Detecting Contradictions in News Quotations

Ricardo Jorge Sá Marques

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pável Pereira Calado
Prof. Bruno Emanuel da Graça Martins

Examination Committee

Chairperson:	Prof. Mário Jorge Costa Gaspar da Silva
Supervisor:	Prof. Pável Pereira Calado
Member of the Committee:	Prof. Luísa Torres Ribeiro Marques da Silva Coheur

November 2015

Abstract

Detecting contradicting statements is a fundamental and challenging natural language processing and machine learning task, with numerous applications in information extraction and retrieval. For instance, contradictions need to be recognized by question answering systems or multi-document summarization systems. In terms of machine learning, it requires the ability, through supervised learning, to accurately estimate and capture the subtle differences between contradictions and for instance, paraphrases. In terms of natural language processing, it demands a pipeline approach with distinct phases in order to extract as much knowledge as possible from sentences. Previous state-of-the-art systems rely often on semantics and alignment relations. In this work, I move away from the commonly setup used in this domain, and address the problem of detecting contradictions as a classification task. I argue that for such classification, one can heavily rely on features based on those used for detecting paraphrases and recognizing textual entailment, alongside with numeric and string based features. This M.Sc. dissertation provides a system capable of detecting contradictions from a pair of affirmations published across newspapers with both a F1-score and Accuracy of 71%. Furthermore, this M.Sc. dissertation provides an assessment of what are the most informative features for detecting contradictions and paraphrases and infer if exists a correlation between contradiction detection and paraphrase identification.

Keywords: Contradiction , Paraphrases , Classification , Entailment

Acknowledgements

I have much to thank several people for their help and availability over the time that this thesis was elaborated.

First of all, I would like to express my gratitude to Professor Pável Calado and Professor Bruno Martins for their patience and for all the inspiration and constant guidance throughout this journey.

I must also thank my parents for their never ending support and giving me the opportunity to attend Instituto Superior Técnico.

Finally, I have to acknowledge every person that marked my time at Instituto Superior Técnico.

Chapter 1

Introduction

This chapter introduces the purpose of this M.Sc. dissertation, motivating the need for the creation of a system capable of detecting contradictions. A detailed definition of the problem is explained with a set of examples to better understand the challenges of contradiction detection. Furthermore, a list of contributions of this work is presented.

1.1 Motivation

Contradiction detection can have numerous applications. An example would be a system for extracting misleading sentences and affirmations in the domain of political discussion, for instance, in the context of major candidate debates. Through sorting and examining quotations from the same author, one can help people form more informed choices. In like manner, contradiction detection could also be applied to intelligence reports, confirming which information may need further verification. In bio-informatics, where the identification of candidate genes based on unique interactions of proteins is widely studied in order to understand the basis of a disease, automatically finding conflicting facts about such interactions would be highly beneficial.

A simple approach for defining a contradiction is: sentences A and B are contradictory if there is no possible world in which A and B are both true. However, for contradiction detection to be useful, we have to relax the problem a little, and a looser definition is needed to fit a more human intuition (Marneffe *et al.*, 2008). Take for instance the following example:

1. Maria bought a car for Ana.

2. Ana bought a car for Maria.

The previous pair can be marked as a contradiction despite the fact that each person may have bought a car to the other. In consequence, a new definition is given: sentences A and B are contradictory if it is extremely unlikely that A and B are true simultaneously.

Contradictions arise from accessible features acting as antonyms, from negation, and from date, time and number mismatches. Additionally, contradictions may also emerge from complex contrast in factive expressions, from text structure, from certain lexical contrasts and from world knowledge (see Table 2.2). Given that contradictions are often phrases about the same named entities or events, we have that one important part of contradiction detection relates to the similar problem of recognizing entailment between phrases or sentences. Take for instance the next examples:

1. Police agents specializing in explosives defused the rockets. Some 100 people were working inside the plant.
2. About 100 people were injured.

This pair of sentences, assuming that they occur within a description for the same event, is contradictory. Defused rockets cannot explode, and therefore they cannot injure people. Here, for a human judge, it is relatively easy to identify the lack of entailment: the first sentence involves no injuries, so the second is unlikely to be entailed. Consequently, detecting contradictions appears to be a harder task than detecting entailments. Contradictions require deeper inferences and model building. While mismatching information between sentences is often a good suggestion of non-entailment, it is not sufficient for contradiction detection, which requires more precise comprehension of the consequences of events, as expressed over textual sentences.

This M.Sc. dissertation is concerned with the detection of contradictions in news quotations. I will breakdown the complex effort that is contradiction detection in text, assuming that attributed quotations are already provided by some other method (Almeida *et al.*, 2014; Quintão, 2014).

1.2 Problem Description

Since a paraphrase is a type of entailment, i.e., bidirectional entailment (see Section 2.1.4), the proposal of this M.Sc. dissertation is to build a system capable of detecting contradictions given a pair of news quotations by the same author. I consider this problem as a classification task, based on features for detecting paraphrases and recognizing textual entailment next to string and numeric features. Since I intend the system to be language independent and to compare the problem of detecting paraphrases and detecting contradictions, the system was tested against

two datasets: (1) a Portuguese language dataset that was created in the context of this work based on a database provided by *sapo.pt*, which contains news quotations from an author, and (2) the *Microsoft Research Paraphrase Corpus*[1], a well known dataset used for the task of paraphrase identification.

1.3 Contributions

The main contribution of this work consists in building a language independent system capable of detecting contradictions given a pair of quotations from the same author. Beyond building this system, I yield an evaluation of what are the most informative features in detecting contradictions and compare these features against the most informative in detecting paraphrases.

The contributions can be summarized as follows:

- A system capable of detecting contradictions given a pair of quotations from the same author;
- An extensive study of the most informative features for contradiction detection;
- An extensive study of if the most informative features for detecting contradiction are the most informative for paraphrase detection;
- A system for paraphrase identification that competes with previous state-of-the-art related works.

1.4 Document Organization

The remaining contents of the dissertation are organized as follows. Chapter 2 presents fundamental concepts in order to advance to a more detailed explanation of the current state-of-the-art. Additionally, it presents the most important related work, discussing several different implementations and approaches for entailment, paraphrase and contradiction detection. Chapter 3 details the proposed model. Chapter 4 presents the studies carried out in the context of validation proposal of the dissertation. Finally, Chapter 5 sums up the work and discusses possible future work.

[1] http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/

Chapter 2

Fundamental Concepts and Related Work

This chapter establishes and explains the fundamental concepts in which this work is embedded, as well as previous state-of-the-art related works considered the most significant for the context of this M.Sc. dissertation.

2.1 Fundamental Concepts

This section presents the main concepts involved in understanding the topics discussed in this paper. Section 2.1.1 describes approaches for representing text documents. Section 2.1.2 addresses how can we classify documents using machine learning approaches. Section 2.1.3 describes the steps in a natural language processing pipeline that are more relevant for my work. Section 2.1.4 describes methods for the detection of paraphrases and textual entailment. Finally, Section 2.1.5 addresses the types of contradictions that emerge over natural language.

2.1.1 Representation of Text Documents for Computational Analysis

With the rapid growth of on-line information, information extraction and text categorization have become some of the key techniques for handling and organizing text data. Before any of these algorithms can be applied, we first need to build an adequate representation for the textual documents.

The vector space model is a classical approach to compute document similarities based on index

terms, representing documents as vectors of identifiers, normally corresponding to index terms. Each document j is thus represented as:

$$d_j =< w_{1,j}, w_{2,j}, ..., w_{t,j} > \tag{2.1}$$

Each dimension corresponds to a separate term from the vocabulary used throughout a collection of documents. If a term occurs in the document, its value in the vector is non-zero. Several different ways for computing these values, also known as term weighting schemes, have been developed. The definition of term also depends on the application. Typically, terms are single words, keywords, or longer phrases. If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus).

A common term weighting scheme is the TF-IDF (Manning *et al.*, 2008) approach. This measure calculates how important a word is in a document from a document collection. The motivation for this scheme is that there are words occurring in each document that are also very frequent in many other documents and that, thus, should not contribute to the comparison process with the same weight of words that are more specific to some questions. This measure is a combination of two methods: Term Frequency (TF) and Inverse Document Frequency (IDF).

There are many ways to calculate the term frequency component $tf_{t,d}$. The simplest one is by a simple count, in which $tf_{t,d}$ is defined as the number of times that t occurs in d. However, relevance may not increase proportionally with term frequency. A common variant involves using a log-frequency weighting:

$$w_{t,d} = \begin{cases} 1 + \log(tf_{t,d}) & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Document frequency df_t is instead, the document frequency of a term t, i.e., the number of documents that contain t. Inverse Document Frequency can be defined as follows:

$$idf_t = \log(N/df_t) \tag{2.3}$$

The parameter N refers to the number of documents in the considered collection. There is only one idf value for each term t in a collection of size N (i.e., these values are not specific to particular documents from the collection), and as might expected, idf gives more importance to terms that appear more rarely in the collection.

With these two methods combined, the TF-IDF scheme is simply defined as:

$$w_{t,d} = (1 + \log(tf_{t,d})) \times \log(N/df_t) \tag{2.4}$$

The TF-IDF scores increase with the number of occurrences within a document, and it also increases with the rarity of the term in the collection.

Let us denote by $\overrightarrow{V}(d)$ the vector from document d, with one component in the vector for each dictionary term, and weighted according to the aforementioned TF-IDF scheme. The set of documents in a collection may be viewed as a set of vectors in a vector space, in which there is one axis for each term. This representation loses the relative ordering of the terms in each document. To compensate for the effect of document length, the standard way of quantifying the similarity between two documents d_1 and d_2 is to compute the cosine similarity of their vector representations $\overrightarrow{V}(d_1)$ and $\overrightarrow{V}(d_2)$, which is given by:

$$\text{sim}(d_1, d_2) = \frac{\overrightarrow{V}(d_1) \cdot \overrightarrow{V}(d_2)}{||\overrightarrow{V}(d_1)|| \times ||\overrightarrow{V}(d_2)||} \tag{2.5}$$

The numerator represents the dot product of the vectors $\overrightarrow{V}(d_1)$ and $\overrightarrow{V}(d_2)$ while the denominator is the product of their Euclidean norms. The dot product $\overrightarrow{x}.\overrightarrow{y}$ of two vectors is defined as $\sum_{i=1}^{M} x_i y_i$, while the Euclidean norm of a vector $\overrightarrow{x}$ is given by:

$$\sqrt{\sum_{i=1}^{M} \overrightarrow{x_i}} \tag{2.6}$$

The effect of the denominator in Equation 2.5 is thus to length-normalize the vectors $\overrightarrow{V}(d_1)$ and $\overrightarrow{V}(d_2)$ to unit vectors $\overrightarrow{v}(d_1) = \frac{\overrightarrow{V}(d_1)}{||\overrightarrow{V}(d_1)||}$ and $\overrightarrow{v}(d_2) = \frac{\overrightarrow{V}(d_2)}{||\overrightarrow{V}(d_2)||}$.

2.1.2 Machine Learning and Document Classification

Given a predefined set of classes, the goal of automatic classification is to determine to which classe(s) a given object belongs to, on the basis of a training set of data containing observations (or instances) whose category membership is already known. A classification technique (or classifier) is thus a systematic approach to building classification models from an input data set (Michie *et al.*, 1994). Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set that describes objects (i.e., the terms considered when building a vector space model representation, in the case where instances are textual documents) and the class label of the input data. The model generated by a learning algorithm

should both fit the input data well, and correctly predict the class labels of objects it has never seen before.

There are numerous classification techniques, based on different types of machine learning approaches. For instance, decision trees solve a classification problem by asking a series of carefully crafted questions about the attributes of a test record (Kotsiantis, 2013). Each time an answer is received, a follow-up question is applied in order to reach a conclusion about the class label of the record. It is called a decision tree because the series of questions and answers can be organized and represented in the form of a tree, which is a hierarchical structure consisting of nodes and directed edges.

In a decision tree, each terminal node is assigned to a class label. The non-terminal nodes contain the questions, i.e., the attribute test conditions to separate records that have different characteristics. Once the tree is constructed, classifying a test record is very straight-forward. Starting form the root node, one simply applies the test conditions to create a path down the tree to a terminal node, and to the class label associated with that node. The aim of the decision tree learning algorithm is to find a small tree consistent with the training data, by recursively choosing the most significant attribute as a root of a (sub-tree).

The underlying principle of choosing the most significant attribute is to pick the attribute that goes as far as possible toward providing an exact classification. A perfect attribute divides the examples into sets, each of which are all positive or all negative and thus will be leaves of the tree. To rank an attribute, one can use the notion of entropy. In information theory, entropy is the average amount of information contained in each received message. Here, message stands for an event or sample drawn from a distribution or data stream. Entropy thus characterizes our uncertainty about the source of information. The higher the entropy, the more information content. Entropy can be computed as follows:

$$\mathrm{E}(S) = \sum_{i=1}^{n} -\mathrm{P_i} \log \mathrm{P_i}(S) \tag{2.7}$$

In the formula, P_i is the probability of class i. Notice however that entropy only computes the quality from a single subset of examples. Alternatively, one can compute the weighted average over all sets resulting from the split. When an attribute A splits the set S into m subsets S_i, we compute the average entropy and compare the sum to the entropy of the original set S. This allows us to obtain the information gain of attribute A and, the procedure can be formally defined as:

$$\mathrm{Gain}(S, A) = \mathrm{E}(S) - \sum_{i=1}^{m} \frac{|S_i|}{|S|} \times \mathrm{E}(S_i) \tag{2.8}$$

Linear classifiers instead are based on a hyperplane, in the space of the instances, which separates positive from negatives examples. Think of training examples as points in a *d*-dimensional space. Each dimension corresponds to one feature. A hyperplane is a generalization of a straight line to > 2 dimensions. A hyperplane contains all the points in a *d*-dimensional space satisfying the following equation:

$$w_1 x_1 + w_2 x_2, ..., + w_d x_d + w_0 = 0 \tag{2.9}$$

Each coefficient w_i can be thought of as a weight on the corresponding feature. The vector containing all the weights $w = (w_0, ..., w_d)$ is the called the weight vector. The goal of the learning process is to come up with a good weight vector w, using training examples to guide the search process.

2.1.3 Computational Natural Language Processing

Natural Language Processing (NLP) is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis (e.g., morphology, syntax, semantics and pragmatics) for the purpose of achieving human-like language processing for a range of tasks or applications. We can view NLP as a pipeline with various stages, in order to extract as much knowledge as possible. Some of the typical stages are described in the following subsections.

2.1.3.1 Sentence Splitting and Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping the sequence into pieces, called tokens (Jurafsky & Martin, 2009). At the same time it may be useful to throw away certain pieces, such as punctuation. The issues of tokenization are language-specific. Each language presents some new issues. For instance, French has a variant use of the apostrophe for a reduced definite article before a word beginning with a vowel (e.g., l'ensemble) and has some uses of the hyphen with post-posed clitic pronouns in imperatives and questions (e.g., donne-moi). Getting the first case correct will affect the correct indexing of a fair percentage of nouns and adjectives: you would want documents mentioning both *l'ensemble* and *un ensemble* to be indexed under the term ensemble.

In general, sentence splitting and tokenization methods work by building a binary classifier (Islam *et al.*, 2007), which can be based on a sequence of hand written rules, regular expressions, or on machine learning. The principle behind this classifier is to decide, for example, if a period is a part of a word (abbreviation) or if it is a sentence boundary. While many state-of-the-art methods for

both sentence splitting and tokenization are based on hybrid approaches, a popular and simple way to address this task is with a decision tree classifier, a particular model that was explained earlier.

2.1.3.2 Parts-of-Speech Tagging

Parts-of-Speech (POS) tagging is a technique for assigning each word of a text to an appropriate morpho-syntactic category, such as noun, verb or adjective. The significance of parts-of-speech for language processing relates to the large amount of information that these tags give about a word and its neighbors.

There are different types of approaches for parts-of-speech tagging, where the notable ones are rule-based and stochastic approaches. Rule-based approaches generally involve a large dictionary and use hand-crafted rules to tag a text. Usually, there are two main steps: (1) with the help of a dictionary, one tags each word with all its possible labels, and (2) with the help of a set of (disambiguation) rules, one disambiguates these labels into a single one. These rules can specify, for instance, that a word following a determiner and an adjective must be a noun. Of course, this means that the set of rules must be properly written and checked by human experts.

In the stochastic approach, the objective is to choose the best sequence of tags, $T = t_1, t_2, ..., t_n$ for a certain sequence of words $W = w_1, w_2, ..., w_n$. The goal is thus to calculate $T \in \gamma$ that maximizes $\mathrm{P}(T|W)$, being γ the set of all possible tag sequences for W:

$$\mathrm{argmax}_{T\in\gamma}\mathrm{P}(T|W) = \mathrm{argmax}_{T\in\gamma}\mathrm{P}(W|T \times \mathrm{P}(T)) \tag{2.10}$$

Hidden Markov Models (HMM) are commonly used for stochastic parts-of-speech tagging, corresponding essentially to a Markov Process with non-observable (hidden) states. Basically, HMMs assume that the probability of occurrence of a word only depends on its tag (that is, it does not depend on other words) and the probability of a tag only depends on the previous tag. Due to this, the following approximations can be done:

$$\mathrm{P}(W|T) \approx \prod_{i=1}^{n} \mathrm{P}(w_i|t_i) \tag{2.11}$$

$$\mathrm{P}(T) \approx \prod_{i=1}^{n} \mathrm{P}(t_i|t_{i-1}) \tag{2.12}$$

Therefore, the assignment of parts-of-speech tags can be made through:

$$\text{argmax}_{T\in\gamma}\text{P}(W|T) \times \text{P}(T) \approx \text{argmax}_{T\in\gamma} \prod_{i=1}^{n} \text{P}(w_i|t_i)\text{P}(t_i|t_{i-1}) \tag{2.13}$$

Through simple counts over the training data, one can easily estimate the required parameters:

$$\text{P}(t_i|t_{i-1}) = \frac{\text{C}(t_{i-1} \rightarrow t_i)}{\text{C}(t_{i-1})} \tag{2.14}$$

$$\text{P}(w_i|t_i) = \frac{\text{C}(t_i \uparrow w_i)}{\text{C}(t_i)} \tag{2.15}$$

For efficiently finding a sequence of states $T = t_1, t_2, ..., t_n$ such that the probability of observing a sequence $W = w_1, w_2, ..., w_n$ is greater than for any other state sequence, one can use the Viterbi algorithm, which is based on dynamic programming. Algorithm 1 presents the pseudo-code for the Viterbi procedure, considering that T is a state sequence of labels $t_1, ..., t_n$, n is the number of words given in a sequence $w_1, ..., w_n$, SS is an array $T \times n$ that keeps track of the best score of the best sequence at iteration t_i, BP is an array of $T \times n$ dimensions that registers the best probability of a transition from the previous to the current state, and C is a vector of size n that registers the best sequence of labels.

There is also a well-known hybrid approach called the transformation based tagger (Brill, 1992). Like a rule-based tagger, it is based on rules which determine when an ambiguous word should have a given tag. Like stochastic taggers, this approach has a machine-learning component: the rules are automatically induced from a previously tagged training corpus. This method takes the following steps: (1) each word is tagged with the most frequent label; (2) transformational rules are learned from a labeled corpus (rules that replace labels), and (3) transformational rules are applied until some stop condition is reached.

2.1.3.3 Named Entity Recognition

The task of named entity recognition (NER) concerns with labeling particular phrases in a text into predefined categories, such as names of persons, organizations or locations. There are two general approaches to NER: hand-made rules and machine learning. Rule-based approaches focus on extracting names using entity dictionaries together with human-made rules. Generally, these systems consist of a set of patterns using morpho-syntactic (e.g., parts-of-speech), syntactic (e.g., word precedence) and orthographic features (e.g., capitalization), in combination with

Algorithm 1 Viterbi decoding for an HMM

$i \leftarrow 1$

while $i < N$ **do**
    $\mathrm{SS}(i, 1) = \mathrm{P}(w_1|t_i) \times \mathrm{P}(t_i|start)$
    $\mathrm{BP}(i, 1) = 0$
    $i++$

$m \leftarrow 2$

while $m < n$ **do**
    $i \leftarrow 1$

    while $i < N$ **do**
        $\mathrm{SS}(i, m) = \max_{j=1,\ldots,N} \mathrm{SS}(j, m-1) \times \mathrm{P}(t_i|t_j) \times \mathrm{P}(w_t|t_i)$
        $\mathrm{BP}(i, m) = j$ that resulted in the maximum score
        $i++$

    $m++$

$\mathrm{C}(i) = i$ that maximizes $\mathrm{SS}(i, n)$
$i \leftarrow n - 1$

while $i \rightarrow 1$ **do**
    $i--$
    $\mathrm{C}(n) = \mathrm{BP}(\mathrm{C}(i + 1), i + 1)$

dictionaries. This approach has better results for restricted domains, and it is capable of detecting complex entities that learning models have difficulty with. However, rule-based systems lack portability and robustness.

In machine learning-based systems, the NER task is converted into a classification problem (Nadeau & Sekine, 2007). For example, NER can be seen as a task of giving tags to words, much like POS tagging. NER systems can be developed through HMMs, and the Viterbi algorithm can again, be used to find the best sequence of tags for a sequence of words. Most statistical NER systems rely on the IOB encoding of phrases as tags over individual words, where the classifiers are trained to recognize the beginning(B), the inside(I) and the outside(O) of an entity name. Consider the following example:

José/**B** Manuel/**I** works/**O** with/**O** Maria/**B**./**O**

In the example, there are two entities namely *José Manuel* and *Maria*. The first entity is composed by more then one word, and thus the first word is assigned to a (B)eginning tag and other

words are assigned to an (I)nside tag. When the entity is composed by only one word, it is always assigned a (B)eggining tag. The other words have a tag O(ther) because they are not part of entities.

NER models that are slightly more sophisticated than HMMs can use, in combination with the previous encoding approach, features corresponding to markers which are specific to particular entity classes, such as (i) entity names are often proper nouns, appearing capitalized in the text, (ii) names often proceeded by titles such as *Mr*, or (iii) locations can appear in groups and separated by the commas surrounding them (e.g., Sintra, Portugal). Discriminative models such as Conditional Random Fields (Lafferty *et al.*, 2001) can naturally leverage different types of features for named entity recognition.

2.1.3.4 Syntactic Parsing According to Dependencies or Constituents

Parsing consists of finding the structure of a sentence as expressed by a set of directed links between words and/or phrases (Gómez-Rodríguez *et al.*, 2008). The grammar for a given natural language is ambiguous and typical sentences have multiple possible analyses. In fact, for a typical sentence, there may be thousands of potential parses (most of which will seem completely nonsensical to a human). To parse natural language data, first we must agree on the grammar to use, namely what type of sentence structure it will be used. Two popular choices correspond to either constituency or dependency parsing. Moreover, alternatively to these two main parsing formalisms, some NLP applications instead rely on shallow parsing of the text, in which similar techniques to these employed to POS tagging or NER are used to split sentences into chunks corresponding to different types of phrases (e.g., noun phrases, verbs phrases, etc). This approach is typically referred to as NP-chunking.

Constituency parsing tries to find a division of the sentence into segments (constituents) which are then broken up into smaller constituents. Constituency is based in a one-to-more (or one-to-one) relation between the constituents, which means that an element in a sentence can have multiple relations according to the grammar.

Dependency parsing instead focuses on finding dependencies, i.e., only one-to-one directed links between words. Dependency parsing has several advantages in the context of a task such as the one I intend to address: (1) there are no restricted grammar rules, (2) dependency links are close to the semantic relationships needed for a next step of interpretation and, (3) the dependency tree contains one node per word, instead of mid-level nodes as in constituent trees, making the task of parsing more straightforward. The output of a dependency grammar is a dependency

graph $G = (V, A)$ where V are nodes (elements of the sentence) and A are arcs (dependencies). The typical classes of dependencies are semantic, morphological and syntactic. Semantic dependencies are shown in form of a predicate with its arguments, and the arguments are dependent of that predicate. Morphological dependencies instead, capture relations between words, under the idea that parts of words depend of other words. In the example: *Manuel writes* there is a relation where the singular subject *Manuel* requires the suffix *s* in the word *writes*. Finally, syntactic dependencies can be mapped into syntactic functions (grammatical relations). These depend on an inventory of functions such as subject, object, determiner, attribute, predicative, etc. Dependency relation arcs $< h, m >$ are head-to-modifier, and the source word h is called the *head* while the target word m is called the *modifier*.

2.1.4 Detection of Paraphrases and Textual Entailment

Entailment can be defined as a relationship between two natural language units (e.g., two phrases, or two sentences) where the truth of one requires the truth of the other. We can say that a sentence A entails a sentence B if and only if whenever A is true, B is also true. Paraphrases are a special type of entailment, i.e., bidirectional entailment. A paraphrase is a kind of semantic equivalence responsible for the interconnection of statements, i.e., by replacing grammatical classes and variables unchanged between lexical and syntactic structures. Next, we can see a simple example that shows the entailment and paraphrase relationships.

1. Maria saw a bear.
2. Maria saw an animal
3. My mother is in front of my father.
4. My father is behind my mother.

In the pair of sentence 1) and 2), the entailment relation works in only one direction. If Maria saw a bear, then she necessary saw an animal. However if she saw an animal, she could have seen a bear, but not necessarily. When there is only one-way entailment, the sentences are not true paraphrases of each other. The pair with the sentences 3) and 4) behaves somewhat differently. Because of the semantic relationship between *front of* and *behind*, we have a situation of mutual entailment between the sentences. These sentences are paraphrases of each other.

The availability of shared tasks focused on the problem of recognizing textual entailment (RTE) has fostered the experimentation with a number of data-driven approaches applied to semantics (Dagan *et al.*, 2009). Specifically, the availability of RTE datasets for training made it possible

to formulate the problem as a classification task, where features are extracted from the training examples and then used by machine learning algorithms in order to build a classifier, which is finally applied to the test data to classify each pair of sentences/phrases as either entailed or not. Most recent approaches use machine learning algorithms (e.g., linear classifiers) with a variety of features, including lexical, syntactic and semantic matching features, based on document co-occurrence counts, first-order syntactic rewrite rules, and based on extracting the information gain provided by lexical measures.

Different approaches have been produced along the years with the some sort of combination of the features described above. A simple approach is the bag-of-words strategy, in which the comparison of a given sentence pair is calculated using a cosine similarity score. If the score is greater than a threshold value (determined manually or learned through a supervised training data), the sentences are classified as paraphrases. Sometimes simplifying the data set is better. For instance Zhang & Patrick (2005) proposed a classification method were the sentence pair is simplified into canonical forms (through a set of rules) such as changing sentences from passive to active voice. Using a decision tree learning method, the authors exploit lexical matching features such as the edit distance between the tokens. In addition, to lexical matching features, authors like Kozareva & Montoyo (2006) or Ul-Qayyum & Wasif (2012) proposed classification approaches using a combination of lexical and semantic features and heuristics (e.g., negation patterns) to aid in the detection of false paraphrases.

Methods used in most previous approaches work at the sentence level, but as paraphrases regularly involve synonyms or other forms of word relatedness, authors like Mihalcea *et al.* (2006) or Fernando & Stevenson (2008) developed word-level similarity methods to determine if a sentence pair is paraphrase. These methods are based in word-to-word similarity measures (e.g., knowledge-based metrics which use WordNet). Methods based in alignments (e.g., summarization and machine translation metrics, based on word alignment are detailed in Section 2.2.1.1 are also commonly used. Table 2.1 summarizes different previous approaches to the task of paraphrase detection, as evaluated over the well-known Microsoft Research Paraphrase corpus of sentences Dolan *et al.* (2004).

2.1.5 Types of Contradictions in Natural Language

Contradictions may arise in a number of different natural language constructions. Some are very easy to detect, while some others are more complex, even for humans. As defined by Marneffe *et al.* (2008), we can categorize contradictions according to two basic types: (1) those occurring via antonyms, negation, and date/number mismatches, and (2) contradictions arising

Reference	Brief Description	Acc.	F_1
Zhang & Patrick (2005)	Lexical features after text canonicalization	0.703	0.795
Mihalcea *et al.* (2006)	Combination of word-to-word similarities	0.703	0.813
Rus *et al.* (2008)	Graph subsumption	0.706	0.805
Qiu *et al.* (2006)	Sentence dissimilarity classification	0.720	0.816
Islam & Inkpen (2006)	Combination of semantic and string similarity	0.726	0.813
Blacoe & Lapata (2012)	Semantic spaces from word clustering	0.730	0.823
Fernando & Stevenson (2008)	Wordnet metric and vector similarity	0.741	0.824
Ul-Qayyum & Wasif (2012)	Semantic heuristic features	0.747	0.818
Finch *et al.* (2005)	Combination of MT evaluation measures	0.750	0.827
Wan *et al.* (2006)	Dependency-based features	0.756	0.830
Das & Smith (2009)	Product of experts, using dependency parsing	0.761	0.827
Kozareva & Montoyo (2006)	Combination of lexical and semantic features	0.766	0.796
Socher *et al.* (2011)	Recursive auto-encoder with dynamic pooling	0.768	0.836
Madnani *et al.* (2012)	Modern machine translation metrics	0.774	0.841

Table 2.1: Comparison of different methods that have been previously proposed for detecting paraphrases.

from the use of factive or modal words, structural and subtle lexical contrasts, as well as world knowledge. The first category can be marked as easy because these types of contradictions can be caught without full sentence comprehension. Moreover, little external information, required for semantic analysis, is needed to gain a broad coverage of antonyms, negations and date/numeric mismatches. Processing these contradictions only involves a closed set of words or data that can be obtained form existing resources, such as WordNet[1]. Alternatively, contradictions in the second category are more difficult to detect automatically, because they require a deeper analysis and more external information to build a precise model of sentence semantics, i.e., one needs to understand the meaning of the sentence in a given context.

Taking into account the problems describe before, let us see one example to show the underlying complexity that is involved. For instance, in Example 6 from Table 2.2, it is necessary to learn that *X said Y did nothing wrong* and *X accuses Y* are incompatible. We can start to learn such inferences with the types of methods used for recognizing paraphrases and textual entailment, but successfully extending these techniques to learn incompatible phrases poses a challenge, mostly because of data sparseness.

[1] http://wordnet.princeton.edu/

ID	Type	Text	Hypothesis
1	Antonym	A capital punishment is a catalyst for more crime	Capital punishment is a deterrent to crime.
2	Negation	A closely divided Supreme Court said that juries and not judges must impose a death sentence	The Supreme Court decided that only judges can impose the death sentence.
3	Numeric	The tragedy in Iraq has already kill 20 soldiers	An investigation shows that 19 have died already in Iraq.
4	Factive	The bombers had not managed to enter the embassy	The bombers entered the embassy.
5	Structure	Jacques Santer succeeded Jacques Delors as president of the European Commission in 1995	Delors succeeded Santer in the presidency of the European Commission.
6	Lexical	The Canadian parliament's Ethics Commission said former immigration minister, Judy Sogro, did nothing wrong and her staff had put her into a conflict of interest.	The Canadian parliament's Ethics Commission accused Judy Sogro.
7	Word Knowledge	Microsoft Israel, one of the first Microsoft branches outside the USA, was founded in 1989	Microsoft was established in 1989.

Table 2.2: Examples for different contradiction types.

2.2 Related Work

In this section, I present previous work concerning three topics: (1) recognizing textual entailment, (2) detecting paraphrases, and (3) contradiction detection. For each one of these topics I describe the techniques used by previous authors of some of the most important works, presenting examples and summarizing the obtained results. Some of the works are evaluated against the RTE dataset provided in the context of the PASCAL[1] challenge. The dataset is composed by a pair of text-hypothesis and whether the pair has an entailment relation from the first to the second text, or not. In order to provide an opportunity for presenting and comparing different approaches for modeling textual entailment, PASCAL has organized challenges, each one with new datasets content. An example of a tuple one such dataset is shown in Table 2.3.

2.2.1 Detecting Paraphrases

Previous works used approaches based on lexical, word similarity features and based on alignment metrics from the area of summarization and machine translation, together with dependency features, as described in Section 2.1.4. In this section I will show that more recent metrics developed for machine translation (MT) are advantageous for paraphrase identification.

[1] http://pascallin.ecs.soton.ac.uk/

Text	Hypothesis	Entailment
Eyeing the huge market potential, currently led by Google, Yahoo took over search company Overture Services Inc last year	Yahoo bought Overture.	True.

Table 2.3: Example of one RTE dataset tuple.

2.2.1.1 Re-examining Machine Translation Metrics for Paraphrase Identification

Madnani *et al.* (2012) proposed an approach only based in MT metrics. Although the notion of using MT metrics for the task of paraphrase identification is not novel (Finch *et al.*, 2005), the author merits from a thorough re-assessment of these metrics conjointly with the creation of new metrics in order to achieve the best results so far. The author exploits 8 MT metrics:

1. **BLEU.** (Papineni *et al.*, 2002) introduced this metric which is based on the amount of n-gram overlap, for diverse values of n, between the system output and the reference translation together with a penalty for translations that might be too short. The author uses as values for n, 1 to 4, as different features for the system.

2. **NIST.** Based on the BLEU metric, instead of simply calculating $n-$gram precision adding equal weight to each one, it computes how informative a singular n-gram is relying on its frequency (Doddington, 2002). The author uses as values for n, 1 to 5 as different features for the system.

3. **TER.** Translation Edit Rate computes the number of steps that a human would take to execute the transformation from the system output to the reference translation (Snover *et al.*, 2006).

4. **TERp.** TER-plus completes the core TER metric contributing with additional edit operations based on stemming, synonymy and paraphrasing (Snover *et al.*, 2009).

5. **METEOR.** Unlike BLEU which uses only precision, METEOR benefits from a combination of both precision and recall known as the harmonic mean. Moreover, it profits from several features based on synonynmy and stemming (Banerjee & Lavie, 2005).

6. **SEPIA.** Is syntactically-aware metric based on a unification of 2 scores: (1) $n-$gram precision similar to BLEU and (2) span extended structural bi-gram precision (Habash & Elkholy, 2008).

7. **BADGER.** Is language independent translation metric based on compression and information theory. The algorithm used is the Normalized Compression Distance (NCD) applying

Metric	Acc.	F1
MAXSIM	67.2	79.4
BADGER	67.6	79.9
SEPIA	68.1	79.8
TER	69.9	80.9
BLEU	72.3	80.9
NIST	72.8	81.2
METEOR	73.1	81.0
TERp	74.3	81.8

Table 2.4: Classification results with individual metric as features.

the Burrows Wheeler Transformation (BWT). The BWT facilitates taking into account common sentence contexts regardless of the size limit of these contexts (Parker, 2008).

8. **MAXSIM** Computes the maximum similarity metric (based on precision and recall) between items across to sentences. For the similarity metric, maps each word in one sentence to at most one word in the other sentence (Chan & Ng, 2008).

The authors run the system adopting a meta-classifier, i.e., a classifier that does not implement a classification algorithm on its own, but uses other classifiers to do the actual work. In addition, the meta-classifier adds another processing step that is performed before the actual base-classifier sees the data. The authors used a meta-classifier that measures the average of unweighted classifiers to make its final decision. 3 classifiers were used: (1) Logistic Regression, (2) Sequential minimal optimization implementation of a SVM, and (3) a lazy, instance-based classifier that extends the Nearest Neighbor algorithm.

The authors evaluates the system against the Microsoft Research Paraphrase Corpus. In order to analyze the importance of each metric, the authors carry out a test in which each metric is the only feature. The results can be seen in Table 2.4. The author also tested multiple metrics as features, considering the combination of BLEU, NIST and TER as the baseline. The result can be seen in Table 2.5. The comparison between this work and previous is already shown in Table 2.1.

Features	Acc.	F1
Baseline	74.1	81.5
+ TERp	75.6	82.5
+ METEOR	76.6	83.2
+ BADGER	77.0	83.7
+ Others	77.4	80.9

Table 2.5: The top 3 performing MT metrics.

2.2.2 Recognizing Textual Entailment

Recognizing textual entailment (RTE) has a direct link with the detection of contradictions considering that a major task in RTE is, for instance, identifying same named entities or events between two sentences. Several approaches have been presented, using different types of similarity metrics singularly, or using a combination of lexical, syntactic or semantic features. RTE has many applications, such as question answering, information extraction and summarization. I will present next some of the previously proposed state-of-the-art approaches, which mostly use lexical and syntactic methods.

2.2.2.1 Lexical and Syntactic Entailment Detection

Pakray *et al.* (2011) describe a lexical and syntactic approach for solving the textual entailment problem. This method results from the composition of several modules, namely: a pre-processing module, a lexical similarity module and a syntactic similarity module.

The pre-processing module has the goal of cleaning the input text that is received, clearing and replacing some noisy tokens.

The lexical similarity module uses several different mechanisms in order to ease the matching. These include:

1. **WordNet based unigram match.** The various unigrams in the hypothesis, for each text-hypothesis pair are checked for their presence in the text. WordNet synsets (sets of synonyms) are identified for each of the unmatched unigrams in the hypothesis. If any synset for the hypothesis unigram matches with any synset of a word in the text, then the hypothesis unigram is considered as a WordNet based unigram match. If the output value of this method is equal or greater than 0.75, then the pair text-hypothesis is considered as a possible entailment and is assigned the value of 1, otherwise the pair is assigned a value of 0.

2. **Bi-gram matching.** Each bi-gram in the hypothesis is searched for a match in the corresponding text. The output value is the total number of matched bi-grams in a pair, divided by the number of hypothesis bi-grams. If this value is equal or greater than 0.5 then the pair is considered as entailed and is assigned the value of 1, otherwise it is assigned a value of 0.

3. **Longest common subsequence.** If the relative length of the longest common subsequence (LCS) between the pair text-hypothesis is equal or greater than 0.8, then the pair

is considered as a possible entailment and is assigned the value of 1, otherwise the pair is assigned a value of 0.

4. **Skip-grams.** A skip-gram is any combination of n words in the order as they appear in a sentence, allowing arbitrary gaps. In the present work, only 1-skip-bi-grams are considered, i.e., bi-grams with one word gap between a sequence of two words in a sentence. The output value of the method is calculated by $\mathrm{skip\_gram}(T, H)/n$, where the numerator is the number of common 1-skip-grams found in T and H, and where n is the number of 1-skip-bi-grams in H. If the value of this method is equal or greater than 0.5 then the pair is considered as entailed and is assigned the value of 1, otherwise the pair is assigned a value of 0.

5. **Stemming.** Stemming is a mechanism for transforming words in a way that if they represent the same meaning they are captured by the same token. Each word in the text and hypothesis pair is stemmed, using the stemming function provided with WordNet 2.0. If s_1 is the number of common stemmed unigrams between text and hypothesis, and if s_2 is the number of stemmed unigrams in the hypothesis, then the output value is defined as s_1/s_2. If the value is equal or greater than 0.7 then the pair is considered as entailed and is assigned the value of 1, otherwise the pair is assigned the value of 0.

The syntactic similarity module is based on the Stanford parser[1], which normalizes data from the corpus of text and hypothesis pairs, performs a syntactic dependency analysis, and creates appropriate structures. The dependencies are triples in the form: *name_of_the_relation(governor, dependent)* and are constructed as follows:

1. **subject**. There are two types of subjects in relations: the nominal subject *nsubj* and the passive nominal subject *nsubjpass*. A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb, which can be an adjective or a noun. A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.

2. **Object**. The direct object of a verb phrase is the noun phrase which is the (accusative) object of the verb.

3. **Verb**. Verbs are wrapped with either the subject or the object.

4. **Noun**. A noun compound modifier of an noun phrase is any noun that serves to modify the head noun.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

5. **Preposition**. A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, noun, or even another preposition.

6. **Determiner**. A determiner is the relation between the head of an noun phrase and its determiner.

7. **Number**. An element of compound number is a part of a number phrase or currency amount.

The matching module is given the dependency relations, and it compares the hypothesis with the text through the following features. A perfect score of 1 is given when all the arguments in the dependency relations match in both text and hypothesis. The value is 0.5 for partial matches.

1. **Subject-Verb Comparison.** The goal is to compare the subject and verb of the hypothesis with the subject and verb of the text, which are identified through the *nsubj* and *nsubjpass* dependency relations extracted from the previous module. In case of a total match the matching score is 1, otherwise the system considers the next matching process.

2. **WordNet Based Subject-Verb Comparison.** This feature is used in the case that the subject matches in the hypothesis and text, but the verb does not. A comparison is made with WordNet between the verbs. If the output value is less than 0.5, this indicates a relatedness of the corresponding verbs, so a match is considered and a matching score of 0.5 is given.

3. **Subject-Subject Comparison.** A simple comparison between the subject of the hypothesis and the subject of the text is made. If a match is found, then a score of 0.5 is assigned.

4. **Object-Verb Comparison.** The goal is to compare the object and verb of the hypothesis with the object and verb of the text, which are identified through the *dobj* dependency relation extracted from the previous module. In case of a match the matching score is set to 0.5.

5. **WordNet Based Object-Verb Comparison.** This feature is used in cases where the objects match in the hypothesis and text. If a match exists, a comparison is made between the verb associated with the hypothesis object, and the verb associated with the text. If the two verbs do not match, a WordNet distance, which measures the semantic relation between WordNet synsets, is applied (Liu *et al.*, 2011). If the output value is less than 0.5, the matching score is set to 0.5.

6. **Cross Subject-Object Comparison.** The goal is to compare the subject and verb of the hypothesis, with the subject and verb of the text. In case of a match the matching score is

set to 0.5.

7. **Number Comparison.** The goal is to compare numbers and units in the hypothesis, with the numbers and units in the text. In case of a match the matching score is set to 1.

8. **Noun Comparison.** The goal is to compare the noun words of the hypothesis with noun words of the text, which are identified through the *nn* dependency relation. In case of a match the matching score is set to 1.

9. **Prepositional Phrase Comparison.** The goal is to compare the preposition of the hypothesis with the preposition of the text, which are identified through the *prep* dependency relation. Then, the authors compare the noun words that are arguments in the preposition relation, as described earlier. In case of a match, the matching score is set to 1.

10. **Determiner Comparison.** The goal is to compare the determiners of the hypothesis with the determiners in the text, which are identified through the *det* dependency relation. In case of a match, the matching score is set to 1.

The datasets used for experiments and results were RTE-1,2 and 3, all of them separated into development and test sets, the RTE-4 test set and the RTE-5 main development and test sets, to deal with two-way classification. The obtained results outperformed the other systems that participated in the RTE challenge using lexical and syntactic approaches, in all of the RTE challenges. Regarding accuracy the authors achieved results of 0.537 for RTE-1, 0.592 for RTE-2, 0.61 for RTE-3, 0.554 for RTE-4 and 0.603 for the RTE-5 dataset.

2.2.2.2 Lexical and Sentence Structure Entailment Detection

Tsuchida & Ishikawa (2011) proposed a RTE system that uses machine learning methods with features based on lexical and predicate-argument structure level information. The underlying idea is to identify the text-hypothesis pairs that have a high entailment score but are in fact not entailed, i.e., false-positive pairs classified by the system's lexical-level module can latter be rejected by the sentence-level module.

The system first computes the entailment score of the text T and hypothesis H pairs and detects possible entailment by comparing the scores with a threshold obtained from the training data. For this task the authors assumed that T entails H when T has sufficient words common to, or in an entailment relation with, the words in H. The entailment score is defined as follows:

$$\mathrm{ent\_sc}(T,H) = \frac{\sum_{t_h \in H_t} \mathrm{match}(t_h, T_t, R)\mathrm{w}(t_h)}{\sum_{t_h \in H_t} \mathrm{w}(t_h)} \tag{2.16}$$

$$\mathrm{w}(t) = \log\left(\frac{N}{\mathrm{freq}(t)}\right)^{\alpha} \tag{2.17}$$

In the formulas, each T_t and H_t denotes a set of words in each given text T and hypothesis H while $\mathrm{w}(t)$ is the weight of the word t, and $\mathrm{freq}(t)$ is the frequency of the word t in the corpus. N is the number of the texts in the corpus and R consists of WordNet. Finally, $\mathrm{match}(t, T_t, R)$ is assigned to 1 if the word t corresponds to a word in T_t (synonyms and derived words in R are also considered), otherwise it is assigned to 0. The system normalizes all organizations names into their acronyms and then calculates the score. If it is greater than the threshold, the pair is considered as entailed.

Then, the system uses a filtering mechanism that rejects the pairs that are possible entailments but appear to be, in fact, not entailed. For this task, various features are employed. At a lexical level: (1) entailment score $\mathrm{ent\_sc}$, (2) cosine similarity, and (3) entailment score $\mathrm{ent\_sc}$, where a comparison is made between only the words using same parts-of-speech tags. At chunk level, matching ratios for each chunk types, such as noun and verb phrases, in all corresponding chunk pairs. At the predicate-argument structure (PAS) level: (1) matching ratios from each argument type (e.g., A0, A1) (Carreras & Màrquez, 2005).

Numbered arguments define verb-specific roles. Their semantics depends on the verb and the verb usage in a sentence. A0 stands for the agent and A1 corresponds to the *patient* or *theme* of the preposition, in all corresponding PASs pairs for each semantic relation of two predicates; (2) the number of negation mismatches in all corresponding PAS pairs for each semantic relation of two predicates; and (3) the number of modal verb mismatch in all corresponding PASs for each semantic relation of two predicates. For acquiring the features at the PAS-level, the authors need to first detect corresponding pairs that are possible entailments. For doing this, they use a simple alignment method, which first transforms all words contained in the PAS into a vector, using a bag-of-words representation, and then calculates the cosine similarity for all pairs that are generated by combining the PAS from each T and H. The authors consider *corresponding pairs* the most similar PAS from T for each PAS from H.

For detecting if the *corresponding pairs* are in fact entailed, the authors calculate the entailment score between the words of each argument type of the PAS from H, and the words of the same argument type of the PAS from T. After this, the argument type in the pair is considered a match if the score is greater then an empirically defined threshold $T_{arg} = 0.70$. The authors count the number of matching PASs for each argument type in all the pairs, distinguished by each relation type of two predicates, and calculate matching ratios for the predicates. In the filtering mechanism, the authors first detect a threshold T_{prec} with a pre-defined precision obtained in the development set. The filtering mechanism then rejects the pairs that have a greater value

predicted by the model then the threshold $T_{prec} = 0.80$. The results show that the filtering model improves precision in some cases, but the authors claim that the major increase in performance comes from the lexical entailment, because there are no major improvements by the filtering in terms of the F-measure, by the lexical entailment score.

2.2.3 Contradiction Detection

2.2.3.1 Semantic Alignment for Detecting Contradictions

Semantic alignment was proposed by Pham *et al.* (2013) as an approach for detecting contradictions. The idea relies on the alignment of semantic role frames (SRL) extracted from the text and the hypothesis in each pair. Denote an SLR frame by a tuple $S = \{V, E_1, ..., E_k\}$ where V is used to denote the verb predicate, and where E_i represents the i-th SRL element in the frame. Each SRL element has a type and is associated to the underlying words. Types of SRL elements follow the annotation guidelines in Propbank, a well-known annotated corpus with verbal compositions and their arguments (Palmer *et al.*, 2005). SLR elements can be modifiers or arguments. The authors denote two sets of SRL frames T and H by $T = \left\{S_i^{(t)}\right\}_{i=1}^{m}$ and $H = \left\{S_j^{(h)}\right\}_{j=1}^{n}$, in which m and n are the number of SRL frames extracted from text T and hypothesis H, respectively. The contradiction model is based on a contradiction function $\mathrm{F}(T, H)$, where T is an input text and H a hypothesis, that calculates the contradiction measurement of the pair (T, H). The result from $\mathrm{F}(T, H)$ is compared with a threshold value t_1 and if $\mathrm{F}(T, H) \geq t_1$ then (T, H) is contradictory.

The function $\mathrm{F}(T, H)$ returns a contradiction decision if there exists an event indicated by an SLR frame in H, which is incompatible with an event indicated by T. Formally $\mathrm{F}(T, H)$ is defined as:

$$\mathrm{F}(T, H) = \max_{S_i^{(t)} \in T, S_j^{(h)} \in H} \mathrm{f}(S_i^{(t)}, S_j^{(h)}) \tag{2.18}$$

In the previous equation, $S_i^{(t)}$ and $S_j^{(h)}$ are two SRL frames in T and H, respectively, and $\mathrm{f}(S_i^{(t)}, S_j^{(h)})$ is a contraction function defined on the two SRL frames. The authors define a function $\mathrm{f}(S_1^{(t)}, S_2^{(h)})$ of two SRL frames $S_1^{(t)} \in T$ and $S_2^{(h)} \in H$ with basis the alignment of SRL elements across the two frames. Since the number of SRL elements in an SRL frame is not very large, the authors propose a greedy alignment algorithm that considers all possible pairs of SRLs elements in $S_1^{(t)}$ and $S_2^{(h)}$. The core part of the algorithm is the similarity measure between two SRL elements. This measure is a local lexical level matching method (Ido Dagan & Massimo, 2007). After generating the alignment between the SRL elements, a more precise definition of the contradiction function $\mathrm{f}(S_1^{(t)}, S_2^{(h)})$ is needed. The notion of contradiction in this model is based on relatedness. The authors consider that two events are not contradictory if they are not

RTE-3			RTE-4			RTE-5		
Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
13.41	15.27	14.28	22.41	17.33	19.55	22.72	16.67	19.23

Table 2.6: Summarized results of the SLR-method

related. The relatedness of two SRL frames is defined as a product of the relatedness of their verb predicates and SRL elements:

$$\mathrm{R}(S_1^{(t)}, S_2^{(h)}) = \mathrm{R}(V_1, V_2) \times \max_{i,j} \mathrm{R}(E_i^{(1)}, E_j^{(2)}) \tag{2.19}$$

In the previous formula $R(E_i^{(1)}, E_j^{(2)})$ represents the relatedness between two items, where $E_i^{(1)} \in S_1^{(t)}$ and $E_j^{(2)} \in S_2^{(h)}$ are SRL elements, and the parameters V_1 and V_2 refer to the verbs of $S_1^{(t)}$ and $S_2^{(h)}$, respectively.

The measure of how two verbs are related is obtained using external tools like WordNet or VerbOcean (Chklovski & Pantel, 2004). The relatedness of two SRL elements $E_i^{(1)}$ and $E_j^{(2)}$ is defined as a local lexical level matching score. The relatedness of two SRL frames is compared with a threshold. If it is below the threshold, then $S_1^{(t)}$ and $S_2^{(h)}$ are not related. In the case of two frames that are related, the authors consider two situations: (1) two verbs predicates are matching, and (2) two verbs predicates are opposite. In the first situation, we have a case where WordNet identified the verbs as synonyms. In this situation, the function $\mathrm{f}(S_1^{(t)}, S_2^{(h)})$ is defined based on the alignment generated in the alignment process. Incompatibility of aligned arguments and modifiers (e.g., temporal, location, or negation modifiers) are used to calculate $\mathrm{f}(S_1^{(t)}, S_2^{(h)})$. In the second case, two verbs are opposite if they are found as antonym verbs in WordNet or opposite verbs in VerbOcean. In this case, the contradiction function $\mathrm{f}(S_1^{(t)}, S_2^{(h)})$ is defined as the similarity of their SRL elements. The element-based similarity of two frames is defined as the product of similarity scores of the aligned elements having the same type. The authors evaluated this method (SRL-method) against RTE datasets number $3, 4$ and 5. As a baseline the authors used previously proposed approaches (Clark & Harrison, 2009; Marneffe *et al.*, 2008). The results of this approach are shown in Table 2.6. The comparison with the baselines is show in Table 2.8.

2.2.3.2 Binary Relation Matching for Contradiction Detection

Pham *et al.* (2013) also proposed another approach, in this case using binary relation matching. The main idea is to extract triples from T and H. A triple can be seen as tuple that contains two words and a relation verb between them. Triples are extracted using ReVerb (Fader *et al.*, 2011), a tool which can automatically identify and extract binary relations from English sentences. The

input of ReVerb is a POS-tagged and NP-chunked sentence, and its output is a set of extracted tuples of the form $(arg1, R, arg2)$, in which R represents the relation phrase between the two arguments $arg1$ and $arg2$. ReVerb cannot extract some useful relation types, such as *isA* relations which specify the equivalence of two objects. We also have that, sometimes, the relation phrases of two extraction triples cannot be compared without the use of inference rules that specify the entailment relationship between the two triples.

With the output provided by ReVerb, a comparison is made between one triple in H against every triple in T, and this determines whether a contradiction relationship exists in some pairs of triples. Pham *et al.* (2013) denote an extraction triple by (x, r, y), where x and y represent the first and second arguments, and where r represents the relation phrase of the triple. Formally, $T = (x_i^{(t)}, r_i^{(t)}, y_i^{(t)})_{i=1}^{m}$ and $H = (x_j^{(h)}, r_j^{(h)}, y_j^{(h)})_{j=1}^{n}$ where m and n are respectively the numbers of triples in T and H. The contradiction detection task is reduced to searching for incompatible triple pairs across T and H. The definition of the contradiction function is thus as follows:

$$\mathrm{F}(T, H) = \max_{T_i \in T; H_j \in H} \mathrm{g}(T_i, H_j) \tag{2.20}$$

In the previous formula T_i is the i-th triple of T, H_j is the j-th triple of H and $\mathrm{g}(T_i, H_j)$ is the contradiction function of the two triples T_i and H_j. The function $\mathrm{g}(T_i, H_j)$ is based on the mismatch between two triples T_i and H_j. Three cases are taken into consideration: (1) if the relation phrases and first arguments are matching, the mismatch of second arguments will be calculated; (2) if two relation phrases are matching and the roles of arguments in the two triples are exchanged, $\mathrm{g}(T_i, H_j)$ returns the value of 1. This rule is not applied to *isA* relations; (3) if two relation phrases are opposite, some similarity measures between the first arguments and second arguments are taken into account. To determine if two relation phrases are matching or not, WordNet is used to detect whether the main verbs of $r_i^{(t)}$ and $r_j^{(t)}$ are synonyms or antonyms. VerbOcean is used to determine if the verbs have an opposite relation. Regarding the arguments, their level of matching is calculated by using their similarity. The similarity score is computed using the method explained in the previous section, based on the product of similarity scores of the aligned elements having the same type. The contradiction rule is the following: two arguments are contradictory if they include two entities having the same type, but different values. The authors evaluated this method (Triple-method) against the datasets and baseline methods described in the previous subsection. The results are show in Table 2.7.

The authors evaluated the two approaches (i.e., the SLR-based method described in the previous section, and the method based on binary relations) separately and combined. The final system that is proposed (the combination of the two approaches) consistently obtained better recall values, as well as better F1 scores, than the baseline methods from Marneffe *et al.* (2008)

and from Clark & Harrison (2009) for the RTE-2 and RTE-3 test set. As an individual system, the SRL method achieves better recall and F1 scores than the binary relation method. The authors claim that a possible factor is that information contained in shallow semantic representations is richer than that of the extraction triples in the binary relation approach. Table 2.8 summarizes the experimental results and comparison of the different approaches.

2.2.3.3 Functional Relations for Contradiction Detection

Ritter *et al.* (2008) proposed a novel approach where we can map phrases as functional relations in order to do a comparison and infer it they contradict each other. A relation R is functional in a variable x (the subject of the sentence) if it maps it to a unique variable y (the object): $\forall_{x,y_1,y_2}\mathrm{R}(x,y_1) \wedge \mathrm{R}(x,y_2) \Rightarrow y_1 = y_2$. Although this definition seems simple, extracting functional relations in text is more complex due to ambiguity, synonyms and other linguistic phenomena. To build a contradiction test set, the authors extracted a set of phrases with the same relation R and argument x, but with different y values: $\mathrm{R}(x,-)$. For this, instead of analyzing sentences directly, the proposed system relies on the TextRunner Open Information Extraction system (Banko *et al.*, 2007). Although I will only focus on the contradiction detection system, it is important to mention that the authors also introduce and evaluated a new expectation-maximization algorithm for detecting whether phrases denote functional relations, which enables a contradiction detection system to identify functions in arbitrary domains.

The system extracts a set of assertions ε, each in the form $\mathrm{R}(x,y)$ and with an assigned probability p_e of being correct. From ε, a function learner module identifies a set of functional relations ζ and assigns to each relation in ζ a probability p_f of being functional. The contradiction detection module receives pairs of functional relations and builds a set C by filtering seeming contradictions based on synonyms, meronymy, argument types and argument ambiguity, and assigns a probability p_c that the functional relation is a genuine contradiction. In the filter phase, to deal with synonyms, the system relies on WordNet and on Resolver (Yates & Etzioni, 2007), i.e., a system that identifies synonyms from TextRunner extractions. The system also uses the edit-distance algorithm to check the similarity between the objects y, to identify synonyms. For meronyms the authors rely on the Tipster Gazetteer[1] and on WordNet. The authors claim that both of these

[1] http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat

RTE-3			RTE-4			RTE-5		
Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
22.58	9.72	13.59	26.3	10	14.49	19.48	16.67	17.96

Table 2.7: Results of the Triple-method based approach

tools have high precision but low coverage. Regarding argument types, the system relies on the feature that if two values of y have different types, the values are not contradictory. Finally, regarding ambiguity the system computes the probability that a value x is unambiguous as a part of its function learner. A value x can be identified as ambiguous if its distribution of y values is non-functional for multiple functional relations. If a pair of extractions, $\{\mathrm{R}(x, y_1), \mathrm{R}(x, y_2)\}$ does not match any of the features presented, and if R is functional, there is a strong possibility that the sentences underlying the extractions are genuine contradictions of each other. The authors combined the knowledge sources described above using Logistic Regression and a 10-fold cross validation methodology to automatically tune the weights associated with each knowledge source.

The system was evaluated against TextRunner's extractions from a corpus of 117 million webpages. The authors restrained their dataset to the most common relation. The authors labelled each relation as functional or not, and computed an estimate of the probability of being functional. They hand-tagged 8.844 pairs as contradictions or not, and found only 110 genuine contradictions. The authors also evaluated the system against actual web data, and compared it to the handmade dataset. Figure 2.1 shows the results obtained, which suggest that it is much easier to detect contradictions on manually selected datasets with a balanced distributions of negative and positive examples rather a large set gathered from the Web.

2.3 Summary and Discussion

The previous state-of-the-art works in contradiction detection have significant disadvantages. A major flaw is that all the systems are language dependent as they use features based on vocabulary tools that are language driven such as WordNet and VerbOcean. In like manner, the usage of syntactic parsing tools such as Propbank somewhat restricts the system to a language because the syntax of an English sentence is different from a syntax of a Portuguese sentence.

Regarding the paraphrase identification task, we saw that the previous work presented takes

Method	RTE-3			RTE-4			RTE-5		
	P	R	F1	P	R	F1	P	R	F1
De Marneffe	22.95	19.44	21.04	-	-	-	-	-	-
Bleu system	-	-	-	41.67	10	16.13	42.86	6.67	11.54
SRL	13.41	15.27	14.28	22.41	17.33	19.55	22.72	16.67	19.23
Triple	22.58	9.72	13.59	26.3	10	14.49	19.48	16.67	17.96
Combined	14	19.44	16.27	23	22.67	22.82	21.14	28.89	24.4

Table 2.8: Comparison between the authors approaches and the baselines

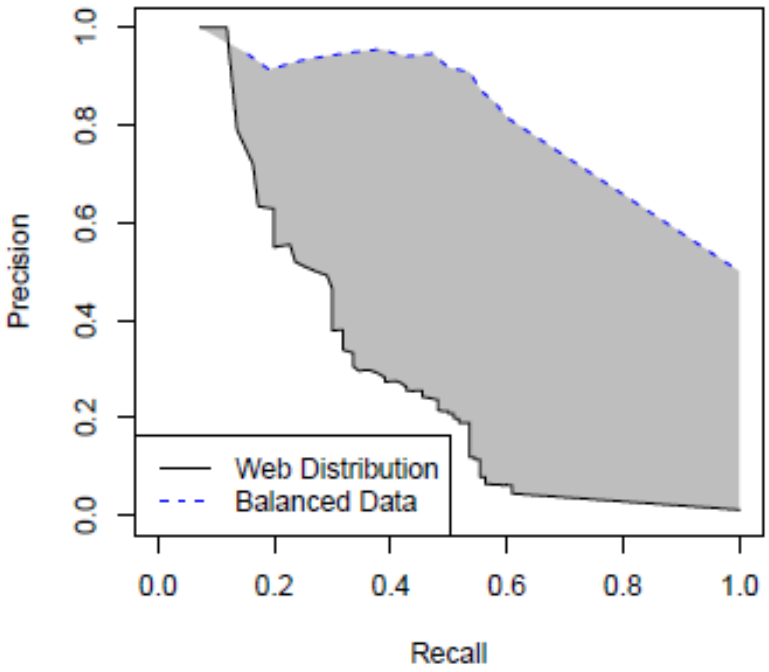


Figure 2.1: Comparison of the results between hand-made and web data.

advantages of methods that are language independent using simple strategies such as counting n-grams. In the same manner, the authors of the recognizing textual entailment task concluded that the lexical modules of their system achieves better results than others like syntactic and sentence structure modules.

In addition, there are some aspects that should be taken into account from Ritter *et al.* (2008). The first one is that genuine contradictions are in fact hard to find, therefore a need to relax the problem of contradiction detection as considered in Section 1.1 is crucial. Moreover, is more reliable to detect contradictions with a balanced dataset, i.e., an equal distribution of positive and negative examples.

Because of the disadvantages reported above, there was a need to shy away from the previous methods and consider a simple novel approach for contradiction detection: treat the problem as a classification task relying on simple lexical and numeric features in addition with paraphrase identification and recognizing textual entailment features.

Chapter 3

Proposed Solution

This chapter presents the solution proposed for the system. It starts by describing the architecture of the system. I present and discuss each component in detail and summarize with an overview of the system.

3.1 Architecture and overview

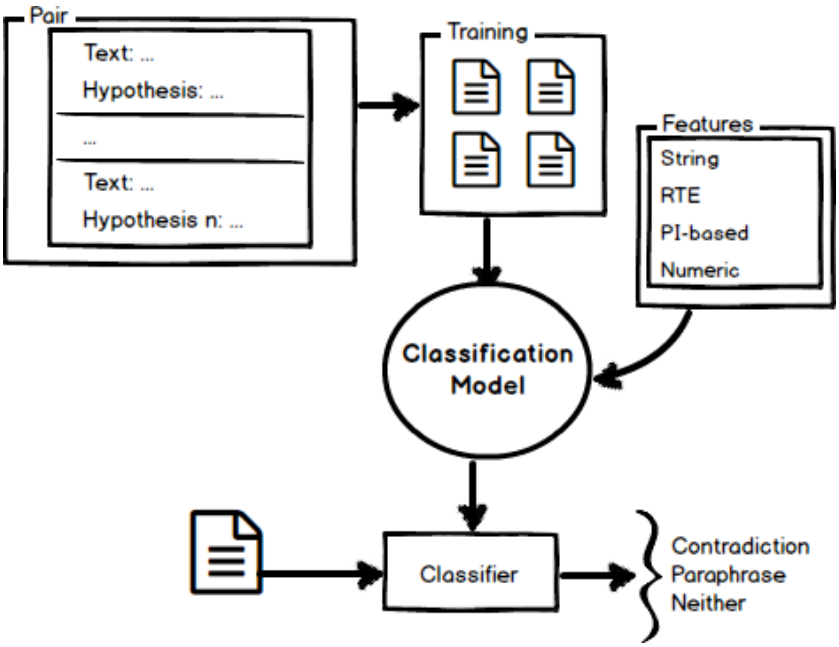


Figure 3.2: Proposed architecture of the system.

The architecture of the system can be observed in Figure 3.2. To better interpret the general

overview of the system, the methodology used is exhibited next:

1. Construct the training set, adding annotated pairs as contradictions, paraphrases or neither.
2. With the training data build the classification model.
3. The classification model is trained with a set of features presented in Section 3.2.
4. Different classifiers were used in this work. The classifiers are presented and explained in Section 3.3.

Consequently, a new pair is given and classified as a contradiction, paraphrase or neither.

3.2 Features

The model has the following features divided into 4 groups: (1) string-based, (2) RTE-based, (3) paraphrase identification-based, and (4) numeric. Besides the features, I used different types of similarity metrics involving different representations for the lexical contents of the sentences.

3.2.1 String-based

The string-based features are:

1. **Longest Common Subsequence**. The size of the Longest Common Subsequence (LCS) between the text and the hypothesis. The value is clamped between 0 and 1 by dividing the size of the LCS by the sentence with the longer length.
2. **Edit Distance**. The size of the minimum edit distance between the text and the hypothesis.
3. **Length**. The absolute length between the text and the hypothesis. Also the maximum and minimum length are considered as separately features.
4. **Cosine Similarity**. The Cosine similarity between the text and hypothesis. This does not include weighting of the words by TF-IDF. Instead of using a TF-IDF strategy, a simpler approach for turning the text into a vector is given: a simple count of the words in the text i.e., the vector contains a counter for each word. The cosine formula used is described in Equation 2.5. The return value is a real number between 0 and 1. The higher the value, the more identical is the text-hypothesis pair.

5. **Jaccard Similarity**. The Jaccard similarity between the text and the hypothesis. The return value is a real number between 0 and 1, where 1 means equal, and 0 totally different. Jaccard similarity coefficient is used for comparing the similarity and diversity of sample sets. It measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. The Jaccard similarity between two sets of words A and B is defined as follows:

$$\mathrm{J(A,B)} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A} \cup \mathrm{B}|} \tag{3.21}$$

6. **Soft TF-IDF**. The Soft-TF-IDF similarity metric measures similarity between vector-based representations of the sentences, but considering an internal similarity metric for finding equivalent words. The Jaro-Winkler similarity metric between words with a threshold of 0.9, as the internal similarity metric. The Jaro distance d_j of two given strings s_1 and s_2 is:

$$d_j = \begin{cases} 0 & \text{if m = 0} \\ \frac{1}{3} \times (\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}) & \text{otherwise} \end{cases} \tag{3.22}$$

where m is the number of matching characters and t is half the number of transpositions. The Jaro-Winkler measure modifies the Jaro measure by adding more weight to a common prefix. Introduces 2 parameters: (1) *PL*, expressed as the length of the longest common prefix between the two strings, and (2) *PW*, the weight to give the prefix.

$$\mathrm{JaroWinkler}(x,y) = (1 - PL \times PW) \times \mathrm{jaro}(x,y) + PL \times PW \tag{3.23}$$

3.2.2 RTE-based

The recognizing textual entailment-based features are:

1. **NE Overlap**. The Jaccard similarity taken into consideration only Named entities words. For simplicity named entities are all words containing capital letters.
2. **NEG Overlap**. The Jaccard similarity taken into consideration only negative words. The negative words are: *não*, *nunca*, *jamais*, *nada*, *nenhum*, *ninguém*, *not*, *no*, *never*, *nothing* and *none*.
3. **MODAL Overlap**. The Jaccard similarity taken into consideration only modal words. The modal words are: *podia*, *poderia*, *dever*, *deve*, *devia*, *deverá*, *deveria*, *faria*, *possivel*, *possibilidade*, *possa*, *can*, *could*, *may*, *might*, *will*, *would*, *must*, *shall*, *should* and *possible*.

3.2.3 Paraphrase-based

The paraphrase identification (PI) based features are:

1. **BLEU**. It is computed as the amount of n-gram overlap, for different values of n between two sentences, tempered by a length penalty (Papineni *et al.*, 2002). This paper took $3, 4$ and 5 as values for n.

2. **METEOR**. The metric is a variation of BLEU based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision (Banerjee & Lavie, 2005).

3. **TER**. Evaluate the Translation Error Rate using a simple python library called pyter[1]. TER is extension of Word Error Rate (WER), WER is a simple metric based on dynamic programming that is defined as the number of edits needed to transform one string into another. TER includes a heuristic algorithm to deal with shifts in addition to insertions, deletions and substitutions (Snover *et al.*, 2006).

4. **NCD**. Evaluate the Normalized Compression Distance comparison between two strings, adapting the implementation yield by a python library called *web2py_utils* [2]. NCD is a way of measuring the similarity between two objects (Li *et al.*, 2004). The underlying idea is that if you compress two strings s_1 and s_2 only the overlapping information is extracted.

5. **ROUGE-N**. N-gram overlap based on co-occurrence statistics (Lin & Hovy, 2003).

6. **ROUGE-L**. Longest Common Subsequence based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically (Lin & Och, 2004).

7. **ROUGE-S**. Skip-bigram based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order (Lin & Och, 2004).

3.2.4 Numeric-based

The idea behind the numeric (num) feature is simple: sentences that refer to the same entities but with different numbers are contradictory, everything else is not. The result value is the multiplication of 2 Jaccard similarities. One between the numeric characters in the pair text-hypothesis, and a second between the surrounding words of such numeric characters. The result value is a float between 0 and 1, being 1 a contradictory statement.

[1] https://pypi.python.org/pypi/pyter
[2] https://pythonhosted.org/web2py_utils

3.2.5 Representation

Besides the features, different representation of text are considered. Text representation is a cognitive entity, a *mental* construct that plays a crucial role in text understanding, text understanding is the result of decoding the information. I specifically considered the following representations for the quotations:

1. **Original tokens**. The quotations extracted from the database (see Figure 3.2).

2. **Lowercased tokens**. Using NLTK[1] (a python NLP Toolboox) lowercase all tokens in the dataset.

3. **Stems lowercased tokens**. Using a Portuguese stemmer provided by NLTK, stem and lowercase all tokens in the dataset.

4. **Word clusters**. Using the Brown algorithm, an agglomerative bottom-up method that aggregates words in binary tree of classes (Turian *et al.*, 2010) through a criterion based on the log-probability of a text under a class-based language model. Much like a Hidden Markov Model, given a cluster membership indicators t_i for the tokens w_i in a text, the probability of the w_i given w_{i-1} is given by equation 2.13. The Brown clustering was applied to a collection of newswire documents from a Portuguese newspaper called *Público*[2].

5. **Double Metaphone**. A second generation of the Metaphone algorithm. An algorithm to code mostly English words (and foreign words often heard in the United States) phonetically by reducing them to a combination of 12 consonant sounds. It returns two codes if a word has two plausible pronunciations, such as a foreign word. This reduces matching problems from wrong spelling (Philips, 1990).

6. **Character trigrams**. Trigrams are a special case of the n-gram, where n is 3. The character trigrams are used as key terms in a representation of the phrase much as words are used as key terms to represent a document.

The model combines features along with these different representations giving a total of 106 features. The combinations can be seen in Table-3.9.

[1] http://www.nltk.org/
[2] http://www.publico.pt/

Feature	Original	Lowercased	Stemmed	Cluster	D. Metaphone	Trigrams
LCS	X	X	X	X	X	
Edit Distance	X	X	X	X	X	
Cosine Similarity	X	X	X	X	X	X
Abs Length	X	X	X	X	X	
Max Length	X	X	X	X	X	
Min Length	X	X	X	X	X	
Jaccard	X	X	X	X	X	
Soft TF-IDF	X	X	X			
NE Overlap	X	X	X	X	X	X
NEG Overlap	X	X	X	X	X	X
Modal Overlap	X	X	X	X	X	X
BLEU-(3,4,5)	X	X	X	X	X	
METEOR	X	X	X	X	X	
ROUGE N	X	X	X	X	X	
ROUGE L	X	X	X	X	X	
ROUGE S	X	X	X	X	X	
TER	X	X	X	X	X	
NCD	X	X	X	X	X	
Numeric	X	X	X			

Table 3.9: Combination of features with representations.

3.3 Classifiers

Taking into account what we learned in Section 2.1.2 and using an open source machine learning toolbox for Python *scikit-learn*[1]. The classifiers used are the following:

1. **Random Forest**. A simple way to understand this classifier is to assume that a random forest is the juxtaposition of many classification trees. For this reason, a random forest is denominated an ensemble method (Breiman, 2001). An explanation of a classification tree was given earlier, nevertheless and to sum up: To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree ”votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest). In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features.

   The main advantages of using classification trees are: (1) simple to understand and to interpret as trees can be visualised, (2) the cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree, and (3) uses a white box

[1] http://scikit-learn.org/stable/

model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.

2. **Support Vector Machine**. Support vector machines (SVMs) are a set of supervised learning methods used for classification and regression. SVMs take some data to start with that's already classified (the training set), and tries to predict a set of unclassified data (the testing set). The data often has a lot of different features, and so we can end up plotting each data item as a point in space, with the value of each feature being the value at a particular coordinate (Hearst *et al.*, 1998). For example, if we only had two features in each item of the data, the graph could look like Figure 3.3.

   Now (for two data features) what we want to do is find some line that splits the data between the two differently classified groups of data as well as we can.

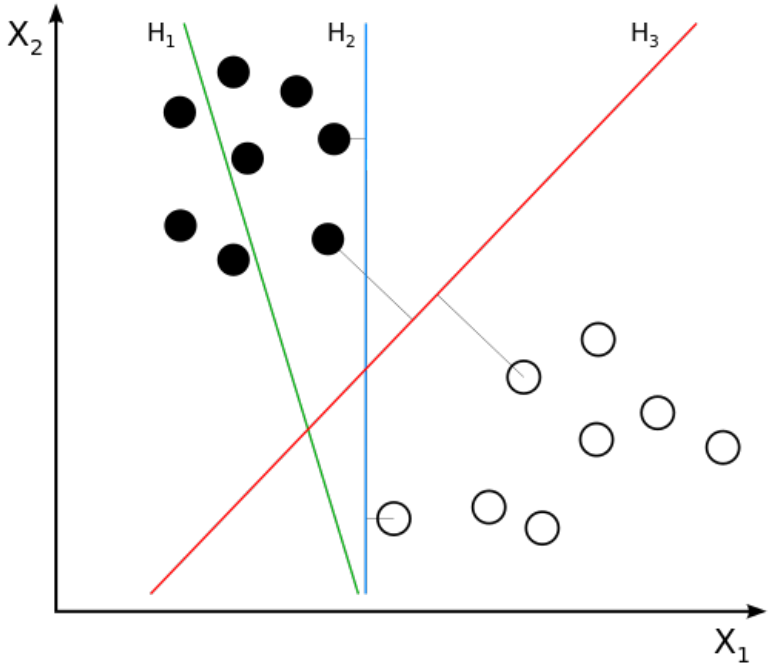


Figure 3.3: Example of a classification using SVM with 2 features.

   This will be the line such that the distances from the closest point in each of the two groups will be farthest away. In the example shown above, that would be the red line, since the two closest points there are the furthest apart from the line. Once we get the line, that's our classifier. Then depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

   We can extend the two feature example to many more features, but the idea still remains

the same. Find the line/plane/hyperplane that separates the two groups of data as much as possible, and then see which side the new data points land on.

3. **Stochastic Gradient Descend**. Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning (Bottou, 2010).

   Many numerical learning algorithms amount to optimizing a cost function that can be expressed as an average over the training examples. The loss function measures how well (or how poorly) the learning system performs on each example. The cost function is then the average of the loss function measures on all training examples, possibly augmented with capacity control terms. Computing such an average takes a time proportional to the number of examples n. This constant always appears in the running time of all optimization algorithm that considers the complete cost function. Stochastic gradient descent instead updates the learning system on the basis of the loss function measured for a single example. Such an algorithm works because the averaged effect of these updates is the same. Although the convergence is much more noisy, the elimination of the constant n in the computing cost can be a huge advantage for large-scale problems

4. **Logistic Regression**. Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier (Freedman, 2009). Logistic Regression is a type of regression that predicts the probability of occurrence of an event by fitting data to a logistic function. The logistic function 3.24 is an S-shaped function whose range lies between 0 and 1 (see Figure 3.4), which makes it useful to model probabilities. When used in classification, an output close to 1 can indicate that an item belongs to a certain class. Like many forms of regression analysis, it makes use of several predictor variables that may be either numerical or categorical.

$$\frac{1}{1+\mathrm{e}^{-x}} \tag{3.24}$$

5. **K-Nearest Neighbors**. Supervised neighbors-based learning has two trends: (1) classification for data with discrete labels, and (2) regression for data with continuous labels. In the context of this work, only KNN based on classification is used. The principle behind

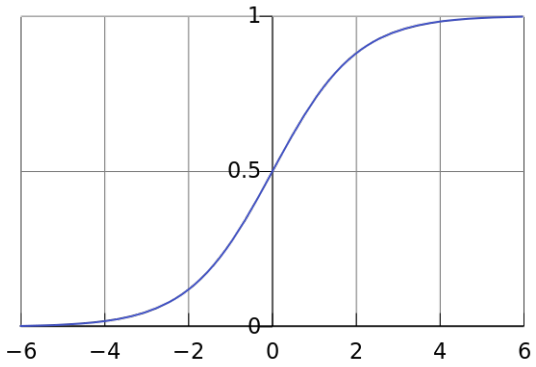


Figure 3.4: Logistic Function.

nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these (Cover & Hart, 1967). The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice, as was used in this work. Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

3.4 Summary

In this chapter I presented my approach towards contradiction and paraphrase detection. I started by describing the features used in the system, separated in 4 groups: (1) string-based, (2) RTE-based, (3) paraphrase identification-based, and (4) numeric. Besides the features, 6 different representations of text were construed, and presented a table that illustrates the combination of features/representation used. Furthermore, a brief description of the machine learning algorithms employed was provided alongside with the strategy for training and testing the system.

Chapter 4

Validation

Chapter 4 describes the methodology used for the evaluation of the system as well as the results, discussion and conclusions attained after the tests.

4.1 Evaluation Methodology

4.1.1 Datasets

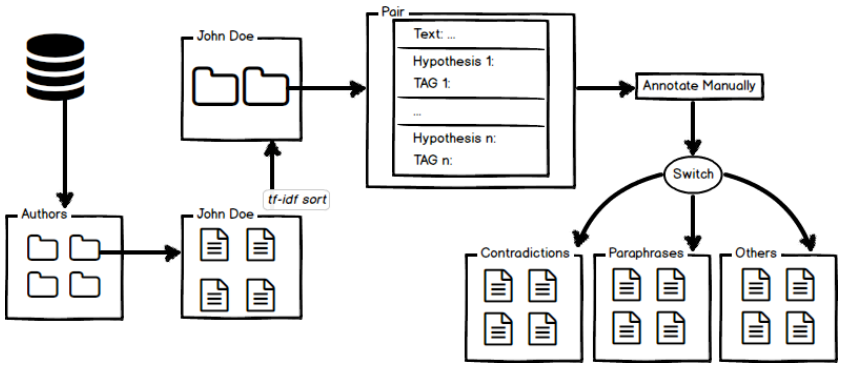


Figure 4.5: Proposed architecture for the *sapo.pt* dataset.

The system was tested against 2 datasets: (1) Microsoft Research Paraphrase Corpus (MSRPC) and, (2) the *sapo.pt* dataset created in the context of my M.Sc. dissertation containing news quotations from a database provided by *sapo.pt*.

The MSRPC corpus consists of a large set of sentence pairs $\{S_1, S_2\}$, together with labels indicating whether the sentences are in a paraphrase relationship or not. The MSRPC dataset

Text	Hypothesis	Paraphrase
They had published an advertisement on the Internet on June 10, offering the cargo for sale, he added.	On June 10, the ship's owners had published an advertisement on the Internet, offering the explosives for sale.	YES
Security lights have also been installed and police have swept the grounds for booby traps.	Security lights have also been installed on a barn near the front gate.	NO

Table 4.10: Example of a paraphrase and non-paraphrase from the MSRPC.

contains 5.801 sentence pairs, and the standard split of 4.076 training pairs (67.5% of which are paraphrases) and 1.725 test pairs (66.5% of which are paraphrases) was used. This corpus was created by mining news articles on the web for topically similar articles and then extracting potential sentential paraphrases using a set of heuristics. Example sentences are given in Table 4.10

The *sapo.pt* dataset has 19.631 quotation authors with a total of 193.528 quotations. There are 12.474 authors with only one quotation, which is obviously not enough for my research purposes. In consequence, a portion of the dataset containing only authors that have more than 2 quotations were taken into consideration. This dataset contains 7.147 authors and 181.055 quotations. With this subset, we have a mean of $25,3$ quotes per author with a standard deviation of $105,2$. The quotations have a mean of $25,6$ words, with a standard deviation of $13,4$. The dataset is made available as a relational database table that contains the following attributes:

1. **ID.** An identifier of the quotation;
2. **The actor.** The person who made the quote;
3. **The speech act.** The speech act used by the actor (e.g., Informed, Said);
4. **The quote.** The content of the quote;
5. **News issue date.** The day, month and year of the news issue;
6. **News issue time.** The hours, minutes and seconds of the news issue;
7. **News sources.** The journal(s) author(s) of the news;
8. **Quantity of news sources.** The number of news sources with such quotation;
9. **Normalized quotation.** The quotation lowercased.

The architecture for the construction of the *sapo.pt* dataset can be observed in Figure 4.5. To better interpret the general overview of the system, the methodology used is exhibited next:

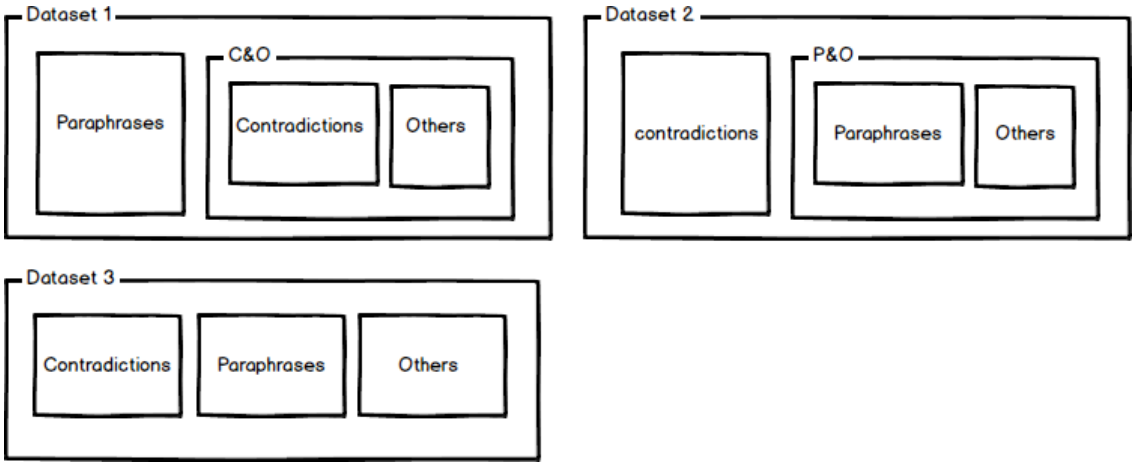


Figure 4.6: Sub-datasets used.

1. Retrieve all the quotations from a news database and group them by author.
2. For each author, compute the TF-IDF scoring model to all the quotations.
3. The output is, for each quotation, a text file containing the pair text-hypothesis. The structure of the text file is simple: for a quotation t (text) is given the top n highest score quotations h (hypothesis). An example of the text file can be seen in Figure 4.5.
4. Annotate manually the text file from the previous step filling in the *TAG* field.
5. Taking into consideration the *TAG* field, a simple switch expression is applied to forward the pair text-hypothesis to the correct annotation group. It is to notice that now the pair text-hypothesis is an unary relationship, i.e., for a quotation t there is only one quotation h.
6. Given the annotated pairs, a balanced distribution is performed and forwarded to the classifier.

In order to show what features are more prone to detect contradictions or to detect paraphrases, the dataset was divided into 3 sub-datasets as follows (see Figure 4.6):

1. **Dataset 1**.Paraphrase versus contradictions and others. This subset is divided as follows: 400 of pairs of sentences, of where 200 are pairs of paraphrases and 200 are not paraphrases. Of the latter, 100 are pairs of contradictions and 100 pairs of other quotations.
2. **Dataset 2**.Contradictions versus paraphrase and others. This subset is divided as follows: 400 of pairs of sentences, of where 200 are pairs of contradictions and 200 are not contradictions. Of the latter, 100 are pairs of paraphrases and 100 pairs of other quotations.

Text	Hypothesis	Type
Talvez tenha sido uma má opção realizarco próximo campeonatocdo mundo de futebol no Brasil, em 2014.	Considera injustas as críticas que têm sido feitas ao Brasil e aposta que o país vai organizar um Mundial extraordinário em 2014.	Contradiction
	A escolha do Brasil para país organizador do Mundial 2014 pode ter sido errada.	Paraphrase
	Tudo estará pronto para o Campeonato do Mundo de 2014.	Other

Table 4.11: Examples of a paraphrase, contradiction and other sentence pairs from the *sapo.pt* dataset.

3. **Dataset 3**.Paraphrase versus contradictions versus others. This subset is divided into 3 categories as follows: 200 pairs of paraphrases, 200 pairs of contradictions and 200 pairs of other quotations.

An example of a paraphrase, a contradiction or neither is given in Table 4.11. The English version is given in Table A.18.

4.1.2 Validation Plan

The evaluation of my approach is based on well defined metrics from the area of NLP that have been used extensively in related works, namely Precision, Recall, F1 and Accuracy. Precision is defined as the fraction of retrieved instances (e.g., detected contradictions) that are true:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.25}$$

In the formula TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives, respectively. Recall is defined as the fraction of relevant instances (e.g., of true contradictions existing in the data) that are retrieved:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.26}$$

Accuracy gives the overall correctness of the system, i.e., the answers that the system got right divided by the total of the answers that the system gave:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.27}$$

Finally, the F-measure is an approach to combine the previous measures of precision and recall into one, and it is defined as follows (β is usually set to 1):

$$\mathrm{F}_\beta = \frac{(\beta^2 + 1) \times \mathrm{Precision} \times \mathrm{Recall}}{\beta^2 \times \mathrm{Precision} + \mathrm{Recall}} \tag{4.28}$$

4.2 Results

4.2.1 Contradiction Detection

Classifier	Acc	F1	Precision	Recall	Combination of features
Stochastic Descendant	0.53 (0.13)	0.55 (0.23)	0.52 (0.23)	0.56 (0.18)	rte + pi + num
KNN	0.62 (0.21)	0.58 (0.29)	0.66 (0.26)	0.56 (0.28)	rte + pi
Logistic Regression	0.63 (0.17)	0.63 (0.18)	0.64 (0.18)	0.63 (0.17)	string + rte + pi + num
Linear SVM	0.66 (0.19)	0.66 (0.19)	0.67 (0.21)	0.66 (0.19)	string + rte + pi + num
Random Forest	0.71 (0.19)	0.70 (0.23)	0.70 (0.22)	0.72 (0.32)	string + rte + pi + num

Table 4.12: Best results in contradiction detection in the *sapo.pt* dataset

.

Concerning contradiction detection, the system was tested in the *sapo.pt* dataset with several classifiers, as explained in Section 3.1. Besides the use of different classifiers, all possible combinations of the 4 groups of features were taken into account. Table 4.12 shows the best result for each classifier accompanied by the best combination (among the possible 15) of the group features for the task of contradiction detection.

Classifier	Acc	F1	Precision	Recall	Combination of features
Stochastic Descendant	0.60 (0.21)	0.58 (0.24)	0.62 (0.24)	0.68 (0.23)	rte + pi + num
KNN	0.66 (0.26)	0.67 (0.23)	0.67 (0.30)	0.68 (0.27)	rte + pi + num
Logistic Regression	0.68 (0.26)	0.68 (0.26)	0.68 (0.26)	0.68 (0.26)	rte + pi + num
Linear SVM	0.69 (0.24)	0.68 (0.26)	0.71 (0.25)	0.69 (0.24)	rte + pi + num
Random Forest	0.73 (0.21)	0.73 (0.24)	0.74 (0.26)	0.71 (0.29)	string + pi +num

Table 4.13: Best results in paraphrase identification in the *sapo.pt* dataset.

Classifier	Acc	F1	Precision	Recall	Combination of features
KNN	72%	81%	88%	75%	rte + pi
Stochastic Descendant	74%	82%	93%	74%	pi + num
Linear SVM	75%	82%	88%	78%	string + rte + pi +num
Logistic Regression	75%	83%	88%	77%	rte + pi
Random Forest	76%	83%	87%	79%	string + rte + pi +num

Table 4.14: Best results in paraphrase identification in the MSRPC dataset.

4.2.2 Paraphrase Identification

Regarding paraphrase identification, the system was tested against the MSRPC and the *sapo.pt* datasets with the same classifiers used in contradiction detection. Like before, all combinations of the 4 groups of features were included. Table 4.13 and Table 4.14 shows the best result for each classifier followed by the associated group of features for the *sapo.pt* and the MSRPC dataset respectively.

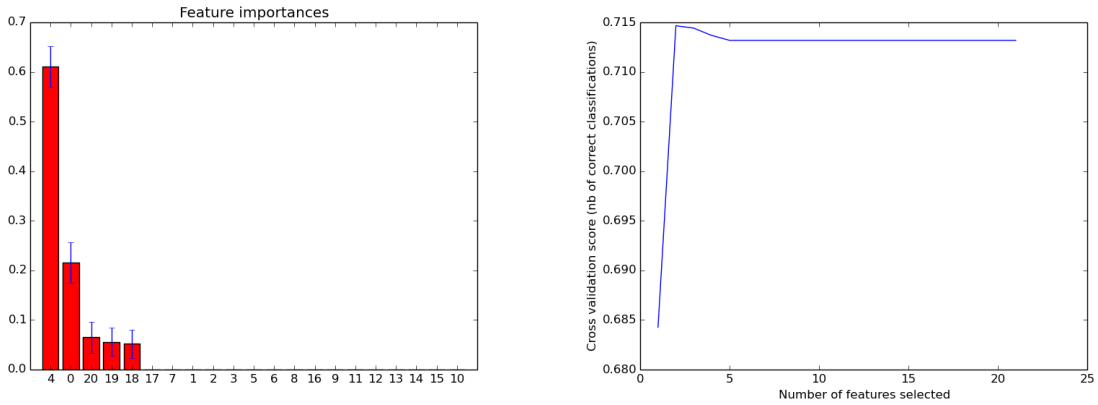
An additional study and comparison is made between the most informative features for paraphrase identification amongst the 2 datasets available, which is presented in Table A.19.

4.2.3 Paraphrase, Contradiction or neither

The task of classifying a pair of quotations as a contradiction, paraphrase or neither, again the system was tested with all combinations of the 4 groups of features and with different classifiers. Table 4.15 shows the best results of task of classifying a sentence pair as a contradiction, a paraphrase or neither in the *sapo.pt* dataset.

Classifier	Acc	F1	Precision	Recall	Combination of features
Stochastic Descendant	0.48 (0.23)	0.47 (0.20)	0.46 (0.19)	0.45 (0.19)	rte + pi + num
KNN	0.55 (0.16)	0.54 (0.16)	0.55 (0.16)	0.55 (0.16)	rte + pi
Logistic Regression	0.59 (0.17)	0.59 (0.19)	0.60 (0.20)	0.59 (0.17)	string + rte + pi +num
Linear SVM	0.60 (0.17)	0.60 (0.17)	0.60 (0.19)	0.60 (0.17)	string + rte + pi +num
Random Forest	0.66 (0.19)	0.65 (0.19)	0.68 (0.19)	0.65 (0.16)	string + rte + pi +num

Table 4.15: Best results in classifying a sentence pair as a contradiction, paraphrase or neither in the *sapo.pt* dataset.



(a) Feature weights.

(b) RFE output

Figure 4.7: RFE strategy for the paraphrase identification task in the MSRPC using 20 features.

4.2.4 Recursive Feature Elimination

To extract better results from the system, a recursive feature elimination (RFE) strategy was used. The goal of RFE is given an external estimator (e.g.,the coefficients of a linear model), select features by recursively considering smaller and smaller sets of features. The estimator is trained on the initial set of features and weights are assigned to each of them. Then, features whose absolute weights are the smallest are pruned from the current set. This procedures is greedy and repeated on the pruned set until the desired number of features to select is reached.

An example of how this algorithm works is shown in Figure 4.7. In figure-4.7(a) we have an example of the coefficients weights. In figure 4.7(b) we have the end result of the RFE strategy. To notice that the optimal number of features for this example is 2 instead of 20.

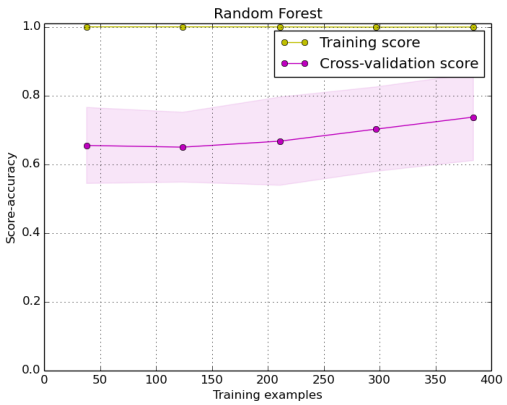
Table 4.16 present the best result for each task using a RFE strategy.

Task	Dataset	Acc	F1	N. Features
Contradiction Detection	*sapo.pt*	71%	71%	44
Paraphrase Identification	*sapo.pt*	74%	73%	12
Paraphrase Identification	MSRPC	76.1%	83.1%	72
CD vs PI vs Others	*sapo.pt*	67.1%	67.2%	81

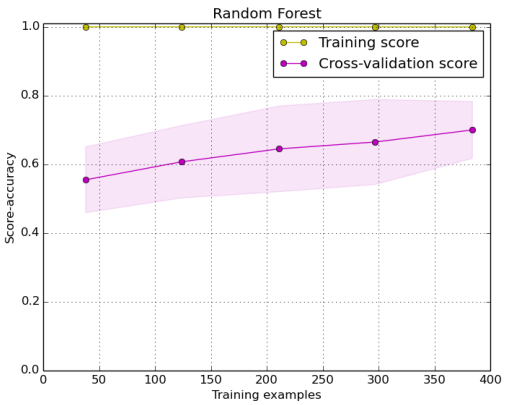
Table 4.16: Results using RFE strategy.

4.3 Cross validation and Learning Curves

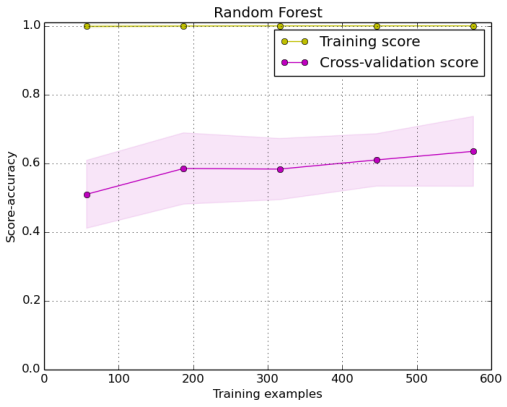
To prevent overfitting, i.e., a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data a cross validation (CV) procedure was used. Usually, to prevent overfitting a common practice when performing a (supervised) machine learning experiment is to hold out part of the available data as a test set, generally 40%. However, by partitioning the available data into two sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets. As we can see in Figure 4.6 the dataset used for the contradiction detection model is very limited, therefore a 25-fold CV was used. The following procedure is followed for each of the k folds: (1) A model is trained using $k-1$ of the folds as training data; (2) the resulting model is validated on the



(a) Learning Curve for the task of paraphrase identification.



(b) Learning Curve for the task of contradiction detection.



(c) Learning Curve for the task of detecting contradiction, paraphrase or neither.

Figure 4.8: Leaning curves for each task regarding Accuracy.

remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy and F1-score). The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as it is the case when fixing an arbitrary test set), which is a major advantage in problem such as inverse inference where the number of samples is very small.

From the results with the *sapo.pt* dataset, it is visible that the variance changes with the classifier. In the worst case, the variance is in the order of 30% considering accuracy. With the purpose of evaluate this variance the learning curve was plotted. A learning curve shows the validation and training score of an estimator for varying numbers of training samples. It is a tool to find out how much we benefit from adding more training data. If both the validation score and the training score converge to a value that is too low with increasing size of the training set, we will not benefit much from more training data. If the training score is much greater than the validation score for the maximum number of training samples, adding more training samples will most likely increase generalization.

Figure 4.8 shows the learning curves, confirming that the system can reduce the variance and gain better results by adding more training instances to the dataset.

4.4 Summary and Discussion

As outlined in Section 1.3, this work establishes more contributions: (1) a study of how significant a feature is, and (2) a comparison between features for detecting contradiction and paraphrases in order to evaluate if there is a correlation. From Table 4.16 we can contemplate the optimal number of features, i.e., the most informative features for each task and compare them (see Table 4.17).

Table A.19 infers several things. To begin with, a large dataset requires more features to identify paraphrases then a small dataset such as *sapo.pt*. Secondly, almost all of the features used to capture paraphrase in the *sapo.pt* are the same used in the MSRPC dataset. This is because the features for paraphrase identification are language independent. In addition, recognizing textual entailment features are not so important to the paraphrase identification task.

Table 4.17 claims several things. First, it takes more features to detect contradictions than paraphrase identification. Second, in fact there is a strong connection between paraphrase identification and contradiction detection. Third, this paper proofs in fact that some paraphrase identification based features are crucial for contradiction detection. In contrast, recognizing textual

entailment based features did not have the same impact as paraphrase identification based features as hypothesized in the beginning of this work.

Furthermore, from both Tables A and 4.17 it is clear that the numeric feature is very important for every task. I surmise that the importance of this feature is substantial because it takes into consideration the surrounding words in the sentence.

In conclusion, this paper achieves a classification of 71% for both F-score and accuracy regarding the contradiction detection task in the *sapo.pt* dataset. Concerning paraphrase identification this paper achieves a classification accuracy of 76.1% and F-score of 83.1% showing that out-of-the-box approaches on simple learning algorithms and somewhat effortless features can compete with previous state-of-the-art results as seen in Table 2.1.

	Contradictions		Paraphrases	
Feature/ Representation	String	lcs / stem_lowered	String	cosine / trigram
		lcs / metaphone		tfidf / original
		len_cluster		
		len / metaphone		
		jaccard / lower_case		
		jaccard / stem_lowered		
		jaccard / metaphone		
		jaccard / trigram		
		tfidf / original		
		tfidf / lower_case		
		tfidf / stem_lowered		
	RTE	overlap('ne') / original	RTE	racio_neg / lower_case
		overlap('ne') / metaphone		racio_modal / trigram
		racio_neg / original		
		racio_neg / lower_case		
		racio_neg / trigram		
		racio_modal / original		
		racio_modal / lower_case		
		racio_modal / trigram		
	PI	ter / original	PI	ncd / metaphone.
		ncd / original		rouge_n / stem_lowered
		ncd / lower_case		rouge_n / cluster
		ncd / cluster		rouge_n / metaphone
		ncd / metaphone		rouge_l / metaphone
		bleu-3 / original		
		bleu-3 / stem_lowered		
		bleu-4 / original		
		bleu-4 / stem_lowered		
		bleu-5 / original		
		bleu-5 / stem_lowered		
		rouge_n / original		
		rouge_n / lower_case		
		rouge_n / stem_lowered		
		rouge_n / metaphone		
		rouge_l / original		
		rouge_l / lower_case		
		rouge_l / metaphone		
		rouge_s / original		
		rouge_s / lower_case		
		rouge_s / stem_lowered		
		rouge_s / cluster		
	Numeric	num / original	Numeric	num / original
		num / lower_case		num / lower_case
		num / stem_lowered		num / stem_lowered

Table 4.17: The cells that are outlined are the features in common to both contradiction and paraphrase detection.

Chapter 5

Conclusions and Future Work

In this chapter I review all the work done and the results achieved, and discuss possible directives for future work.

5.1 Conclusions

With the fast growth of information content in a large news corpus, it is important to have a system capable of detecting misleading sentences. Contradiction detection has multiple applications, for instance, in the context of major candidate debates and political campaigns. Contradiction can arise in many ways, some rather easy to detect while others require more deeper world knowledge.

To overcome these challenges, this work addresses the problem of detecting contradiction as a classification task, where given pair of sentences s_1 and s_2 from the same author, detects if the pair is contradictory, a paraphrase or neither by applying several features based from recognizing textual entailment and paraphrase identification - mostly machine translation and summarization metrics. These features together with string and numeric features represents a novel approach that shies away from the most recent line of work of the area, which mostly focused on building systems based on semantic alignment and binary relations matching.

Additionally, ascertain if the more informative features for paraphrase identification are the more informative features for contradiction detection in the *sapo.pt* dataset. Furthermore, see if the more informative features for paraphrase identification in the *sapo.pt* dataset are the more informative features for the MSRPC dataset.

The published system achieved very satisfactory results with both an accuracy and F-score of

71% for the task of detecting contradictions and an accuracy of 76.1% and F-score of 83.1% for the task of paraphrase identification.

5.2 Future Work

The system developed presents a good performance given the limitations imposed by the lack of the training data. However, there is still room from improvement. Additionally, this work raises several interesting research problems that can be addressed in future work.

First of all, as observed in Section 4.3 the system can improve the results for the task of detecting contradictions when added more training data examples. Second of all, as a possible future work, the model can be enhanced by adding the notion of when the quotation was made.

Finally, a research problem that has room to grow is the study of different representations, for instance, experimenting with the usage of rules to transform sentences using different grammatical patterns (i.e., transforming passive voice into active voice).

Bibliography

ALMEIDA, M.S., ALMEIDA, M.B. & MARTINS, A.F. (2014). A joint model for quotation attribution and coreference resolution. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

BANERJEE, S. & LAVIE, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

BANKO, M., CAFARELLA, M.J., SODERLAND, S., BROADHEAD, M. & ETZIONI, O. (2007). Open information extraction for the web. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

BLACOE, W. & LAPATA, M. (2012). A comparison of vector-based representations for semantic composition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

BOTTOU, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of Computer Statistics*, Springer.

BREIMAN, L. (2001). Random forests. *Machine learning*, **45**.

BRILL, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Conference on Applied Natural Language Processing*.

CARRERAS, X. & MÀRQUEZ, L. (2005). Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Conference on Computational Natural Language Learning*.

CHAN, Y.S. & NG, H.T. (2008). Maxsim: A maximum similarity metric for machine translation evaluation. In *Proceedings of the Conference on Association for Computational Linguistics*.

CHKLOVSKI, T. & PANTEL, P. (2004). Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

CLARK, P. & HARRISON, P. (2009). Recognizing textaul entailment with logical inference. In *Proceedings of the Textual Analysis Conference*.

COVER, T.M. & HART, P.E. (1967). Nearest neighbor pattern classification. *Information Theory*, **13**.

DAGAN, I., DOLAN, B., MAGNINI, B. & ROTH, D. (2009). Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, **15**.

DAS, D. & SMITH, N.A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the Annual Meeting of the ACL and the International Joint Conference on Natural Language Processing of the AFNLP*.

DODDINGTON, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, Morgan Kaufmann Publishers Inc.

DOLAN, B., QUIRK, C. & BROCKETT, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the International Conference on Computational Linguistics*.

FADER, A., SODERLAND, S. & ETZIONI, O. (2011). Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

FERNANDO, S. & STEVENSON, M. (2008). A semantic similarity approach to paraphrase detection. In *Proceedings of the Annual Research Colloquium on Computational Linguistics in the UK*.

FINCH, A., HWANG, Y.S. & SUMITA, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the International Workshop on Paraphrasing*.

FREEDMAN, D.A. (2009). *Statistical models: theory and practice*. Cambridge University Press.

GÓMEZ-RODRÍGUEZ, C., CARROLL, J.A. & WEIR, D.J. (2008). A deductive approach to dependency parsing. In *Proceedings of the Annual Meeting on the Association for Computational Linguistics*.

HABASH, N. & ELKHOLY, A. (2008). Sepia: surface span extension to syntactic dependency precision-based mt evaluation. In *Proceedings of the Workshop on Metrics for Machine Translation at Antenna Measurement Techniques Association*.

HEARST, M.A., DUMAIS, S.T., OSMAN, E., PLATT, J. & SCHOLKOPF, B. (1998). Support vector machines. *Intelligent Systems and their Applications*, **13**.

IDO DAGAN, D.R. & MASSIMO, F. (2007). A tutorial on textual entailment.

ISLAM, A. & INKPEN, D. (2006). Second order co-occurrence pmi for determining the semantic similarity of words. In *Proceedings of the International Conference on Language Resources and Evaluation*.

ISLAM, M.A., INKPEN, D. & KIRINGA, I. (2007). A generalized approach to word segmentation using maximum length descending frequency and entropy rate. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*.

JURAFSKY, D. & MARTIN, J.H. (2009). *Speech and Language Processing*. Prentice-Hall, Inc.

KOTSIANTIS, S.B. (2013). Decision trees: A recent overview. *Artificial Intelligence Review*, **39**.

KOZAREVA, Z. & MONTOYO, A. (2006). Paraphrase identification on the basis of supervised machine learning techniques. In *Proceedings of the International Conference on Advances in Natural Language Processing*.

LAFFERTY, J.D., MCCALLUM, A. & PEREIRA, F.C.N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

LI, M., CHEN, X., LI, X., MA, B. & VITÁNYI, P. (2004). The similarity metric. *Information Theory, IEEE Transactions on*, **50**.

LIN, C.Y. & HOVY, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

LIN, C.Y. & OCH, F.J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

LIU, G., WANG, R., BUCKLEY, J. & ZHOU, H.M. (2011). A wordnet-based semantic similarity measure enhanced by internet-based knowledge. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*.

MADNANI, N., TETREAULT, J. & CHODOROW, M. (2012). Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

MANNING, C.D., RAGHAVAN, P. & SCHÜTZE, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

MARNEFFE, M.C., N. RAFFERTY, A. & D. MANNING, C. (2008). Finding contradictions in text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

MICHIE, D., SPIEGELHALTER, D.J., TAYLOR, C.C. & CAMPBELL, J. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

MIHALCEA, R., CORLEY, C. & STRAPPARAVA, C. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the National Conference on Artificial Intelligence*.

NADEAU, D. & SEKINE, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, **30**.

PAKRAY, P., BANDYOPADHYAY, S. & GELBUKH, A. (2011). Textual entailment using lexical and syntatic similarity. *Internacional Journal of Artificial Intelligence and Applications*, **2**.

PALMER, M., GILDEA, D. & KINGSBURY, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, **31**.

PAPINENI, K., ROUKOS, S., WARD, T. & ZHU, W.J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

PARKER, S. (2008). Badger: A new machine translation metric. *Metrics for Machine Translation Challenge*.

PHAM, M.Q.N., NGUYEN, M.L. & SHIMAZU, A. (2013). Using shallow semantic parsing and relation extraction for finding contradiction in text. In *Proceedings of the International Joint Conference on Natural Language Processing*.

PHILIPS, L. (1990). Hanging on the metaphone. *Computer Language Magazine*, **7**.

QIU, L., KAN, M.Y. & CHUA, T.S. (2006). Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

QUINTÃO, M. (2014). Msc thesis on quotation attribution for portuguese news corpora. *Department of Electrical and Computer Engineering, Técnico Lisboa*.

RITTER, A., DOWNEY, D., SODERLAND, S. & ETZIONI, O. (2008). It's a contradiction—no, it's not: A case study using functional relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

RUS, V., MCCARTHY, P.M., LINTEAN, M.C., MCNAMARA, D.S. & GRAESSER, A.C. (2008). Paraphrase identification with lexico-syntactic graph subsumption. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*.

SNOVER, M., DORR, B., SCHWARTZ, R., MICCIULLA, L. & MAKHOUL, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*.

SNOVER, M.G., MADNANI, N., DORR, B. & SCHWARTZ, R. (2009). Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, **23**.

SOCHER, R., HUANG, E.H., PENNIN, J., MANNING, C.D. & NG, A.Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the Conference on Neural Information Processing Systems*.

TSUCHIDA, M. & ISHIKAWA, K. (2011). A method for recognizing textual entailment using lexical-level and sentence structure-level features. *In Proceedings of the Text Analysis Conference*.

TURIAN, J., RATINOV, L. & BENGIO, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

UL-QAYYUM, Z. & WASIF, A. (2012). Paraphrase identification using semantic heuristics features. In *Research Journal of Applied Sciences, Engineering and Tecnhology*, vol. 4.

WAN, S., DRAS, M., DALE, R. & PARIS, C. (2006). Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*.

YATES, A. & ETZIONI, O. (2007). Unsupervised resolution of objects and relations on the web. In *Proceedings of the Human Language Technology Conference*.

ZHANG, Y. & PATRICK, J. (2005). Paraphrase identification by text canonicalization. In *Proceedings of the Australasian Language Technology Workshop*.

Appendix A

Annexed Materials

Text	Hypothesis	Type
Maybe it was a bad idea to perform the next World Championship Soccer in Brazil, in 2014.	Considers unfair the criticisms that have been made to Brazil and bet that the country will organize an extraordinary World Cup in 2014.	Contradiction
	The choice of Brazil to manage the World Cup in 2004 may have been wrong.	Paraphrase
	Everything will be ready for the World Cup in 2014.	Other

Table A.18: Examples, in English, of a paraphrase, contradiction and other sentence pairs from the *sapo.pt* dataset.

Table A.19: The cells that are outlined are the features in common to both datasets.

	MSRPC		*sapo.pt*	
Feature / Representation	String	lcs / original	String	cosine / trigram
		lcs / stem_lowered		tfidf / original
		lcs / cluster		
		edit_distance / lower_case		
		edit_distance / stem_lowered.		
		edit_distance / cluster		
		edit_distance / metaphone		
		cosine / original		
		cosine / edit_distance		
		cosine / stem_lowered		
		cosine / cluster		
		cosine / metaphone		
		cosine / trigram		
		len / original		
		len / lower_case		
		len / stem_lowered.		
		len / cluster		
		len / metaphone		
		minimo / lower_case		
		maximo / lower_case		
		maximo / metaphone		
		jaccard / original		
		jaccard / lower_case		
		jaccard / lowered		
		jaccard / cluster		
		jaccard / metaphone		
		jaccard / trigram		
		tfidf / original		
		tfidf / lower_case		
		tfidf / stem_lowered		
	RTE	overlap('ne') / original	RTE	racio_neg / lower_case
		overlap('ne') / metaphone		racio_modal / trigram
		racio_neg / lower_case		
		racio_neg / stem_lowered		
		racio_neg / trigram		
		racio_modal / original		
		racio_modal / lower_case		
		racio_modal / stem_lowered		
	PI	ter / original	PI	ncd / metaphone.
		ter / lower_case		rouge_n / stem_lowered
		ter / stem_lowered		rouge_n / cluster
		ter / cluster		rouge_n / metaphone
		ter / metaphone		rouge_l / metaphone
		ncd / lower_case		
		ncd / stem_lowered		
		ncd / cluster		
		ncd / metaphone		

	bleu-3 / original		
	bleu-3 / lower_case		
	bleu-3 / cluster		
	bleu-4 / stem_lowered		
	bleu-4 / metaphone		
	bleu-5 / original		
	bleu-5 / lower_case		
	bleu-5 / stem_lowered		
	bleu-5 / cluster		
	bleu-5 / metaphone		
	rouge_n / original		
	rouge_n / lower_case		
	rouge_n / stem_lowered		
	rouge_n / cluster		
	rouge_n / metaphone		
	rouge_l / original		
	rouge_l / lower_case		
	rouge_l / cluster		
	rouge_s / original		
	rouge_s / lower_case		
	rouge_s / stem_lowered		
	rouge_s / cluster		
Numeric	num / original num / lower_case num / stem_lowered	Numeric	num / original num / lower_case num / stem_lowered