furthermore, determination of recurrent sounds between languages was made with the help of an algorithm for aligning words (Melamed, 2000). Basically, after N iterations, each word in the source side has a correspondent translation on the target side. That translation process is based on assigning words with likelihood scores. By utilizing this alignment algorithm, the author mapped alignments on word phoneme pairs, just like in translation. Finally, the author used semantic similarity measures. The first one used semantic similarity by assuming that two words can often be detected as cognates by comparing their respective glosses [2]. That assumption was not totally right because the simple fact that two words have common lexemes in their glosses is not demonstrative that they are cognates. So, a second measure was introduced. Due to the fact that several lexemes in glosses may carry very little meaning, a keyword method for selecting lexemes was created. Not every words in glosses were compared, only the keywords. Those keywords can be defined by a POS tagger. Normally, nouns are the most meaningful part in a sentence. However, that selection may cause flaws. This technique is dependent from the POS tagger performance, which can be a handicap due to the general morphological ambiguity of words. The third measure is to use Wordnet [3]. The method utilizes directly linked knots by a relationship link, and estimates the semantic similarity based on the type of link. The links can be synonyms, hypernyms or meronyms. Figure 2.1 [4] exemplifies the words *lump*, *roe*, *fish* and *egg* by showing the correspondent keywords in relations like synonymy [5], hypernymy [6] and meronymy [7].

Word	Synonyms	Hypernyms	Meronyms
lump	ball, clod, glob, clump, chunk, swelling, klutz, puffiness, lout, clod, goon, stumblebum, oaf, lubber, lummox, gawk, hunk	agglomeration, piece, part, symptom, clumsy person	—
roe	hard roe	spawn, **egg**, seafood	**fish**
fish	chump, fool, gull, mark, patsy, fall guy, sucker, shlemiel, soft touch, mug, go fish	foodstuff, food product, victim, card game, cards, dupe, aquatic vertebrate	pisces, school, shoal
egg	testis, gonad, testicle, ball, ballock, bollock, nut	endocrine gland, ductless gland, ovum, egg cell, foodstuff, food product	male genitalia, family jewels

Figure 2.1: Lists of semantically related words extracted from WordNet

The phonetic and the correspondence-based approaches produced continuous scores. The semantic method supplies a vector of eight binary semantic features. The three types of approach are combined

[2] en.wikipedia.org/wiki/Gloss
[3] wordnet.princeton.edu
[4] example taken from (Kondrak, 2009)
[5] http://en.wikipedia.org/wiki/Synonym
[6] http://en.wikipedia.org/wiki/Hyponymy
[7] http://en.wikipedia.org/wiki/Meronymy

into a single value in [0,1], as when that value is close to 1, it meant that words are closely to be considered cognates. Therefore, by acquiring a set of cognates from two languages that are thought to be related, their proto-language can be reconstructed.

Cognates can be used in lexicon extraction as well. When there is a bilingual corpus whereby words are extracted, this task is normally performed with contextual information of words from both sides. Cognates can be used for filling that information.

The context of a word can be defined by words that appear in surrounding positions. In fact, the information that is used for the context retrieving are not the words *per si*, but their frequency counted over window positions. The context in the source language is translated into the target language. Then, the word with the most similar context in the target language is retrieved (Koehn & Knight, 2002). (Rapp, 1999) used a 2-word window to collect words to compose the context, that is, two words ahead and two words following the target word. Each context word is counted over the four individual positions of the window. This means that a set of four context vectors is used, each per position. This approach uses an initial lexicon, commonly referred as the seed, which is normally filled with cognate words. It is used to count the co-occurrences of the context vectors. Each vector dimension, that is, each position of the context vector (Rapp, 1999), takes as value the number of co-occurrences between the target word and a seed word in a given position. Thus, each target word is represented by 4 context vectors, forming an association vector for that word. Each vector of the association vector is composed by the co-occurrences of N words, where N is the size of the seed. Then, the four vectors are combined forming only one vector with size $4N$. This association vector is then compared with the association matrix in the target language, which is formed by all association vectors. The best ranked vectors in the target matrix are the candidate vectors. The association vector in the target language with the most similar score to the association vector in the source language holds the translation chosen for the word. (Fung & Yee, 1998) used a similar method, but the length of the window was variable, once counts were based on how often context words co-occurred with the target word in the same sentence. For a better visualization, the following describes the translation process of a given source word from the source corpus using a seed.

First, in Figure 2.2, the first phase is described. Four vectors vertically arranged are computed in the way that each cell of a vector corresponds to the frequency of each seed word in that position relatively to the target word. The rows represent seed words translated into the source language and the columns represent the different positions around the target word.

Then, as shown by Figure 2.3, the four vectors are combined into one association vector that is further compared to each one of the association vectors of the target association matrix. Each row corresponds to an association vector of a word from the target corpus and each column holds a seed word translated into the target language. Each cell is computed by the frequency values of the seed words

	dim 1	dim 2	source word	dim 3	dim 4
seed 1	f(seed 1)	f(seed 1)		f(seed 1)	f(seed 1)
...	...	...		...	...
seed N	f(seed N)	f(seed N)		f(seed N)	f(seed N)

Figure 2.2: Position of context seed words

on the target corpus. Hence, using similarity measures for context vectors, the target vector that shows greater similarity to the source vector holds the most reliable translation for the considered source word. *Cosine* (Fung & Yee, 1998) and *CityBlock* (Rapp, 1999) are two similarity measures very used in vector comparison.

Other uses can be found on the application of cognates. (Uitdenbogerd, 2005) found that the use of cognates improves readability of texts written in a foreign language like French. The experiment relied on people with different skills on French language, mostly native English speakers. The experiment consisted in provide different kind of books with different difficulty level. It was concluded that the number of cognates in the text was reasonably highly correlated with readability, as with a high cognate number and short length sentences, both factors influenced readability in a positive manner rather than other features.

Statistical translation models can also be improved with cognates. (Kondrak et al., 2003) decided to incorporate cognates into the translation models of (Brown et al., 1993). The method was simple: first, cognates were extracted. For that matter, spelling similarity measures were used, like Simard's condition (Simard et al., 1992), Dice's coefficient, and Longest Common Subsequence Ratio. Once those cognates were retrieved, the objective was to perform additional co-occurrence counts between cognates and other words that were already counted. Then, each segment is split into words, and every possible one-to-one equivalents are stored into a file. That file is sorted by the value of the computed measures. By setting a threshold on the similarity value, a subset of pairs of words were considered cognates. Finally, the pairs were added to the training corpus along with the preexistent parallel sentences. Results showed that the word alignments improved, leading to better quality translation models.

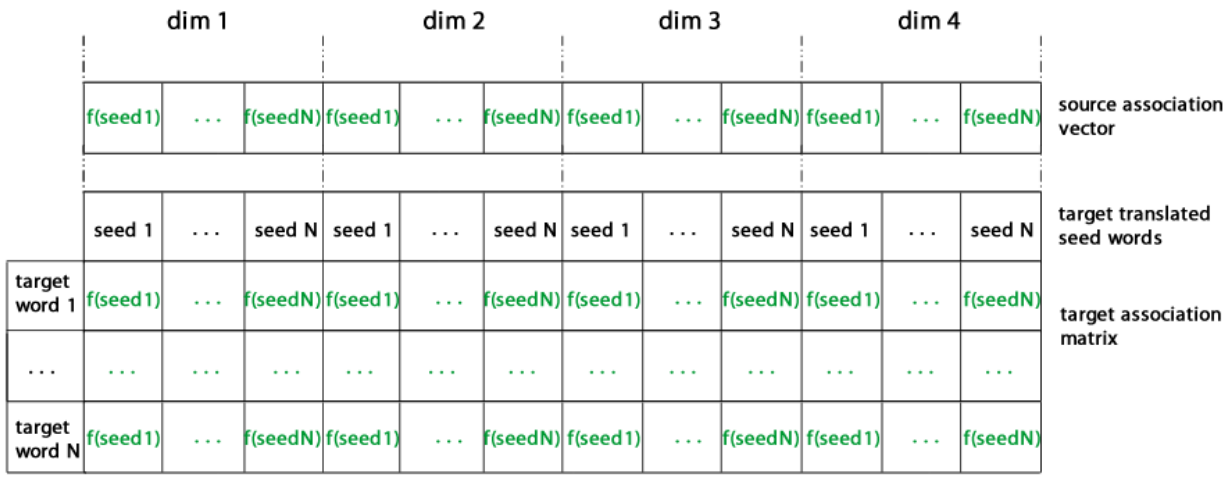

Figure 2.3: Vector comparison using seed words as context

2.3 Cognate Techniques

In section 2.2, several applications of cognates were referenced. This section explains what techniques are behind the cognate identification problem.

Basically, there are three elementary methods for recognizing cognates: through orthographic similarity, phonetic similarity and semantical similarity. The orthographic approaches normally disregard the fact that alphabetic symbols express sounds, employing a binary identity function on the level of character comparison, that is, there is similarity only if the same letter occurs in the considered words.

Most of the times, applications recognize cognates by spelling resemblance of words or adaptive supervising algorithms that learn from a training corpus. For instance, (Rapp, 1999) used spelling similarity measures that essentially correspond to the identification of nearly identical words and Longest Common Subsequence Ratio (Melamed, 1995) in order to detect cognates. (Tiedemann, 1999) built a similarity measure that learns which letter changes occur more likely between cognates of two languages, and applied that on cognate detection. (Mulloni & Pekar, 2006) created a method based on edit distance from a set of known cognates. The method captures orthographic mutations of words when put into rules of another language. The rules are then applied as a preprocessing step before measuring the orthographic similarity between candidate cognates. (Koehn & Knight, 2002) 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. (Mann & Yarowsky, 2001) 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.

The phonetic methods, on the other hand, take advantage of the phonetic characteristics of individual sounds in order to estimate their similarity. A sound transcription of the words into a phonetic representation is required. Phonetic approaches, such as Soundex (Hall & Dowling, 1980) or Editex (Zobel & Dart, 1996) attempt to take advantage of the phonetic features of individual characters in order to estimate their similarity. Soundex only takes into account individual sounds of characters on a word. Editex, on the contrary, uses combinations of letters and assign them specific sounds.

Calculating cognates based on semantic similarity can use structured information about word relations. The easiest way to calculate semantic similarity of two words is by comparing their lexemes and verify their common glosses. For instance, (Kondrak, 2001) used Wordnet [8] to compute semantic similarity between words. Obviously, the use of the WordNet for semantic similarity detection is possible only if the glosses are translated into English. The possible workaround is to translate the glosses into English, but that would add a considerable overhead to the system. Instead, EuroWordNet (Vossen et al., 1998) can be used once it extends semantics to many European languages.

Other methods for cognate detection can work. Techniques that use context of words have a prominent place too. For instance, (Nakov & Nakov, 2007) developed a method that analyzes the word local contexts using the *web* as a corpus in order to distinguish true cognates and false ones. It is assumed that if two words are semantically related, both should appear in the local contexts of each other. Thus, a vector is computed containing the frequencies of the context words. A seed glossary is needed, and those are the words that are counted in the context vectors. Furthermore, Cosine of both vectors is calculated in order to check context similarity. If both words have the same context, the method considers them as true cognates.

(Mackay & Kondrak, 2005) used an approach that made correspondence tokens between words of both languages. Then, each pair of words gets aligned using operations like *substitution*, *insertion* and *deletion*. Each operation assigns a different probability to the pair of words. The method uses a technique called Pair Hidden Markov Model (PHMM) (Arribas-gil et al., 2006), which assigns values to match probabilities, gap probabilities and transition probabilities evolving the referred operations during training.

[8] wordnet.princeton.edu

2.4 Translation of Named Entities Techniques

A large quantity of new named entities are entered into newspaper agencies every day and they have to be translated into different languages quite often. Unfortunately, normally, translation of named entities cannot be found in dictionaries.

According to (Jiang et al., 2007), in translation of named entities there are two choices: 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. The latter is supervised, in the way that for the transliteration process, this method utilizes trained data. In transliteration, a model is built with the goal of helping a web mining process to generate candidates. Afterwards, a Maximum Entropy model (Ratnaparkhi, 1998) is used with different features in order to rank the candidates. This combination enhances low-frequency word translation, and results on great improvements such as coverage enlargement of candidates and accuracy enhancement on the order of the candidate ranking.

Other authors proposed some simpler approaches. For instance, (Huang & Vogel, 2002) performed named entity translation by tagging the named entities from each language first. The author proposed an integrated approach to extract a named entity translation lexicon from a bilingual corpus. The named entities are first extracted independently for each language. Then, in order to align named entities, an alignment procedure is performed based on statistics. The names are added to the lexicon when the probabilities reach a certain threshold and when the element are named entities.

In order to learn translations of NE sentences, (Moore, 2003) developed sentence translation models from parallel software manuals.

With the success of search engines such as Google [9] or Bing [10], these systems have been collecting data that may permit to improve named entities translation. NEs can be retrieved based on several alignment features applied to those web pages. Since they keep every texts stored in their caches, they get to have a huge corpus that can be used for web-mining tasks. For instance, considering a system that is trying to translate a given English name like *France* into the Spanish equivalent *Francia*, *France* is entered into a query for searching pages in *google.com*. By counting over the times that *France* appears into the first 5 results, it is possible to conclude that *France* is one of the most frequent words. By switching *google.com* for the target language search engine *google.es*, the same word (*France*) is entered into a query too. Now, in spite of the word France occurs many times as well, the word *Francia* also appears as often as *France*, perhaps more times if considering a 20 or 30 length window for results. So, a

[9] www.google.com
[10] www.bing.com

simple approach like this can be used for translating named entities. By adding more parameters on the *google.com* query, the translation process may lead to even better results.

(Hassan et al., 2007) developed a system for translating NEs which is language independent. It performs an alignment of comparable corpora based on semantics. The alignment of documents is supported by generation of candidates of possible aligned documents, and the identification of the correct ones. First, documents of the target language are roughly translated into the source language. Next, a query is performed based on extracted keywords of the documents. Finally, according to some criteria, the corresponding document is retrieved. Once they are aligned, an extractor of named entities is performed, which generates candidates for each source language named entity. Finally, based on transliteration mapping rules, the best translation of named entity is retrieved.

(Alegria et al., 2006) 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 edit 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.

3 Architecture

This Chapter describes the general architecture of the system which is composed by two modules: cognate detection and translation of named entities.

3.1 Overview

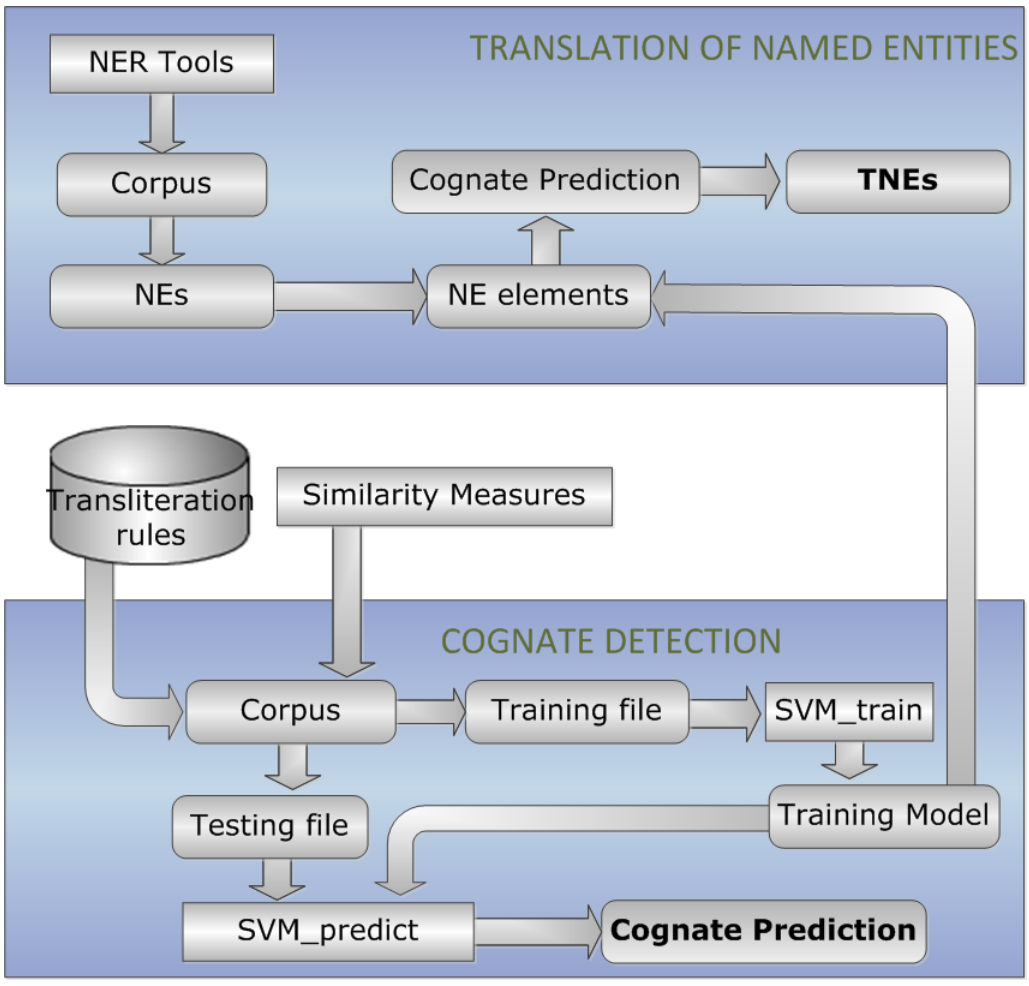

Figure 3.1: General Architecture

Figure 3.1 shows that the central part of the system 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, the method Support Vector Machines [1] is used. This method receives the training file as input, returning a training model (*svm_train*). Then, this model is further used on the prediction (*svm_predict*) of whether a user-defined pair of words are cognates or not.

Taking advantage of this framework, a module for translation of named entities is added to the system. First, already established named entity recognizers are used to retrieve named entities (NEs). Stanford Named Entity Recognizer (Finkel et al., 2005) is used in order to detect names from the source language. For the target language, Xip Named Entity Recognizer (Mamede, 2007) and an adaptive method are used. Then, the cognate detector is used in order to verify whether the elements of the retrieved NEs are cognates. Then, each named entity is checked against the NEs from the other language, and if the majority of the elements of a named entity are considered cognates by the model, even if they have different order, then that pair of NEs are considered to translation of each other.

3.2 Application Example

For a better comprehension of the system architecture, an example is given, explaining the different steps traversed by the architecture, since a pair of texts sample written in English and Portuguese are provided to the system until cognate pairs and translated names are returned. The following texts are an excerpt of news from the website *euronews.net* translated into English and Portuguese respectively.

> US president Barack Obama described him as the most popular man on earth, but after eight year's in the job President Luiz Inacio Lula da Silva is about to step down. Known simply as Lula, the former shoeshine boy, lathe operator, and militant trade union leader rose to become Brazil's first left leaning head of state. While, he may not have achieved the global celebrity of Nelson Mandela, Lula's legacy is likely to linger long after this presidential poll.

> É incontestável que se está a virar uma página da história do Brasil com o fim da era Lula, o antigo sindicalista de inspiração trotskista que se tornou o primeiro presidente brasileiro de esquerda em 2002. Luís Inácio Lula Da Silva tinha enormes responsabilidades e gerava muitas expectativas. Mas soube impor uma terceira via à moda do Brasil e trazer o país para o primeiro plano.

[1] en.wikipedia.org/wiki/Support_vector_machine

3.2.1 Cognate Detection

First, a corpus is defined for training. This corpus contains news from the website *euronews.net* translated into English and Portuguese. For each pair of texts, a combination of pairs of words from both languages is made by computing the scores of every similarity measures. Those scores also result from the application of transliteration rules for each word pair. Therefore, a combinatory explosion of word pairs takes place, as result of the N x N combination of words for each text of the training corpus. Given the fact that in translated texts cognate words are in much less quantity than the normal ones, a mechanism for discarding words that do not share any resemblance is created. This permitted to have a more balanced training process, as the percentage of cognate words increased with this preprocessing procedure. As input, the Support Vector Machines receives a file containing a description of normalized similarity measures of pairs of words that are filtered as mentioned. It generates a model responsible for classifying future word pairs as cognates or not. Afterwards, the Support Vector Machines receive a test file containing every combination of pairs of words, ranked by the every measures.

The following is an example of a pair of words extracted from the application example that is described in terms of its similarity measures.

```
presidential presidente 2:1.0 3:0.929 4:0.8 5:0.857 6:0.727
 7:0.75 8:0.833 9:0.93 10:0.75 11:0.9
```

This is the case of the pair *presidential , presidente*. This format is extended to every pair that is entered into the Support Vector Machines. Omitted measures are 0.0. So, measure 1, *identical words*, is 0.0. The second measure, *Soundex*, is 1.0. Measure 3, *Cmpg*, is 0.929. *Jaccard* is represented by the fourth measure and its score is 0.8. *Lcsr* is represented by the measure 5. Measure 6, *Dice*, has the worst score 0.727. 7, 8, 9, 10 and 11 correspond to *Levenshtein*, *Word Length*, *Jaro Winkler*, *Lcsm* and *Sequence letters*, respectively. Section 4.2 explains each similarity measure in detail.

For this particular example, the framework detected as cognates the following words:

```
[president]->[presidente]
[inacio]->[inacio]
[lula]->[lula]
[silva]->[silva]
[brazil]->[brasil]
[presidential]->[presidente]
```

This is an application example for calculating the cognates of two texts written in a source and a

target language. As shown in Section 3.3, this option corresponds to option 2: *detect cognates on a pair of texts using the default train model*.

3.2.2 Translation of Named Entities

First, with the help of a named entity recognizer such as the Stanford NER software package (Finkel et al., 2005) [2], named entities from the English text are retrieved. Then, for the Portuguese text, possible names are found with an adaptive method as well. These possibilities are found regarding words that start with capital letters. With the help of the cognate framework, particularly the use of the generated model from the training phase, it is possible to detect cognates on parts of names translated in both languages. When the cognate system detects cognates on most of those parts, the system considers them as a translation of each other.

The English NER retrieved the following entities:

```
[Barack Obama] [Luiz Inacio Lula da Silva] [Mandela] [Lula] [Nelson US] [Brazil]
```

The Portuguese NER found these possible entities:

```
[E] [Brasil] [Lula] [Luis] [Inacio Lula Da Silva] [Mas]
```

Finally, the system detected these translated names:

```
[Luiz Inacio Lula da Silva]->[Inacio Lula Da Silva]
[Lula]->[Lula]
[Brazil]->[Brasil]
```

This is the result of applying these application texts in option 1 of the Translation of Named Entities Module: *translate NEs using the Stanford NER and the default train model*. This is one of many options that this module provides to the user. Section 3.3 shows every options available.

3.3 User Interface

The user interface is a command line, where the user gets to choose what actions to perform. The interaction contemplates the cognate module and the named entities translation module. Each module has its own methods and options.

[2] nlp.stanford.edu/software/CRF-NER.shtml

```
+------------------------------+
|   Cognate Detection Module   |
+------------------------------+
Detect cognates on a pair of words using the default train model
Usage: cd 1 {word1} {word2} - e.g.: cd 1 night noite

Detect cognates on a pair of words using a userdefined train model
Usage: cd 1 {word1} {word2} [1-11]* - e.g.: cd 1 night noite 2-3-6-8

Detect cognates on a pair of texts using the default train model
Usage: cd 2 {file1} {file2} - e.g.: cd 2 /user/file1 /user/file2

Detect cognates on a pair of texts using a userdefined train model
Usage: cd 2 {file1} {file2} [1-11]* - e.g.: cd 2 /user/file1 /user/file2 2-3-6-8

Get similarity measure score of a pair of words
Usage: cd 3 {word1} {word2} [1-11] - e.g.: cd 3 night noite 7

Get most similar words from two texts
Usage: cd 4 {file1} {file2} - e.g.: cd 4 /user/file1 /user/file2


+------------------------------+
|   Translation of NE Module   |
+------------------------------+
Translate NEs using Stanford NER and the default train model
Usage: tne 1 {file1} {file2} - e.g.: tne 1 /user/file1 /user/file2

Translate NEs using Stanford NER and a userdefined train model
Usage: tne 1 {file1} {file2} [1-11]* - e.g.: tne 1 /user/file1 /user/file2 2-3-6-8

Translate NEs using Stanford NER plus Xip NER and the default train model
Usage: tne 2 {file1} {file2} - e.g.: tne 2 /user/file1 /user/file2

Translate NEs using Stanford NER plus Xip NER and a userdefined train model
Usage: tne 2 {file1} {file2} [1-11]* - e.g.: tne 2 /user/file1 /user/file2 2-3-6-8
```

4 Cognate Detection and Translation of Named Entities

4.1 Introduction

This chapter describes the procedures that lead to the construction of a cognate detection module and the translation of named entities module. Section 4.2 explains the specifications of each similarity measure. Section 4.3 presents the reasons for the use of transliteration rules. In Section 4.4, an overview over Support Vector Machines technique take place, and its conceptualization regarding this work is described. Section 4.6 and section 4.7 show the algorithms and procedures of both Cognate Detection and Translation of Named Entities modules.

4.2 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.

4.2.1 Dice

$$Dice(x, y) = \frac{2|x \cap y|}{|x| + |y|} \tag{4.1}$$

Given two words, *Dice* can be calculated by Equation 4.1. 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.

4.2.2 Jaccard

$$Jaccard(x, y) = \frac{|x \cap y|}{|x \cup y|} \tag{4.2}$$

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.

4.2.3 Jaro Winkler

$$d_j = \frac{1}{3}(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}) \tag{4.3}$$

$$\lfloor \frac{\max(|x|,|y|)}{2} \rfloor - 1 \tag{4.4}$$

$$d_w = d_j + (lp(1-d_j)) \tag{4.5}$$

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 4.3 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 [1]. 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 4.5 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 $x = night$ and $y = nacht$, $m = 3$. There are no mismatched characters, so $t = 0$. Computing the *Jaro distance*, it is given by $d_j = \frac{1}{3}(\frac{3}{5} + \frac{3}{5} + \frac{3-0}{3}) = 0.73(3)$.

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

$d_w = 0.73(3) + (1 * 0.1(1 - 0.73(3))) = 0.7597.$

4.2.4 Lcsr

$$lcsr(x, y) = \frac{|lcs(cons(x), cons(y))|}{max(|cons(x)|, |cons(y)|)} \tag{4.6}$$

The *longest common subsequence ratio* (*LCSR*) is a character-based measure. (Melamed, 1995) formalized it as quotient between the number of sequence letters that are common in both words and the length of the

[1] en.wikipedia.org/wiki/Jaro-Winkler_distance

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 *nght* and *ncht*, as they have both the same length. So, the final score is $\frac{3}{4} = 0.75$.

4.2.5 Lcsm

$$lcsm(x, y) = \frac{|lcs(x, y)|}{max(|x|, |y|)|} \tag{4.7}$$

Lcsm uses the procedure longest common substring (LCS) [2] 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 4.7 demonstrates how to normalize LCSM, in order to fit in the interval $[0, 1]$. Taking the words *night* and *nacht* into account, $LCS(night, nacht) = nht$, which means $|LCS(night, nacht)| = 3$. Normalizing it, the result is $lcsm = \frac{3}{5} = 0.6$.

4.2.6 Identical words

$$ident\_words(x, y) = \begin{cases} 1, case x = y \\ 0, otherwise \end{cases} \tag{4.8}$$

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.

4.2.7 Word Length

$$w\_l(x, y) = ||x| - |y|| \tag{4.9}$$

$$1 - \frac{max(|x|, |y|) - min(|x|, |y|)}{max(|x|, |y|)} \tag{4.10}$$

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 4.10 is applied so that for words with the similar length, the result is approximately 1, and 0 otherwise.

[2] en.wikipedia.org/wiki/Longest_common_subsequence_problem

4.2.8 Sequence letters

$$s\_l(x, y) = max(|commonSequenceLetters(x, y)|) \tag{4.11}$$

$$\frac{s\_l(x, y)}{min(|x|, |y|)} \tag{4.12}$$

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 4.12. 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 $\frac{2}{5} = 0.4$.

4.2.9 Levenshtein

$$\frac{max(|x|, |y|) - lev(x, y)}{max(|x|, |y|)} \tag{4.13}$$

The *Levenshtein distance* (Levenshtein, 1966) 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 4.13 performs the normalization.

4.2.10 Soundex

Soundex (Hall & Dowling, 1980) 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 [3]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.

4.2.11 Cmpg

$$cmpg(x, y) = soundex(x, y) * jaroWinkler(x, y) \tag{4.14}$$

[3]www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm

This is a similarity measure (Moreira & Gonçalves, 2009) 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.

4.3 Transliteration Rules

Some languages have words that share patterns of letters that are frequently repeated. For instance, in English, the suffix *tion* frequently matches the Portuguese suffix *ção* for the same word, as showed by the pair *constitution, contituição*. Another example, are the suffixes *ly* and *mente* which are shared by cognates like *fortunately, afortunadamente* or *namely, nomeadamente*. Once this kind of patterns are learned, they can be used to improve the values of the similarity measures. By adding these rules, the words that share these fragments are no longer penalized by those particular letters, as they no longer not count for measuring similarity between words. These rules 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 [4], and by inspection of cognates in the *web*. This set of rules can be edited by the user, so any languages can be used, for which makes the platform language independent. The following explains transliteration rules with more detail.

4.3.1 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]*ecial*. This makes the spelling of both words closer to each other.

4.3.2 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

[4] www.learnenglish.de/grammar/suffixtext.htm

[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->m*, 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*.

4.3.3 Suffixes

Suffix rules are built exactly from the same process as the prefix ones, only with suffixes. An example is the pair of words *impossible, impossível*. This pair, as many other English-Portuguese pairs, has a suffix *ble->vel*, which is repeated among many other word pairs.

For instance, Table 4.1 shows the impact of the use of transliteration rules on a pair of words. With the application transliteration rules, such as *tion* > *ção*, a representative similarity measure like Levenshtein distance gets much better scores.

Levenshtein without transliteration		**Levenshtein with transliteration**	
word pair	**score**	**word pair**	**score**
computa[tion] computa[ção]	3.0	computa[tion] computa[ção]	0.0

Table 4.1: Example of transliteration

4.4 Support Vector Machines

Support Vector Machines (SVM) is a machine learning technique which can be used for predicting a certain phenomena based on prior events. In the present case, the SVM software *libsvm* (Chang & Lin, 2001) was used in order to train a model based in similarity measures (features) between strings, and to test new data based on the same features. The objective was to classify new pairs of words as cognates or non-cognates.

For each pair of words (instance), the software used the following data format:

```
label 1:[0,1] 2:[0,1] 3:[0,1] 4:[0,1] 5:[0,1] 6:[0,1] 7:[0,1] 8:[0,1]
      9:[0,1] 10:[0,1] 11:[0,1]
```

Each instance is described by a set of values in [0,1], for which a *scaling* operation is not necessary. The label coefficient is defined as 1 for positive, and -1 for negative. This format is used for both training and

testing. Nevertheless, when the software is training, the *label* value is relevant because it states whether the instance is a cognate or not, for which is based on the association of the combination of each feature value with the label that the training model is created. For testing, this value is absolutely irrelevant, since the *label* value corresponds to the diagnosis value the system is trying to achieve by classifying words as cognates or not cognates. So, when the software is testing words, it ignores the *label* value.

In order to train a model, there are certain parameters in *libsvm* to select. These parameters can be related with the future effectiveness of the generated predictive model.

In background, during the learning process, the Support Vector Machines technique uses the training data set and transforms the feature values into bi-dimensional pairs(xi, yi). *Libsvm* draws a hyperplane on a x, y referential in order to maximize the distance between points of category 1 and points of category 2, when dealing with binary choices such as the cognate classification.

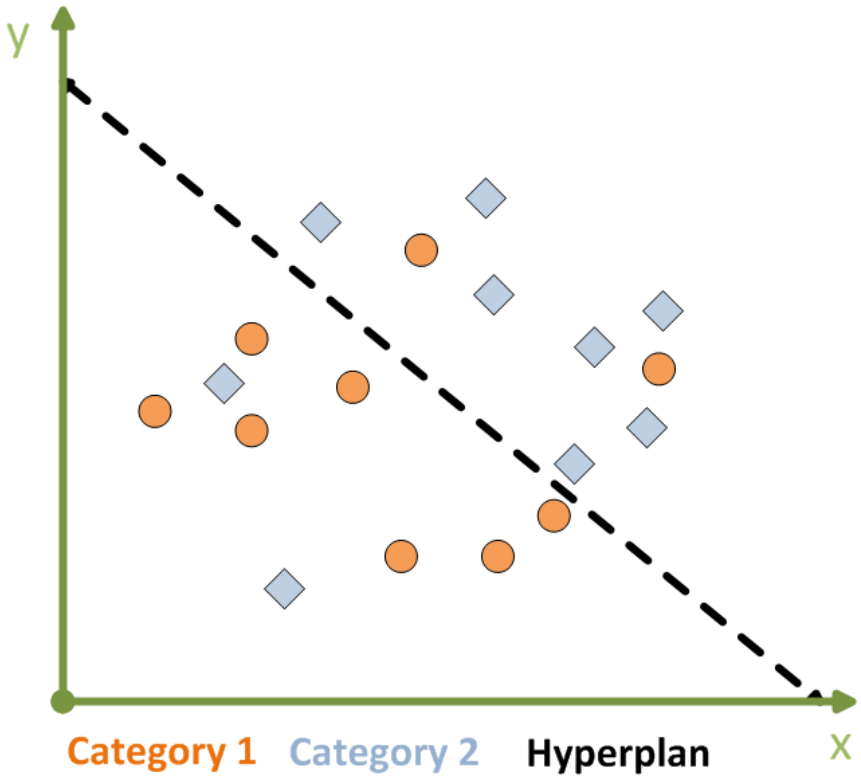

Figure 4.1: Support Vector Machines classification process

Figure 4.1 demonstrates the training process of the Support Vector Machines software *libsvm* made by a certain type of classifier. Depending on that type, the drawn hyperplane can be more or less effective in the category separation. However, some pairs can fall into the wrong category. The graphic shows that there are two pairs of each category that are at the wrong place. Nevertheless, the objective at training is to maximize the distance from the hyperplane towards every pair of category 1 and category 2. *Libsvm* provides four types of classifiers also known as kernels: linear, polynomial, radial and

sigmoid. The selection of the type of kernel is reflected on the type of curve. For instance, the kernel used in Figure 4.1 is clearly linear, once the hyperplane has a straight shape. If using, for instance, a radial kernel, that might lead to better separation of the categories, as the curve of the hyperplane might be more rounded. This work used the radial basis, which was the type of kernel that, after several experiments, led to better results.

4.5 Named Entity Recognizers

The framework uses two external recognizers and creates another method for named entity extraction: Stanford NER (Finkel et al., 2005) recognizes English entities. XIP NER (Mamede, 2007) retrieves the Portuguese names. Also for the Portuguese language, an adaptive method was created in order to detect named entities too. These three systems recognize different types of entities. The Stanford NER only recognizes locations, proper names and organizations. The XIP NER retrieves several types of entities like locations, person names, dates, organizations, etc. The adaptive method did not centered in any specific type of entity. In spite of the three recognizers detect different types of entities, that is not an issue once the aim was to detect translated names, regardless of their type.

The adaptive method retrieves entities by the following method: first, only words starting with a capital letter are considered to be part of a named entity. Then, the named entities are composed by groups of words. In order to build those groups, the method opens a possible named entity when a capital word is caught, and closes it when a low case word is detected. At the same time, function words that can be part of the named entities are defined in an external file. For instance, the location Estados Unidos da América (United States of America) contains the function word *da* that starts with a low case letter. By doing so, the system is enabling the possibility of this type of words to be part of named entities. As demonstrated by this real example, it is not rare to appear in a named entities. Nevertheless, in order to consider these words as part of named entities, they have to follow a few rules: they cannot be the only element of the entity and cannot be at the beginning nor at the end of the entity. This makes sense because prepositions like *de* (*of*) or *da* (*of the*) just happen the way that was mentioned by these three rules in the context of named entities. Therefore, these words are externally defined as exceptions, so that the system, despite of these words do not start with capital letter, acknowledge that they also can be included in names as well.

4.6 Cognate Detection

4.6.1 Algorithm

The algorithm for cognate detection is composed by 3 sub-algorithms: preprocessing, training and testing. The following shows the pseudo-code of these three algorithms.

```
procedure preprocess ()
{
    pre = new Array                 //return preprocessed pairs
    for each text in 1..30          //15 economics + 15 politics
    {
        for each source word in source text
        {
            for each target word in target text
            {
                if (raw_word_length(source word, target word, trans_rules) < 4 &
                    raw_lcsm(source word, target word, trans_rules) > 0.4 &
                    raw_lcsr(source word, target word, trans_rules) > 0.33 &
                    raw_jaro_winkler(source word, target word, trans_rules) > 0.55 &
                    raw_sequence_letters(source word, target word, trans_rules) > 2)
                {
                    pre.add (source word, target word)
                }
            }
        }
    }
    return pre
}
```

Figure 4.2: Preprocessing phase of cognate detection algorithm

Figure 4.2 shows that only some similarity measures are used for this step. They are in raw state, that is, they may not be in [0,1]. This step was accomplished using the set of cognates that belonged to each pair of news. The objective is to narrow the corpus by discarding words that are completely unrelated and, theoretically, with no chance to be considered as cognates. Hence, for each similarity measure, the worst score of each similarity measure is extracted from within the cognates, defining thresholds, so that only above those, words would be worthed to be considered. However, those thresholds are not formed *ipsis verbis* from the cognates. The cognates number would have to be much bigger so that thresholds extracted could be unaltered. Instead, based on a trial and error process, the best configuration is achieved. This process maximizes the number of correct cognates and minimizes the wrong ones. Hence, from the whole set of similarity measures, only *word length*, *lcsm*, *lcsr*, *jaro winkler* and *sequence letters* are used for preprocessing. Moreover, only words that begin with the same letter are filtered in

this step.

Next, classification of cognates takes place. Figure 4.3 shows the training process. It only uses those filtered pairs of words resulting from the early preprocessing step.

```
procedure train ()
{
      pre = preprocess ()
      instance = new Array
      instances = new Array
      for each pair in pre              //after select best features
      {
            instance.reset ()
            instance.add ( soundex ( source word, target word, transl_rules ))
            instance.add ( lcsr ( source word, target word, transl_rules ))
            instance.add ( lcsm ( source word, target word, transl_rules ))
            instance.add ( levenshtein ( source word, target word, transl_rules ))
            instance.add ( jaccard ( source word, target word, transl_rules ))

            instances.add (instance)
      }
      train_model = svm_train (instances)
      return train_model
}
```

Figure 4.3: Training phase of cognate detection algorithm

This time, instead of raw data, every similarity measures are normalized in order to produce normalized data. After a feature selection procedure, only measures like *Soundex*, *Lcsr*, *Levenshtein*, *Lcsm* and *Jaccard* are used for training. After using *radial kernel RBF* in *libsvm* in the operation of *svm_train*, a training model is returned.

Finally, Figure 4.4 refers to the testing process where a given text translated into the source and the target languages provides pairs of words which will be tested if they are cognates, and returned as such. This final step uses every similarity measures in order to describe instances of pairs of words.

```
procedure test ()
{
    train_model = train ()
    instance = new Array
    instances = new Array
    cognates = new Array

    for each source word in source text
    {
        for each target word in target text
        {
            instance.reset ()
            instance.add ( soundex ( source word, target word, transl_rules ))
            instance.add ( lcsr ( source word, target word, transl_rules ))
            instance.add ( lcsm ( source word, target word, transl_rules ))
            instance.add ( levenshtein ( source word, target word, transl_rules ))
            instance.add ( jaccard ( source word, target word, transl_rules ))
            instance.add ( dice ( source word, target word, transl_rules ))
            instance.add ( word_length ( source word, target word, transl_rules ))
            instance.add ( jaro_winkler ( source word, target word, transl_rules ))
            instance.add ( identical_words ( source word, target word, transl_rules ))
            instance.add ( sequence_letters ( source word, target word, transl_rules ))
            instance.add ( cmpg ( source word, target word, transl_rules ))

            instances.add ( instance )
            }
        }
    }
    cognates = svm_predict ( instances , train_model)
    return cognates
}
```

Figure 4.4: Testing phase of cognate detection algorithm

4.7 Translation of Named Entities

4.7.1 Algorithm

The algorithm shown in Figure 4.5 states that the primary step evolves retrieving named entities from both source and target languages.

Secondly, the cognate module is applied to every words in the considered test files in order to return every cognates. Finally, the entities already recognized are compared, by checking if they are composed by the cognates retrieved before. If the longest named entity is mostly composed by cognates, both entities are considered to be translation of each other.

```
procedure named_entities_translations( source , target )
{
    sourceNEs = sourceNER( source )
    targetNEs = targetNER( target )
    cognates = test( source , target )
    tnes = new Array

    for each sourceNE in sourceNEs
    {
        n = 0
        for each tokenSource in sourceNE
        {
            for each targetNE in targetNEs
            {
                majority = ceil ( max (|sourceNE| , |targetNE| ) / 2)
                for each tokenTarget in targetNE
                {
                    for each cognatePair in cognates
                    {
                        if ( cognatePair.first = tokenSource &
                            cognatePair.second = tokenTarget)
                            n++
                    }
                }
            }
        }
        if ( n >= majority )
            result.add (sourceNE + targetNE)
    }
    return tnes
}
```

Figure 4.5: Algorithm of Translation of Named entities

5 Evaluation

5.1 Introduction

This Chapter describes the experiments and results used in order to evaluate the system. These experiments are described for both cognate detection and translation of named entities modules. Section 5.2 includes the setup used for the experiments, the preprocessing process used for cognates, the kernel selection in *libsvm*, feature selection and the evaluation of the created model. Section 5.3 also describes the experimental data used for translation of named entities evaluation and contemplates a survey on the used NER systems, by checking their effectiveness. Finally, using those NER tools, this module is also evaluated.

5.2 Cognate Detection

5.2.1 Experimental Setup

5.2.1.1 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 5.1, Table 5.2 and Table 5.3 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* (Chang & Lin, 2001) software package is used for classification of cognates.

Theme	Economics	Politics
Training	15 documents	15 documents
Testing	4 documents	4 documents

Table 5.1: Number of documents used

Theme	Economics	Politics
Training	107809 word pairs	156453 word pairs
Testing	35688 word pairs	41300 word pairs

Table 5.2: Number of compared word pairs

5.2.1.2 Transliteration Rules

For the experiments, transliteration rules between English and Portuguese are used. Table 5.4 shows the complete list of them alphabetically.

5.2.2 Preprocessing

The first preprocessing configuration is based strictly on the scores of the manually annotated cognates present in the training corpus. This configuration only uses a subset of the whole set of similarity measures. These values are the worst coefficients, as from them it is worth to consider words as potential cognates. Table 5.5 shows the coefficients used in this configuration.

The results of the baseline configuration are shown by Table 5.6. The first column represents the number of cognates found. The second shows the number of cognates missed by the used configuration. The column # NCog states the number of filtered pairs that are not cognates. The last column shows the compressing ratio given by the proportion between the number of word pairs obtained versus the total number of word pairs before preprocessing.

The baseline configuration permitted to narrow the training corpus on a proportion of $\frac{4410+432}{264262} = 0.0183$, which is the ratio between the number of pairs of words that result from preprocessing and the number of pairs of words in the training corpus before preprocessing. 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, a trial and error method takes place in order to improve the baseline results, by changing the configuration, increasing 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 is achieved, which is shown by Table 5.7.

This configuration permits to accomplish better results in terms of filtered non-cognates, but the

Theme	Economics	Politics
Avrg words	136 words	168 words
Avrg cognates	13 cognates	17 cognates
Avrg sentences	11 sentences	13 sentences

Table 5.3: Statistics per document

Prefix rules
am>m former>ex k>c k>qu l>n ph>f pr>a s>es sh>x th>t then>ex w>v
Middle rules
b>v b>p bb>b c>g c>z ck>c ct>c ea>i ee>i gh>h ha>a he>e hi>i ho>o k>c k>qu ll>l mm>m nn>n oo>o ou>u pp>p sh>x ss>x t>c t>cc th>d tt>t v>b w>u x>s y>i
Suffix rules
able>avel al>imo ance>ancia ate>ata cy>cia dom>dade en>em ence>encia er>or fy>ficar ian>ia ible>ivel ic>ico ical>ico ify>ar ing>ando ious>oso ise>izar ish>ixe ism>ismo ist>isto ity>idade ive>ivo ize>izar ly>mente ment>mento ness>dade or>or ous>oso ship>ade sion>cao sis>ise tion>cao ty>dade

Table 5.4: Transliteration rules

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 are more important than others for this stage.

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

5.2.3 Support Vector Machines Kernel

In order to choose which *libsvm* kernel to use for classification, an experiment for comparing which kernel provides better predictive skills is done. Four kernels are tested in *libsvm*: radial, polynomial, sigmoid and linear. For that purpose, the training corpus is trained using the same eleven similarity measures for every kernel. The testing corpus is described with the eleven similarity measures as well. Results are shown in Table 5.9.

The linear kernel is the best in terms of *F-measure*, but when comparing *Precision*, it fails when comparing to other kernels. The sigmoid kernel presents a very low *Precision* value, for which is discarded. The radial is the best option. When compared to polynomial, the radial kernel is beaten only by a slight difference in terms of *Precision*, but in *Recall*, the radial is better, a fact that is reflected by the *F-measure* percentage.

Next, classification takes place. A training model is generated with the eleven similarity measures and tested against the testing corpora. Table 5.10 shows the produced results.

Dic	Jacc	WLen	Lcsm	Lcsr	Lev	JWin	Seq
>0.25	>0.142	<5	>0.2	>0.33	<5	>0.559	>1

Table 5.5: Baseline configuration

#Cog found	#Cog missed	#NCog found	Compress ratio
432	11	4410	0.018

Table 5.6: Results of baseline configuration

5.2.4 Feature Selection

In order to improve the results showed by Table 5.10, an experiment was done in order to infer about which similarity measures maximized the training model. Each similarity measure was evaluated in terms of its individual predictive abilities. For that, the training corpus was trained with only one measure at a time, and tested against the testing corpus. The process was extended for every measures. Table 5.11 demonstrates the results for each similarity measure.

As shown by Table 5.11, *Lcsm*, *Levenshtein* and *Dice* are the best measures to use in a model regarding *F-measure*. During this experiment, when other similarity measures were added to the model, *F-measure* decreased. Moreover, by choosing the top measures by *Precision* or *Recall* instead of *F-measure*, results were never better than the obtained by the three mentioned measures.

5.2.5 Cognate Detection Evaluation

After generating the model based on the similarity measures selected in 5.2.4, the model was used on the testing corpus. Table 5.12 shows the result of this module evaluation.

5.3 Translation of Named Entities

5.3.1 Experimental setup

5.3.1.1 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 5.13 and Table 5.14 describe the corpus with more detail.

WLen	Lcsm	Lcsr	JWin	Seq
<4	>0.4	>0.33	>0.55	>2

Table 5.7: Final configuration

#Cog found	#Cog missed	#NCog found	Compress ratio
410	32	2197	0.0098

Table 5.8: Results of final configuration

5.3.2 Individual Skills of the NER Systems

Using the testing corpora, three methods for performing recognition of named entities are tested individually: Stanford NER for English names, and XIP NER and Adpative Method for Portuguese ones. The result is showed in Table 5.15.

Results show that the English NER system is the one that gets better scores. For Portuguese, the XIP NER presents better results when comparing to the Adaptive Algorithm.

5.3.3 Translation of Named Entities Evaluation

Table 5.16 shows that the method Stanford plus Adaptive Method achieves better results.

5.4 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 *Lcsm*, *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 percent-

Kernel	Precision	Recall	F-measure
Radial	58.33 %	62.5 %	60.34 %
Polynomial	58.62 %	60.71 %	59.64 %
Sigmoid	46.15 %	75.0 %	57.14%
Linear	57.48 %	65.17 %	61.08 %

Table 5.9: Kernel evaluation in *Libsvm*

Precision	Recall	F-measure
73.78 %	58.46 %	65.23 %

Table 5.10: Results of the used model without feature selection

age 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.

Similarity Measure	Precision	Recall	F-measure
Lcsm	70.69 %	63.07 %	66.66 %
Levenshtein	59.73 %	68.46 %	63.80 %
Dice	75.58 %	50.0 %	60.18 %
Lcsr	30.02 %	86.15 %	44.53 %
Soundex	37.59 %	76.61 %	49.99 %
Identical words	94.73 %	27.69 %	42.85 %
Jaccard	5.56 %	94.61 %	10.51 %
Cmpg	0.29 %	100.0 %	0.59 %
Jaro Winkler	0.0 %	0.0 %	0.0 %
Sequence Letters	0.0 %	0.0 %	0.0 %
Word Length	0.0 %	0.0 %	0.0 %

Table 5.11: Individual capabilities of the similarity measures

Precision	Recall	F-measure
69.42 %	64.61 %	66.93 %

Table 5.12: Results of the used model after feature selection

Theme	Economics
Testing	19 documents

Table 5.13: Number of documents used

Theme	Economics
Avrg words	136 words
Avrg cognates	13 cognates
Avrg sentences	11 sentences

Table 5.14: Statistics per document

NER System	Precision	Recall	F-measure
Stanford NER	83.90 %	73.73 %	78.48 %
Adaptive Algorithm	64.51 %	72.07 %	68.08 %
XIP NER	76.28 %	66.66 %	71.14 %

Table 5.15: Results of the individual NER methods

Stanford plus Adaptive Method			**Stanford plus Xip**		
Precision	Recall	F-measure	Precision	Recall	F-measure
94.44 %	60.71 %	73.91 %	96.29 %	46.42 %	62.65 %

Table 5.16: Results of Translation of Named Entities module

6 Conclusion and Future Work

6.1 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 *Lcsm*, *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.

6.2 Contributions

The follow shows the main contributions achieved by this work:

– Survey on cognate detection and translation of named entities.

– Construction of a module for detection of cognates based on different similarity measures.

– Utilization of the cognate detection module for building a translation of named entities module.

– Evaluation of both modules.

6.3 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*, the 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*, *presidencial* 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.

Bibliography

Alegria, I., Ezeiza, N., & Fernandez, I. (2006). Named entities translation based on comparable corpora. In *Proceedings of multi-word-expressions in a multilingual context workshop in eacl.*

Arribas-gil, A., Gassiat, E., & Matias, C. (2006). Parameter estimation in pair hidden markov models. *Journal of Statistics*.

Brown, P. F., Pietra, V. J., Pietra, S. A. D., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*, 263–311.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines.* (Software available at `http://www.csie.ntu.edu.tw/\~cjlin/libsvm`)

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *In acl* (pp. 363–370).

Fung, P., & Yee, L. Y. (1998). An ir approach for translating new words from nonparallel comparable texts. In *Coling-acl* (p. 414-420).

Hall, P. A. V., & Dowling, G. R. (1980). Approximate string matching. In *Computing surveys* (pp. 381–402).

Hassan, A., Fahmy, H., & Hassan, H. (2007). *Improving named entity translation by exploiting comparable and parallel corpora.*

Huang, F., & Vogel, S. (2002). Improved named entity translation and bilingual named entity extraction. In *Icmi'02.*

Jiang, L., Zhou, M., Chien, L. feng, & Niu, C. (2007). Named entity translation with web mining and transliteration. In *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 1629–1634).

Koehn, P., & Knight, K. (2002). Learning a translation lexicon from monolingual corpora. In *In proceedings of acl workshop on unsupervised lexical acquisition* (pp. 9–16).

Kondrak, G. (2000). *A new algorithm for the alignment of phonetic sequences.*

Kondrak, G. (2001). Identifying cognates by phonetic and semantic similarity. In *Naacl '01: Second meeting of the north american chapter of the association for computational linguistics on language technologies 2001* (pp. 1–8). Morristown, NJ, USA: Association for Computational Linguistics.

Kondrak, G. (2009). Identification of cognates and recurrent sound correspondences in word lists. In *Traitement automatique des langues et langues anciennes* (pp. 201–235).

Kondrak, G., Marcu, D., & Knight, K. (2003). Cognates can improve statistical translation models. In *In proceedings of hlt-naacl 2003* (pp. 46–48).

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 707.

Mackay, W., & Kondrak, G. (2005). *Computing word similarity and identifying cognates with pair hidden markov models.*

Mamede, N. (2007, Maio). *A cadeia de processamento xip.* L^2F – Laboratório de Sistemas de Língua Falada.

Mann, G. S., & Yarowsky, D. (2001). Multipath translation lexicon induction via bridge languages. In *Naacl*.

Melamed, I. D. (1995). Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. *CoRR*. (informal publication)

Melamed, I. D. (2000). Models of translational equivalence among words. *Computational Linguistics*, *26*, 221–249.

Moore, R. C. (2003). Learning translations of named-entity phrases from parallel corpora. In *In proc. of eacl* (pp. 259–266).

Moreira, C., & Gonçalves, P. (2009). Automatic machine translation using comparable corpora.

Mulloni, A., & Pekar, V. (2006). *Automatic detection of orthographic cues for cognate recognition.*

Nakov, S., & Nakov, P. (2007). Cognate or false friend? ask the web. In *In proceedings of the ranlp 2007 workshop: Acquisition and management of multilingual lexicons.*

Oakes, M. P. (2000). Computer estimation of vocabulary in protolanguage from word lists in four daughter languages. In *Journal of quantitative linguistics, vol. 7, n 3* (p. 233-243).

Rapp, R. (1999). Automatic identification of word translations from unrelated english and german corpora. In *Acl.*

Ratnaparkhi, A. (1998). *Maximum entropy models for natural language ambiguity resolution*. Unpublished doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

Simard, M., Foster, G. F., & Isabelle, P. (1992). Using cognates to align sentences in bilingual corpora. In *in proceedings of the fourth international conference on theoretical and methodological issues in machine translation* (pp. 67–81).

Tiedemann, J. (1999). Automatic construction of weighted string similarity measures. In *Joint sigdat conference on empirical methods in natural language processing and very large corpora* (pp. 213–219).

Uitdenbogerd, S. (2005). *Readability of french as a foreign language and its uses.*

Vossen, N. P., (editor, P. V., & Hirst, G. (1998). *Eurowordnet: A multilingual database with lexical semantic.*

Zobel, J., & Dart, P. (1996). Phonetic string matching: Lessons from information retrieval. In *In proceedings of sigir'96* (pp. 166–172).