Detecting Contradictions in News Quotations

Ricardo Marques
ricardo.sa.marques@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

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. 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%. I propose to address this problem as a classification task using features based on those used for detecting paraphrases and recognizing textual entailment, alongside with numeric and string based features. Furthermore, this M.Sc. dissertation provides an assessment of what are the most informative features for detecting contradictions and infer if exists a correlation between detecting paraphrases and contradictions.

Detectar afirmações contraditórias é uma tarefa fundamental e desafiadora da linguagem natural e de aprendizagem automática, com inúmeras aplicações na extração e recuperação de informação. Por exemplo, contradições precisam de ser reconhecidas por sistemas de questão-resposta ou sistemas de sumarização de vários documentos. Relativamente a aprendizagem automática, requer a capacidade, através de aprendizagem supervisionada, para estimar com precisão e capturar as diferenças entre contradições e, por exemplo, paráfrases. Esta dissertação de Mestrado proporciona um sistema capaz de detectar contradições a partir de um par de afirmações publicado através de jornais com ambos resultados de F1 e precisão de 71%. Proponho-me a abordar este problema como um tarefa de classificação usando recursos baseados na detecção de paráfrases e reconhecimento de vinculação textual, juntamente com recursos numéricos e de cadeias de caracteres. Além disso, esta dissertação de Mestrado fornece uma avaliação de quais são as características mais informativas para detectar contradições e inferir se existe uma correlação entre a detecção de paráfrases e contradições.
Keywords: Contradiction, Paraphrases, Classification, Entailment

1. Introduction

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.

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 is 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. 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.

Since a paraphrase is a type of entailment, i.e., bidirectional entailment, 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.

2. Related Work

2.1. 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. Ana saw a bear.

2. Ana 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 *Ana* 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 (i.e., 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.

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 1: Comparison of different methods that have been previously preposed for detecting paraphrases.

These methods are based in word-to-word similarity measures (e.g., knowledge-based metrics which use WordNet[1]). Methods based in alignments (e.g., summarization and machine translation metrics are also commonly used, for instance, Madnani *et al.* (2012) proposed yet another approach based on evaluation methods from the area of machine translation, in light of recent developments. These authors showed that a meta-classifier (i.e., a classifier that uses the average of the unweighted probability estimates given by three constituent classifiers to make the final decision, in which the constituent classifiers are based on logistic regression, support vector machines, and nearest-neighbor searching) using nothing but modern MT metrics (i.e., 8 different MT metrics, including 6 that were not available in 2005) outperforms many recent paraphrase identification approaches relying on more linguistically-informed features used.

Finch *et al.* (2005) proposed a similar method to those of Zhang & Patrick (2005) or of Kozareva & Montoyo (2006), but instead relying on standard evaluation methods for Machine Translation (MT), which are based on n-gram overlaps or edit distances (e.g., BLEU, NIST, WER and PER) as the similarity metrics between the sentences, and using a Support Vector Machine (SVM) classifier with radial basis function kernels to classify the data. The authors used stemming to conflate morphologically related words (i.e., verbs and adjectives) to the same root, as a pre-processing step to canonicalize the sentences. The authors also introduced a method based on the PER MT evaluation metric, which leverages part-of-speech information of the words contributing to the word matches and non-matches in the sentence. Table 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).

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

2.2. Contradiction detection

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. 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 notion of contradiction in this model is based on relatedness. The authors consider that two events are not contradictory if they are not related. The relatedness of two SRL frames is defined as a product of the relatedness of their verb predicates and SRL elements. 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 $f(S_1^{(t)}, S_2^{(h)})$ 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 $f(S_1^{(t)}, S_2^{(h)})$. 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 $f(S_1^{(t)}, S_2^{(h)})$ 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.

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. The authors 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 function $g(T_i, H_j)$ task is reduced to searching for incompatible triple pairs across T and H. 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, $g(T_i, H_j)$ 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. The contradiction rule is the following: two arguments are contradictory if they include two entities having the same type, but different values.

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): $\forall_{x,y_1,y_2} \mathrm{R}(x, y_1) \wedge \mathrm{R}(x, y_2) \Rightarrow y_1 = y_2$. 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[2] and on WordNet. 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. Table 2 summarizes the experimental results and comparison of the different approaches, using as a baseline the work by Marneffe *et al.* (2008).

3. Proposed Method

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.

3.1. Features

3.1.1 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.

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

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: Comparison between the authors approaches and the baselines.

4. **Cosine Similarity**. The Cosine similarity between the text and hypothesis. This does not include weighting 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 1 where $\overrightarrow{V}(d_1)$ and $\overrightarrow{V}(d_2)$ denote vector representations. The return value is a real number between 0 and 1. The higher the value, the more identical is the text-hypothesis pair.

$$\text{sim}(d_1, d_2) = \frac{\overrightarrow{V}(d_1) \cdot \overrightarrow{V}(d_2)}{||\overrightarrow{V}(d_1)|| \times ||\overrightarrow{V}(d_2)||} \quad (1)$$

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:

$$\text{J(A, B)} = \frac{|\text{A} \cap \text{B}|}{|\text{A} \cup \text{B}|} \quad (2)$$

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:

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

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.

$$\begin{aligned} \text{JaroWinkler}(x, y) = (1 - PL \times PW) \times \\ \text{jaro}(x, y) + PL \times PW \end{aligned} \quad (4)$$

3.1.2 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 *no*, *nunca*, *jamais*, *nada*, *nenhum*, *ningum*, *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*, *dever*, *deveria*, *faria*, *possivel*, *possibilidade*, *possa*, *can*, *could*, *may*, *might*, *will*, *would*, *must*, *shall*, *should* and *possible*.

3.1.3 Paraphrase-based

The paraphrase identification (PI) based features are:

1. **BLEU**. It is a 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). In this paper we 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[3]. 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* [4]. 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).

3.1.4 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.

3.2. 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 *sapo.pt* database.

2. **Lowercased tokens**. Using NLTK[5] (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. 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 5. The Brown clustering was applied to a collection of newswire documents from a Portuguese newspaper called *Públlico*[6].

$$\begin{aligned} &\text{argmax}_{T\in\gamma}\text{P}(W|T)\times\text{P}(T)\approx \\ &\text{argmax}_{T\in\gamma}\prod_{i=1}^{n}\text{P}(w_i|t_i)\text{P}(t_i|t_{i-1}) \end{aligned} \tag{5}$$

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.

All representations are used for the following features: Cosine similarity, NE Overlap, NEG Overlap and Modal Overlap. Using only the original, lowercased and stemmed representations are the following features: Soft TF-IDF and numeric. The remaining features used all the representations except from the Character trigrams representation. With this combination the model has a total of 106 features.

3.3. Classifiers

Using an open source machine learning toolbox for Python *scikit-learn*[7], the classifiers used are the following:

[3] https://pypi.python.org/pypi/pyter
[4] https://pythonhosted.org/web2py_utils
[5] http://www.nltk.org/
[6] http://www.publico.pt/
[7] http://scikit-learn.org/stable/

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.

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). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall 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 is an S-shaped function whose range lies between 0 and 1, 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.

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 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 paper. 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.

4. Results

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[8] 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 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.

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 methodology for constructing the *sapo.pt* dataset is exhibited next:

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).
4. Annotate manually the text file from the previous step.
5. Taking into consideration the annotation from the previous step, 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.

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

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:

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. 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.
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.

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. Table 3 presents 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 3: Results using RFE strategy.

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 3 we can contemplate the optimal number of features, i.e., the most informative features for each task and compare them.

From comparing the most informative features for paraphrase identification against the most informative features for contradiction detection I infer several points. First, it takes more features to detect contradictions than paraphrase identification. Second, is that 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, 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.

5. Conclusions

This work addressed the problem of detecting contradiction given a pair of sentences s_1 and s_2 using features that are highly adopted from the task of recognizing textual entailment as well as paraphrase identification. These features together with string and numeric based 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. The solution was to create a classifier that, given pair of sentences from the same author, detects if the pair is contradictory, a paraphrase or neither. Additionally, ascertain if the more informative features for paraphrase identification are the more informative features for contradiction detection. Furthermore, see if the more informative features for the *sapo.pt* dataset are the more informative features for the MSRPC dataset regarding paraphrase identification.

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. Despite the interesting results, there are also several ideas for future improvements. First add more training data examples, secondly experiment with the usage of rules to transform sentences using different grammatical patterns (i.e., transforming passive voice into active voice) and thirdly, adding the notion of when the quotation was made.

References

Banerjee, S. & Lavie, A. (2005). In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization.*

Blacoe, W. & Lapata, M. (2012). In: *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning ,.*

Bottou, L. (2010). In: *Proceedings of Computer Statistics.* Springer.

Breiman, L. (2001). *Machine learning,* **45** (1).

Chklovski, T. & Pantel, P. (2004). In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Cover, T. M. & Hart, P. E. (1967). *Information Theory,* **13** (1).

Dagan, I., Dolan, B., Magnini, B., & Roth, D. (2009). *Natural Language Engineering,* **15** (04).

Das, D. & Smith, N. A. (2009). 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.*

Dolan, B., Quirk, C., & Brockett, C. (2004). In: *Proceedings of the International Conference on Computational Linguistics.*

Fernando, S. & Stevenson, M. (2008). In: *Proceedings of the Annual Research Colloquium on Computational Linguistics in the UK.*

Finch, A., Hwang, Y.-S., & Sumita, E. (2005). In: *Proceedings of the International Workshop on Paraphrasing.*

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

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., & Scholkopf, B. (1998). *Intelligent Systems and their Applications,* **13** (4).

Islam, A. & Inkpen, D. (2006). In: *Proceedings of the International Conference on Language Resources and Evaluation.*

Kozareva, Z. & Montoyo, A. (2006). In: *Proceedings of the International Conference on Advances in Natural Language Processing.*

Li, M., Chen, X., Li, X., Ma, B., & Vitányi, P. (2004). *Information Theory, IEEE Transactions on,* **50** (12).

Lin, C.-Y. & Hovy, E. (2003). 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). In: *Proceedings of the Annual Meeting on Association for Computational Linguistics.*

Madnani, N., Tetreault, J., & Chodorow, M. (2012). In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics.*

Marneffe, M.-C., N. Rafferty, A., & D. Manning, C. (2008). In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Mihalcea, R., Corley, C., & Strapparava, C. (2006). In: *Proceedings of the National Conference on Artificial Intelligence.*

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). In: *Proceedings of the Annual Meeting on Association for Computational Linguistics.*

Pham, M. Q. N., Nguyen, M. L., & Shimazu, A. (2013). In: *Proceedings of the International Joint Conference on Natural Language Processing.*

Philips, L. (1990). *Computer Language Magazine,* **7** (12).

Qiu, L., Kan, M.-Y., & Chua, T.-S. (2006). In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Ritter, A., Downey, D., Soderland, S., & Etzioni, O. (2008). 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). In: *Proceedings of the Florida Artificial Intelligence Research Society Conference.*

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). In: *Proceedings of association for machine translation in the Americas.*

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., & Ng, A. Y. (2011). In: *Proceedings of the Conference on Neural Information Processing Systems.*

Turian, J., Ratinov, L., & Bengio, Y. (2010). In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Ul-Qayyum, Z. & Wasif, A. (2012). In: *Research Journal of Applied Sciences, Engineering and Tecnhology.*

Wan, S., Dras, M., Dale, R., & Paris, C. (2006). In: *Proceedings of the Australasian Language Technology Workshop.*

Yates, A. & Etzioni, O. (2007). In: *Proceedings of the Human Language Technology Conference.*

Zhang, Y. & Patrick, J. (2005). In: *Proceedings of the Australasian Language Technology Workshop.*