Computational Detection of Irony in Textual Messages

Filipe Nuno Marques Pires da Silva

Thesis to obtain the Master of Science Degree in:

Information Systems and Computer Engineering

Supervisors: Doctor Bruno Emanuel da Graça Martins
Doctor David Manuel Martins de Matos

Examination Committee

Chairperson: Doctor Daniel Jorge Viegas Goncalves
Supervisor: Doctor Bruno Emanuel da Graça Martins
Member of the Committee: Doctor Mário Jorge Costa Gaspar da Silva

November 2016

Abstract

Analyzing user generated content in social media, using natural language processing methods, involves facing problematic rhetorical devices, such as sarcasm and irony. These devices can cause wrong responses from review summarization systems, sentiment classifiers, review ranking systems, or any kind of application dealing with the semantics and the pragmatics of text. Studies to improve the task of detecting textual irony have mostly focused mostly on simple linguistic cues, intrinsic to the text itself, although these have been insufficient in detecting ironic sentences that have no intrinsic clues. However, a new approach for classifying text, which can be made to explore external information (e.g. in the form of pre-trained word embeddings) has been emerging, making use of neural networks with multiple layers of data processing. This dissertation focuses on experimenting with different neural network architectures for addressing the problem of detecting irony in text, reporting on experiments with different datasets, from different domains. The results from the experiments show that neural networks are able to outperform standard machine learning classifiers on the task of irony detection, on different domains and particularly when using different combinations of word embeddings.

Keywords: irony detection , deep neural networks , social media , machine learning , natural language processing

Resumo

A utilização de técnicas de processamento de linguagem natural, na análise de contéudo gerado por utilizadores de redes sociais, envolve a detecção do uso de instrumentos retóricos como o sarcasmo e a ironia. Estes instrumentos podem causar uma resposta errada em sistemas de sumarização de resenhas, classificadores de sentimento, sistemas de classificação de resenhas ou qualquer outro tipo de aplicação que lide com a semântica e a pragmática associada a textos em língua natural. Existem estudos que tentaram abordar a tarefa de detectar ironia através de indícios simples intrínsecos ao texto. Contudo, estes métodos não têm sido suficientes para detectar algumas frases irónicas que não incluem esses indícios. Contudo, uma nova abordagem para a classificação de texto que permite explorar informação externa (e.g., expressas na forma de *word embeddings*), tem vindo a ganhar uma importância crescente na área, explorando eficazmente o uso de redes neuronais como forma de melhorar os resultados. Esta dissertação descreve experiências com diferentes arquitecturas de redes neuronais, reportando os resultados obtidos no problema de detecção de ironia em texto. Os resultados das experiências mostram que as redes neuronais, empregando diferentes combinações de *word embeddings*, conseguem superar, em diferentes domínios, os algoritmos clássicos de aprendizagem automática na tarefa de detecção de ironia.

Palavras Chave: detecção de ironia , redes neuronais , redes sociais , aprendizagem automática supervisionada

Acknowledgments

I would like to thank my advisors, Professor Bruno Martins and Professor David Matos, and also the research group from project EXPRESS for the opportunity to work with them and for the knowledge they shared with me.

I would also like to share the financial support from *Fundação para a Ciência e Tecnologia*, through the scholarship with reference UTAP-EXPL/EEi/ess/0031/201 in the aforementioned EXPRESS research project, whose main objectives overlapped of those from my dissertation.

Finally, I would like to extend my gratitude to those who, in a way or another, contributed anonymously with ideas and constructive criticism during the work of this dissertation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of Natural Language Processing (NLP) has been addressing the development of methods for analyzing social media data, for instance, to support review summarization systems, sentiment classifiers, and review ranking systems.

One of the problems that has been recurring in these NLP applications is the existence of sarcasm and irony (i.e., the use of words to convey a meaning that is the opposite of its literal meaning) in user comments. For example, ironic utterances might cause the NLP applications to misinterpret a given comment as a positive review when, in actuality, it should have been classified as a negative review, since the user comment was intended to be interpreted ironically. With this in mind, the work developed in the context of my MSc thesis relates to the computational detection of irony and sarcasm in text.

1.1 Motivation

Some previous studies have already addressed the automated detection of sarcasm and irony. However, most have focused on semantic or lexical cues directly available from the source texts, often within relatively simple models, without properly exploring external context. Examples of previous work include:

- studies concerned with the identification of ironic cues by leveraging patterns and lexicons for matching text with expressions usually associated with ironic speech, such as *yeah right* (Carvalho *et al.*, 2009; Tepperman *et al.*, 2006), as well as through the presence of profanity and slang (Burfoot & Baldwin, 2009);
- studies concerned with identifying words and phrases that contrast positive sentiments and negative situations, which usually denote irony (Riloff *et al.*, 2013);

- studies concerned with finding simple linguistic clues linked to specific pragmatic factors often associated with irony (González-Ibáñez *et al.*, 2011);

- studies that try to benefit from external context, for example, relevant newspapers and Wikipedia (Karoui *et al.*, 2015), or past user texts (Bamman & Smith, 2015).

An idea that may help improve the results for this problem is the use of an externally contextualized model similar to a human being (Wallace, 2015). People collect information about other people (e.g., by talking/reading what they said/wrote in the past) and about events (e.g., by reading/watching the news). A classification algorithm that also has that information about the utterer of the ironic speech, and about the subject of the utterance, will also likely be able to detect the nuances of an ironic expression. Several studies that use contextual information have already been developed and reported on several publications (e.g., Khattri *et al.* (2015), Bamman & Smith (2015)).

A new trend on the classification of textual documents consists of using classifiers based on deep neural networks (i.e., a feedforward neural network with *many* layers) and pre-trained word embeddings (e.g., Amir *et al.* (2016) or Ghosh & Veale (2016)). These algorithms, which contrast with machine learning techniques requiring a heavy use of feature engineering (i.e., the process of transforming data into features that help the performance of the classification approach being used), have been achieving similar or better results without the demanding task of implementing a large feature set. Instead, they rely on word embeddings (e.g., representations for words based on dense vectors, which are effective at capturing semantic similarity) to give semantic or/and contextual information about the documents being classified (Mikolov *et al.*, 2013a).

1.2 Thesis Statement

The intention behind my research was to verify if deep learning algorithms, that have been used on similar sentiment analysis tasks, can yield equivalent or improved results when compared to classic linear machine learning algorithms (e.g., Support Vector Machines (SVMs) or Logistic Regression models), on several tasks of irony detection with datasets from different domains.

1.3 Contributions

The work described in this dissertation consists of multiple experiments using deep learning algorithms, with different model architectures, which are then compared with machine learning algorithms based of feature engineering, similar to those that have been used on previous studies on the task of irony detection in textual messages.

The results from the experiments using deep neural networks show that it is possible to outperform feature-based classifiers using neural networks, although the differences in classification accuracy are relatively small. Moreover, the results show that combining multiple word embeddings, trained with different data and/or different procedures, helps to increase the performance of the neural network classification models, even if those embeddings were not created specifically for the domain of the target task or dataset.

The main contributions that can be derived from the results of the research performed in this dissertation include:

- new approaches for classifying texts based on the presence of irony, together with their experimental evaluation;
- a neural network architecture, combining convolutional and recurrent layers, that performs well on multiple datasets from different domains;
- a way to increase the performance of neural networks on the task of irony detection using multiple word embeddings in combination.

1.4 Organization of the Dissertation

The next chapter of this dissertation will start by introducing fundamental concepts regarding the automated classification of documents. Chapter 3 presents an overview of previous studies that have addressed the subject of irony detection. Chapter 4 describes the algorithms/architectures that have been used in my experiments, as well as the modeling of the features and word embeddings that are used with those algorithms. Chapter 5 details the datasets, the experimental methodology, and the results from the experiments. The dissertation the ends with a few observations that can be retrieved from the results, and with a discussion on the future work that can be undertaken from the results of this dissertation.

Chapter 2

Fundamental Concepts

To understand the research reported in this dissertation, a few fundamental concepts should first be introduced. Thus, in this chapter, I describe some of the concepts that are essential for the understanding of the rest of this document and the work reported in this dissertation.

2.1 Automated Document Classification

The task of document classification consists of assigning one or more classes (or labels), from a pre-defined set of classes, to a given textual document, as can be seen in Figure 2.1.

The model that is usually employed to represent textual documents, prior to their classification, is the Vector Space Model. This approach represents the documents as vectors of weighted identifiers (i.e., as feature vectors). These identifiers can, for instance refer to terms or n-grams of phrases, occurring in the document.

Each document *d* has its own vector where, for each term *t* or n-gram contained in a collection of documents, a weight *w* is given, usually according to the frequency with which the identifier appears on the document.

$$d = (w_{t_1}, w_{t_2}, w_{t_3}, ..., w_{t_n})$$

Leveraging these representations, document classification algorithms are usually based on statistical machine learning, i.e., they involve algorithms that deal with the problem of finding a predictive function based on a training set. These algorithms have two phases, namely a phase for learning the model parameters, using a training set of labeled documents, and a second phase in which the model is used to predict the class of unlabeled documents. Depending on the type of classifier, the model can be changed according to the availability of new training documents.

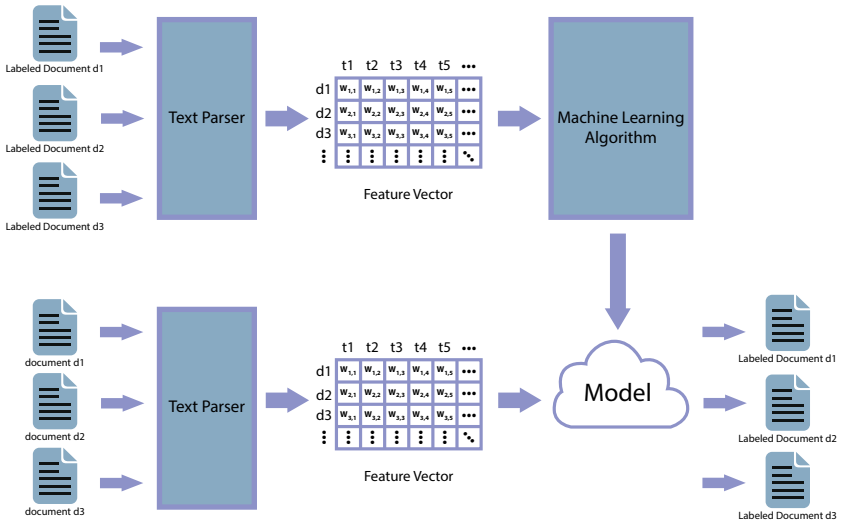


Figure 2.1: A diagram representing the two phases involved in the process of document classification, using a machine learning approach.

The labeled documents (i.e., examples of textual contents associated to the corresponding class label) are usually obtained from human evaluators with expertise on the domain of the classification problem, for instance through platforms such as Amazon Mechanical Turk (Ipeirotis, 2010) where annotators can request to participate in the annotation of a given dataset online. Another common annotation method when dealing with text from social media is to use specific tags (e.g. hash-tags of tweets) inside the messages, in order to automate the collection of labels (i.e., label all tweets with hash-tags like *#sarcasm* as being ironic). The hash-tags are associated to twitter posts by the authors of the messages, indicating/summarizing their actual contents. Although tags it might contain noisy information, leveraging this type of information is a practical approach for collecting very large datasets.

2.2 Evaluating the Classification of Documents

Common metrics to evaluate automated document classification tasks are precision, recall, the F1-score, and accuracy.

Accuracy (Acc) is a statistical measure of how well a classification method correctly makes decisions, which is simply calculated by dividing the number of correctly classified documents (Ccd) to the total number of documents (D):

$$Acc = \frac{Ccd}{D} \tag{2.1}$$

Precision (P) is a per-class metric (e.g., usually computed for the positive class, in the case of binary classification problems) that corresponds to the number of instances classified correctly

for a given label (true positive - TP) divided by the total number of instances classified with that label (true positive and false positive - TP and FP):

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

Recall (R) is also a per-class metric, that instead corresponds to the number of instances classified correctly for a given label (true positive - TP) divided by the total number of instances that actually have that label (true positive and false negative - TP and FN):

$$R = \frac{TP}{TP + FN} \tag{2.3}$$

The F1-score ($F1$) is a measure that corresponds to a harmonic mean of the precision and recall scores, promoting methods that perform equally well on both metrics:

$$F1 = 2 * \frac{P * R}{P + R} \tag{2.4}$$

It is important to emphasize that in binary classification problems, such as the one addressed in this work, it is common to compute precision, recall, and F1-score for the positive class label (e.g., for the ironic class), or instead report on averaged values from both class labels.

2.3 Sentiment Analysis and Opinion Mining

Sentiment analysis, also known as opinion mining, is the task of identifying and classifying subjective information in documents (Pang & Lee, 2008). This information refers to the sentiments and opinions that the utterer transmits, and the simplest and most common task in the area relates to classifying an utterance as either expressing a positive or a negative overall opinion (i.e., opinion polarity classification).

Common methods for opinion polarity classification involve statistical models and supervised learning, for instance relying on Support Vector Machines (SVMs) and carefully engineered features (e.g., presence of terms in lexicons of words which are known to be associated to a particular opinion class), or instead relying on convolutional or recurrent neural networks (Severyn & Moschitti (2015a), Severyn & Moschitti (2015b)).

Another approach for addressing the task of sentiment analysis is to use existing word lists which are scored for positivity/negativity (Thelwall *et al.*, 2012) or for emotion strength (e.g., using the dimensions of valence, arousal, and dominance (Warriner *et al.*, 2013)). These scores can be

directly used within heuristic procedures (Danforth & Dodds, 2010), or used as features in statistical methods.

2.4 Machine Learning Algorithms Used in Text Classification

The development of classification algorithms normally use a training set of instances whose class labels are already known. Taking as example my proposal, I will be training classifiers to label each document, in a set of observation instances, with the ironic and non-ironic labels, based on datasets of labeled documents.

2.4.1 Feature-based Machine Learning Algorithms

The most widely used classification algorithms in the detection of irony in text are the naïve Bayes (Rish, 2001) and the Support Vector Machine (SVM) (Cortes & Vapnik, 1995) classifiers.

These classification algorithms rely heavily on feature engineering to represent the instances, i.e., they rely on a carefully designed set of independent variables used as a measurable property to predict categories of an individual instance of the observation set. All the features on a classifier are included on what is denoted as a feature vector. For instance, the most common linguistic features used on automated classifiers for the detection of irony include:

Lexical features, which are based on the use of words or expressions to classify an instance. These features can use the lexicon directly, or they can be based on heuristics for capturing semantic cues, i.e., the meaning a single word or expression conveys. For example, *love* conveys a positive sentiment towards an entity. Obscene language is also an important semantic cue, since it might indicate that the utterance, due to its lack of formality on a formal domain, is not to be taken literally. Some pragmatic cues are also included in the subset of lexical-based features. Heavy punctuation (e.g., *!!!* or *?!*) and emoticons (e.g., *:)* or *XD*) are common on the detection of irony, since they offer a certain, although limited, sentiment context to the text being interpreted.

Pattern features, which are based on linguistic cues on a set of two or more words or phrases. These features usually rely on parts-of-speech (POS) tags, using patterns to classify instances, whether they are grammatical patterns or patterns with words that belong to a certain category. An example of a pattern feature is a bigram with two verbs, or a bigram containing a verb followed by a word belonging to a certain category, for example, possible classes of animals (e.g., mammals, birds, insects).

In what follows, to explain the most common classification algorithms, I will use the following

formal notation:

- χ denotes the input/observation set;
- γ denotes the output set which consists of K classes, $\gamma = \{c_1, c_2, ..., c_k\}$;
- X denotes a random variable that takes values within set χ;
- Y denotes a random variable that takes values within set γ;
- x and y are particular values of X and Y, respectively,
- D denotes the training set, where $D = \{(x^1, y^1), (x^2, y^2), ..., (x^M, y^M)\}$, with M being the number of input-output pairs of the set.

2.4.1.1 Generative Classifiers

A generative classifier learns a model using a joint probability of $\Pr(x, y)$ and tries to predict the class y for the instance x using the Bayes rule to calculate its probability $\Pr(y \mid x)$. This type of classifier does not directly assign a class to an instance, instead giving a probability of the instance being of the class and then picking the class which resulted in the highest probability.

A **naïve Bayes classifier** is a simple probabilistic classifier that uses the Bayes' theorem to classify the observations together with a crude simplification for how instances are formed. Naïve Bayes is a generative classifier, since it tries to estimate the class prior $\Pr(Y)$ and the class conditionals $\Pr(X \mid Y)$ using a training set D. This estimation is usually called training or learning. After training, when we obtain a new input $x \in \chi$, we will want to make a prediction according to:

$$\hat{y} = \arg \max_{y \in \gamma} \hat{\Pr}(y) \, \hat{\Pr}(x \mid y) \tag{2.5}$$

Making this prediction, using the estimated probabilities, is also called a-posteriori inference or decoding.

During training, we need to define the distributions for $\hat{\Pr}(y)$ and $\hat{\Pr}(x \mid y)$. For defining these distributions, we decompose the input X into J components, making a naïve assumption that all of the components are conditionally independent given the class:

$$X = (X^1, X^2, ..., X^J) \tag{2.6}$$

$$\Pr(x \mid y) = \prod_{j=1}^{J} \Pr(x_j \mid y) \tag{2.7}$$

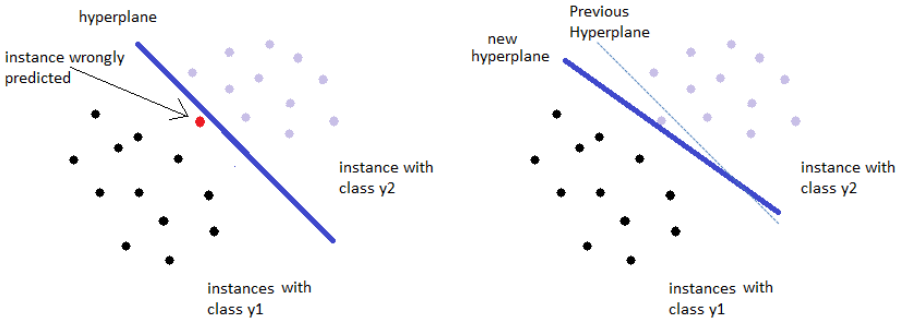


Figure 2.2: Representation of an adjustment of the decision hyperplane for a discriminative classification model, after an instance was wrongly predicted.

For the estimation of the parameters, we can use the maximum likelihood estimation procedure, maximizing the probability of the training samples.

$$\Pr(D) = \prod_{m=1}^{M} \Pr(y^m) \prod_{j=1}^{J} \Pr(x_j^m \mid y^m) \tag{2.8}$$

In practice, training naïve Bayes classifiers amounts to counting events in the training dataset, and normalizing these counts.

2.4.1.2 Discriminative Classifiers

A discriminative classifier creates and adapts models according to observed data (as in Figure 2.2). The instances of the training dataset can be seen as points in a given vector space where an hyperplane separates instances of different classes. The model created by this type of classifier is less dependent on the assumptions and distributions that are available initially, and, therefore, the predictions, are heavily dependent on the quality of the training data.

The **perceptron** is an example of an online discriminative classification algorithm. Its training procedure is based on receiving the input instances x one at a time, and for each, making a prediction. If the prediction is correct, then nothing is changed in the model, since it is correctly predicting elements of γ. However, if the prediction is wrong, then the model is adjusted accordingly.

In the case of binary classification problems, the perceptron algorithm, and also other linear classifiers, maps the inputs as belonging or not to the positive class, according to the weights

associated to the input features, the input, and a bias term:

$$\mathrm{f}(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \text{ with, } w \cdot x = \sum_{i=1}^{n} w_i x_i \tag{2.9}$$

The value of $\mathrm{f}(x)$ is the result of the classification, corresponding to 1 if x belongs to the positive class and 0 otherwise. The parameter w is a vector of weights, where each individual weight is modified at a period t, by an amount that is proportional to the product of the difference between y and the desired output for each d_j in the training set D. The parameter b is a bias, which shifts the decision boundary from the origin.

The pseudocode for the perceptron learning algorithm, is as follows:

1: Initialize the weights to small random numbers;
2: **repeat**
3: **for all** $j \in D$ **do**
4: Calculate the actual output as shown in Equation 2.9;
5: Update the weights according to a learning rate, α:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x \tag{2.10}$$

6: **end for**
7: **until** (iteration error $<$ threshold) or (iteration $=$ Maximum Number Of Rounds)

Since the perceptron is a linear classifier, if the instances of the training set are not linearly separable (i.e., if instances from different classes cannot be separated by a hyperplane), then the classifier will never get to a state where all the input vectors are correctly classified. While the perceptron is guaranteed to converge on some solution if the training set is linearly separable, the output of the classification might be one of many solutions with varying quality. To solve this problem, the perceptron of optimal stability was created. This new perceptron is more commonly known as the Support Vector Machine (Cortes & Vapnik, 1995).

The training of **Support Vector Machine (SVM)** classifiers, unlike the perceptron, usually works with a batch of instances instead of single data points. Also, instead of simply adjusting the hyperplane to encompass wrongly predicted class instances, it will adjust the model even if the

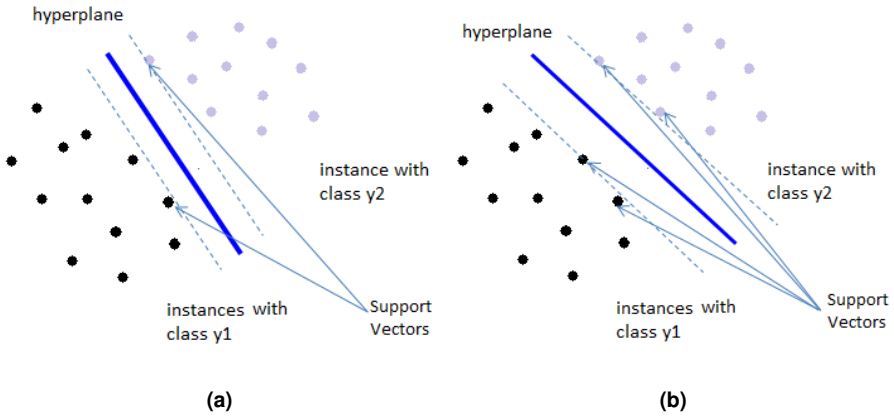


Figure 2.3: A model representation of the decision function from a SVM. In this example, the chosen hyperplane would be the one from (b), since the margin between the vector is the greatest.

prediction is correct, trying always to maximize the distance (or the margin) between the hyperplane and the instances of the different classes. A representation of a SVM model can be seen in Figure 2.3. In the figure, two different ways of separating the points of each class can be seen. An SVM will use different combinations of vectors (points) to try and obtain the largest possible margin. These points are called the support vectors.

There are at least two other well-known types of discriminative classifiers that have been used in previous work focusing on irony detection in textual messages, namely logistic regression and decision tree classifiers.

Logistic regression models are very similar to SVMs, but they have a probabilistic interpretation (i.e., the confidence in the classification is a probabilistic value obtained through a logistic function of a linear combination of the predictors). Details about this procedure can be found on the paper by Yu *et al.* (2011).

Decision tree classifiers create a tree data structure where, in each non-leaf node, a test condition over one of the features is made to assess the likelihood of an instance belonging to a given class. Leaf nodes correspond to possible classifications. At inference stage, the path of nodes is followed until reaching the leaves. Details about these procedures can be found on the paper by Quinlan (1986).

2.4.2 Deep Learning Algorithms

Another approach for the classification of textual documents involves the use of deep learning algorithms. Two major types of methods have been used, namely Convolutional Neural Networks

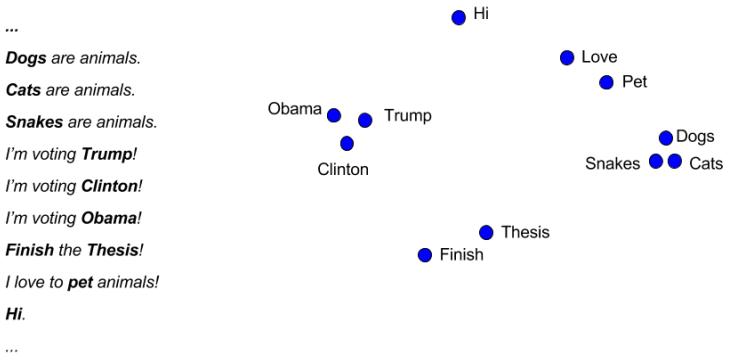


Figure 2.4: Example of a vector-space representation of words, using an algorithm that uses sentence similarity to create the embeddings space.

(CNNs) and Recurrent Neural Networks (RNNs), specifically a type of RNN commonly referred to as Long Short-Term Memory networks (LSTMs). These algorithms consist of a network of multiple connected layers, where each layer corresponds to a transformation of the input that involves hyper-parameters (e.g., activation functions).

These networks can also leverage external information in the form of embeddings for information on the input (e.g., word embeddings). Word embeddings consist of continuous vector-space representations for individual words, from which one can derive semantic relatedness between different words (Mikolov *et al.*, 2013b). The embeddings can be initialized randomly or initialized with pre-trained representations, which will be adjusted while the neural network is being trained. An example of the result of an algorithm that uses sentence similarity can be seen in Figure 2.4. The example shows that, when creating word embeddings through the use of similarity of sentences, the points representing each individual word that appear on similar sentences, are closer set into points closer to each other.

Although artificial neural networks have existed for some time, the required time to train a multi-layered neural network was, until recently, unfeasible for any kind of real world problem. The algorithm that solved this issue, and one of the reasons this approach has become more popular together with advances in computer hardware, is the backward propagation of errors algorithm (Rumelhart *et al.*, 1988), or just **backpropagation**. This algorithm is a method of training a multi-layered neural network that learns appropriate internal representations for the multiple layers, allowing the networks to learn any arbitrary mapping of input into an output.

The algorithm consists of re-iterating two phases until the performance of the network is satisfactory. The first is a propagation phase consisting of two steps: the forward propagation where, from the output one is able to obtain the error by comparing the prediction with the real values,

then, a backward propagation step minimizes that error (a lower error indicates a better performance). The second phase updates the output weights of each of the neurons.

The pseudo-code for the backpropagation algorithm is as follows:

1: Initialize the weights to small random numbers;
2: **repeat**
3: **for all** $d \in \chi$ **do**
4: prediction $=$ neural-network-output(d) ▷ (phase 1: step 1)
5: actual-label $=$ get-document-label(d)
6: compute-error(prediction, actual-label)
7: compute Δw_h for all weights from hidden layer to output layer ▷ (phase 1: step 2)
8: compute Δw_i for all weights from input layer to hidden layer
9: update-network-weights( ) ▷ (phase 2)
10: **end for**
11: **until** all predicted labels are correct or the error is low enough

2.4.2.1 Convolutional Neural Networks

A **convolutional neural network (CNN)** is a type of artificial neural network for modelling sequences (e.g., sequences of words), where the input is processed only one-way, from the input nodes through the hidden nodes to the output node (Kim, 2014).

The main building blocks for this type of neural network consist of one or multiple convolutional layers. These layers consider a window for how the input is processed, a window at a time, for example, for a convolutional layer with a windows size of 3, when processing the following sequence *this is how convolutions work*, the layer processes, first, *this is how*, followed by *is how convolutions*, and ending with *how convolutions work*.

After processing the input through the convolutional layers, the intermediate results go through an activation function where elements of the sub-sequence are combined, and then the results is processed by a pooling layer, which is a form of non-linear down-sampling. Usually a max-pooling function is used as the pooling layer where the maximum value of the activation function is obtained.

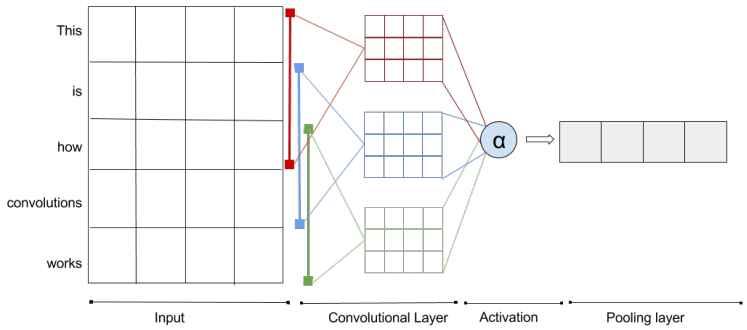


Figure 2.5: Representation of a convolutional neural network architecture with a single convolutional layer with window size 3.

2.4.2.2 Long Short-term Memories

The recurrent neural network architecture known as **Long Short-term Memory (LSTM)** is a neural network that is capable of learning long-term dependencies in sequences by persisting some of the input information (Hochreiter & Schmidhuber, 1997). A LSTM block remembers an input for an arbitrary length of time, saving the inputs that are significant enough to *remember*, and *forgeting* the other values. When the error is low enough (using the backpropagation algorithm) it only outputs the values that the LSTM block considered significant.

Consider Figure 2.6, where x_t is the input and h_t the output of an LSTM block for a position t in a given sequence. A typical LSTM block contains one or multiple input, output, and forget gates, as well as a memory cell that stores the significant values of the input. First, the block decides what information to discard using the forget gate. Then, the block decides which information should be stored on the memory cell (c_t). To do this, two steps are taken: decide which values should be updated (this task falls to the input gate and a vector of possible candidate values to be stored is created. The results from these two steps are then combined to create an updated memory state. Finally, having the state been stored on c_t, the output gate decides which are the values of h_t, according to the information stored on c_t.

2.5 Summary

This chapter introduced fundamental concepts, required to understand the following chapters of this dissertation.

Automated document classification methods were described, and I also described how those classifiers are usually evaluated. This was followed by a description of the common NLP tasks of sentiment analysis and opinion mining, pointing out the different classification approaches that have been used for addressing those tasks. These classification approaches consist of using

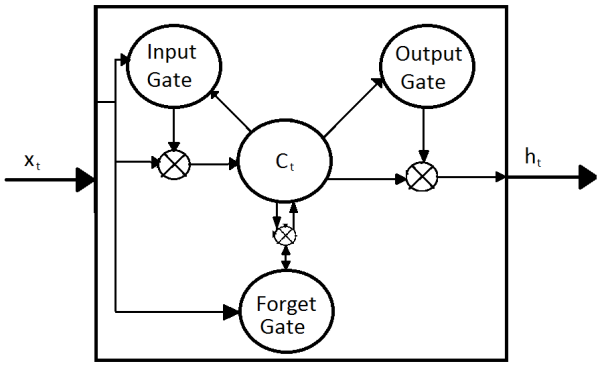


Figure 2.6: An LSTM block with a single input, output and forget gate.

machine learning algorithms, from generative classifiers (e.g., Naïve Bayes) to discriminative classifiers (e.g., SVMs), and nowadays also deep learning classifiers (e.g., CNNs). The chapter has succinctly described the backpropagation algorithm as well as convolution and recurrent neural networks.

Chapter 3

Related Work

This chapter describes previous studies focusing on the task of irony and sarcasm detection. First, I describe previous work in the task of finding common linguistic patterns and cues for capturing ironic utterances. Then, I describe studies which focus on finding the source of irony. Section 3.3 describes experiments that make use of external contextual information to improve the performance of irony classifiers. Finally, on the last section on this chapter, I describe recent studies that make use of neural networks to detect irony.

3.1 Finding Linguistic Cues in Ironic Utterances

The following studies try to find common linguistic cues on ironic utterances by using, for example, the length of a sentence and/or the occurrence of heavy punctuation and emoticons. In these studies, one can also find a heavy use of patterns that seem to appear on ironic utterances. There are also a few attempts at trying to infer emotions that can give a nuance of irony.

3.1.1 Clues for Detecting Irony in User-Generated Contents

Carvalho *et al.* (2009) explored a set of rule-based linguistic clues, associated with ironic Portuguese expressions over user-generated content, which have a negative connotation even though they can appear associated to positive expressions.

This approach consisted of using surface patterns (i.e., patterns that do not make use of possible underlying meanings behind the pattern) with 4-gram text phrases (i.e., sequences of four word tokens) that contained at least one prior positive adjective or noun. If those phrases contained any negative element (i.e., when considering a word with negative semantic value) they were excluded. The patterns used on the study are shown in Table 3.1.

pattern	match:
P_{dim}	$(4\text{-}Gram^{+}\ NE_{dim} \mid NE_{dim}\ 4\text{-}Gram^{+})$
P_{dem}	$DEM\ NE\ 4\text{-}Gram^{+}$
P_{itj}	$ITJ_{pos}\ (DEM\ ADJ_{pos})^{*}\ NE(?\mid!\mid ...)$
P_{verb}	$NE\ (tu)^{*}\ ser_{2s}\ 4\text{-}Gram^{+}$
P_{cross}	$(DEM \mid ART)\ (ADJ_{pos} \mid ADJ_{neut})\ de\ NE$
P_{punct}	$4\text{-}Gram^{+}$ (!! |?! |?!)
P_{quote}	$"(ADJ_{pos} \mid N_{pos})\{1,2\}"$
P_{laugh}	$(LOL \mid AH \mid EMO^{+})$

Table 3.1: Patterns used on the study of Carvalho *et al.* (2009). These patterns must have at least a positive adjective or noun and no negative elements.

	ironic	not-ironic	undecided	ambiguous
P_{itj}	44.88%	13.39%	40.94%	0.79%
P_{punct}	45.72%	**27.53**%	26.75%	0.00%
P_{quote}	**68.29**%	**21.95**%	21.95%	7.03%
P_{laugh}	**85.40**%	0.55%	11.13%	2.92%

Table 3.2: Results of the study of Carvalho *et al.* (2009), the most productive patterns, i.e., those with 100+ matches.

P_{dim} is a pattern that contains a diminutive named entity. A phrase matching the P_{dem} pattern must have a demonstrative determiner before a named entity. P_{itj} represents interjections, which usually appear frequently on subjective text and provide valuable information concerning the emotions, feelings and attitudes of the author. P_{verb} is a verb morphology pattern which, in this case, is a specific pattern for Portuguese text (i.e., the Portuguese language has two different pronoun expressions for *you*, namely *tu* which is used with people where there is degree of familiarity/proximity, and *você* which is used in a formal conversation). P_{cross} is a cross-construction pattern for common Portuguese expressions where the order of the adjective and noun are reversed through the use of a preposition (e.g., an expression such as *O comunista do ministro* which could roughly be translated to *The communist of the minister*). P_{punct} is a heavy punctuation pattern and P_{quote} is a pattern where quotation marks are used. Finally, emoticons and similar expressions are caught by the P_{laugh} pattern.

For the evaluation of the patterns, a set of 250,000 user posts (about one million sentences) were retrieved from the website of a popular Portuguese newspaper. For the patterns that picked up at least 100 sentences, a manual evaluation was carried out. The evaluated sentences were classified as ironic, not ironic, undecided (where there was not enough context) or ambiguous.

The patterns which had the most productivity were P_{itj}, P_{punct}, P_{quote} and P_{laugh}. The results for those patters are shown in Table 3.2.

With these results, it can be concluded that the P_{laugh} and P_{quote} patterns are good clues for detecting irony. Even though the P_{quote} pattern extracted a relative large portion of non-ironic

sentences, this can be justified by the fact that there are typical situations where quotes are used within non-ironic text (e.g., they are often used to delimit multi-word expressions and to differentiate technical terms or brands).

3.1.2 Semi-Supervised Recognition of Sarcastic Sentences

Davidov *et al.* (2010) proposed a semi-supervised algorithm for sarcasm identification, evaluating it on Amazon reviews and Twitter posts. The proposed procedure makes use of two modules:

1. A semi-supervised pattern acquisition approach for identifying sarcastic patterns that serve as features for a classifier;

2. A classification stage that assigns each sentence to a sarcastic or non-sarcastic class.

For the classification algorithm, labeled sentences were used as seeds. Those seed sentences were annotated with numbers on a scale from 1, which represents clear absence of sarcasm, to 5, representing a clearly sarcastic sentence.

Both lexical and pattern features were used in the feature vectors that represent sentences, where the main feature type was based on surface patterns. In these patterns, the targets of the utterance (e.g., product, company and author names) are abstracted into less specific tags.

The algorithm starts by creating generalized meta-tags for targets, users, links, and other targets. To extract patterns automatically, the words were classified as either high frequency words, which are words whose corpus frequency is above a threshold F_H, or content words, which are words whose corpus frequency is less than a threshold F_C.

After extracting hundreds of patterns, filtering was necessary to remove the patterns that were too general or too specific. Patterns that only appeared in a single product/book (i.e., patterns were inferred on a corpus of Amazon reviews), as well as patterns which occurred on sentences labeled with both 1 and 5 were removed, consequently filtering out generic and uninformative patterns from the training sets.

In addition to the pattern features, the authors also used the following lexical features: sentence length (in words), the number of occurrences for punctuation symbols ”!” and ”?”, quotes, and capitalized/all capitals words in the sentences.

The evaluation process consisted of two experiments. The first tested the pattern acquisition process, checking its consistency and evaluating to what extent it contributed to correct classification. In the second experiment, the authors evaluated the proposed method on a test set of unseen sentences, comparing their output to a gold standard created using Mechanical Turk.

On the first experiment, with the Amazon dataset, the precision achieved was 91.2% with an F1-Score of 82.7%. It is worth noting that using only pattern+punctuation features achieved comparable results, yielding a precision of 86.89% and an F1-score of 81.2%.

For the second experiment, on the Amazon dataset, the authors achieved a precision of 76.6%, with an F1-score of 78.8%. On the Twitter dataset, the results were similar, with 79.4% for precision and 82.7% for the F1-score.

3.1.3 A Closer Look at the Task of Identifying Sarcasm in Twitter

González-Ibáñez *et al.* (2011) reported on a study where lexical features, with an emphasis on pragmatic cues, were used to distinguish sarcasm from positive and negative sentiments. The study used Twitter messages that contained specific hash-tags that convey those sentiments (e.g *#sarcasm*, *#love* or *#joy*) as the golden standard.

For the lexical factors, unigrams and dictionary-based lexical features were used. The dictionary-based feature consisted of (i) a set of 64 word categories grouped into four general classes: linguistic processes, psychological processes, personal concerns and spoken categories; (ii) WordNet Affect (Strapparava & Valitutti, 2004), which labels words according to emotion; and (iii) a list of interjections and punctuation symbols. For the pragmatic factors three features were used, namely positive emoticons, negative emoticons and replies to other tweets.

The classification experiments involved SVM and logistic regression models leveraging the feature set described previously, where SVMs achieved better results.

On this study, a 3-way comparison of sarcastic, positive and negative messages, as well as 2-way comparisons of sarcastic and non-sarcastic, sarcastic and positive, and sarcastic and negative messages, was used. Given the context of this dissertation I will only describe the 2-way comparison of sarcastic and non-sarcastic messages.

Evaluation results yielded 65.44% of accuracy using the SVM approach and 63.17% of accuracy for the logistic regression model. These results were low, but they can be explained by the lack of explicit context in this type of messages (i.e., tweets), which makes the classification of sarcastic phrases difficult, whether it is made by machines or by humans.

3.1.4 A Multidimensional Approach for Detecting Irony

Reyes *et al.* (2013) proposed a model capable of representing the most salient attributes of verbal irony in a text, using a set of discriminative features to distinguish ironic from non-ironic text. The model constructed in this study is composed by a set of textual features for recognizing irony at a linguistic level, and it was evaluated over tweets with the irony, education, humor and politics

tags, and along two dimensions, namely representativeness and relevance.

The proposed approach can be organized according to four types of contextual features:

Signatures are features characterized by typographical elements (e.g., punctuation or emoticons) and discursive elements that suggest opposition within a text. These features are composed by three dimensions:

- pointedness, i.e., phrases that contain sharp distinctions in the information transmitted;
- counter-factuality, i.e., phrases that hint at opposition or contradiction, (e.g., *about*, *nevertheless*, *yet*);
- temporal compression, i.e., phrases with elements related to opposition in time (e.g., *suddenly*, *abruptly*).

Unexpectedness is used to capture both temporal and contextual inconsistencies. This feature is composed by two dimensions:

- temporal imbalance, i.e., divergencies related to verbs (e.g., *I hate that when you get a girlfriend most of the girls that didn't want you all of a sudden want you!*);
- context imbalance, to capture inconsistencies within a context by estimating the semantic similarity of concepts in a text.

Style features have the objective of identifying recurring sequences (i.e., patterns) of textual elements which help recognize stylistic factors, suggestive of irony. These features are composed by three dimensions:

- character-grams;
- skip-grams, i.e., a phrase that can skip a word of the original phrase;
- polarity skip-grams, i.e., skip-grams that contain words with different polarities.

Emotional scenarios are used to characterize irony in terms of elements which symbolize abstractions that convey sentiments, attitudes, feelings, or moods. These features are composed by three dimensions:

- activation or arousal, i.e., the degree of response (either passive or active) exhibited in an emotional response;
- imagery, i.e., how easy it is to form a mental picture from a word;
- pleasantness or valence, i.e., degree of pleasure suggested by a word.

For the evaluation of the model, the authors assessed its performance within a classification task, and they also evaluated the different patterns considering their appropriateness and representativeness. In the evaluation of the representativeness of the model, the authors tested if the individual features can be correlated to ironic speech.

The results from the evaluation of the representativeness indicated that all dimensions, except pointedness and temporal imbalance, appear to not be particularly indicative of ironic speech. All dimensions, except these two, had higher representativeness in tweets which were tagged as ironic, while the pointedness and temporal imbalance had higher representativeness in tweets that contained the humor tag.

On the classification task, the proposed ideas were evaluated using naïve Bayes and decision tree classifiers, with both yielding similar F1-scores of 61% and 59%, respectively. It is worth mentioning that the model was evaluated using different combinations of the features. The authors observed that, while no single feature captures irony well, the combination of all four features provides a useful linguistic framework for detecting irony in text.

3.1.5 A Novel Approach for Modeling Sarcasm in Twitter

On the study of Rajadesingan *et al.* (2015), the direct use of words as features is avoided. The authors suggest a novel approach to reduce the complexity of the computational model by decreasing the number of required features and because typical sarcastic expressions are often culturally specific.

With this in mind, the authors implemented a decision tree classifier leveraging features involving frequency of words, written-spoken style, intensity of adverbs and adjectives, structure of a sentence (e.g., length, punctuation and emoticons), contrast in sentiments, synonyms and ambiguity.

The frequency and written-spoken features were used to detect unexpectedness, which is strongly related to situational irony. The structure feature helps capture the style of the writer. The intensity feature was used to capture expressions which might be antonymic to what is actually written. The synonyms feature is used because it is believed that sarcasm conveys two messages (the literal and the opposite of what was uttered) to the audience and the choice of terms is important in order to send both those messages at the same time. The ambiguity feature was also used due to the observation that words with many meanings have a higher probability of appearing in utterances that have more than one message implied. Finally, the sentiments feature was used to identify which sentiments characterize sarcastic utterances.

The experiments developed by the authors consisted of using a decision tree classifier leveraging the features mentioned above, over a dataset of 60000 tweets divided into different topics:

Sarcasm, Education, Humor, Irony, Politics and News. The proposed approach achieved the best results over the news topic, and the authors suggest that this is due to the use of more formal language which is easily distinguishable from sarcasm. They achieved the worst results over the irony topic. On average, the model yielded 83.6% precision, 84% recall, and 83.4% in terms of the F1-score.

3.2 Finding the Source of Irony

This section describes models that try to learn expressions that tend to change depending on whether the utterance was ironic or literal (e.g., an expression with negative polarity that is preceded with an expression with positive polarity). These expressions, with meanings that can change, often denote a sarcastic utterance.

3.2.1 Sarcasm as Contrast Between a Positive Sentiment and a Negative Situation

Riloff *et al.* (2013) presented a bootstrapping algorithm that automatically learns lists of positive sentiment phrases and negative situation phrases from sarcastic tweets. This approach relies heavily on pattern features and was evaluated on a Twitter dataset. From the observation that many sarcastic tweets have a positive sentiment contrasting with a negative situation, the authors proposed to identify sarcasm that arises from the contrast between positive sentiments (i.e., words like *love* and *enjoy*) with negative situations, (e.g., *waiting forever* and *being ignored*).

The authors started by learning negative situations using only the positive sentiment seed word *love* and a collection of sarcastic tweets. To learn these negative situations, they used a POS tagger to detect certain patterns. Specifically, the authors looked for unigrams tagged as a verb (V), 2-grams of words corresponding to certain patterns (e.g V+V, V+ADV, *to*+V, V+NOUN) and word 3-grams (i.e., similar to the 2-grams but capturing some types of verb phrases, like an infinitive verb phrase that includes an adverb, or an infinitive verb phrase followed by an adjective). These n-gram patterns must occur immediately after the positive sentiment words. The resulting POS patterns filter the best candidates for negative situation phrases.

Having obtained examples of negative situations, an analogous process can be used to find positive sentiments (i.e., find positive sentiments using the negative situations obtained from the previous process).

The bootstrapping process for finding more positive sentiments and negative situations that occur in sarcastic tweets involves only executing more iterations of the two steps that were mentioned above, as represented in Figure 3.7.

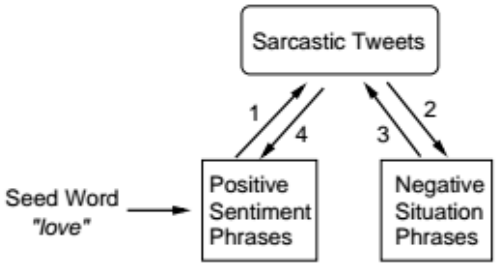


Figure 3.7: The process from Riloff *et al.* (2013) for the learning of negative situations and positive sentiments, using the seed word *love* and a collection of sarcastic tweets.

For the evaluation of their bootstrapped lexicon, the authors proposed an approach based on a constraint stating that a tweet is labeled as sarcastic only if it contains a positive verb phrase that precedes a negative situation in close proximity. This approach yielded an F1-score of 15% with a precision of 70% and a low recall of 9% due to the small and limited lexicon that is captured, when compared to other resources that contain terms with additional parts-of-speech (e.g., adjectives and nouns). However, using this bootstrapped lexicon to complement a baseline, created using an SVM classifier with unigram and bigram features (i.e., a baseline model which yielded an F1-score of 48% on its own) improves the F1-score by 3 p.p., increasing it to 51%. Although the contrast between a positive sentiment and a negative situation is a typical form of sarcasm, one such heuristic is limited to that specific form and ignores other forms of sarcastic utterances. This approach also had the problem that some positive sentiment/negative situation phrases were incorrectly being identified as sarcastic because even though they were usually used together when expressing sarcasm, they were not always meant sarcastically, (e.g., a phrase like *I love working*, which may or not be sarcastic depending on the context in which it is used).

3.2.2 Harnessing Context Incongruity for Sarcasm Detection

A similar study to that of Riloff *et al.* (2013) was described by Joshi *et al.* (2015). The authors presented a computational system that detects sarcasm using internal and external context incongruity, i.e., utterances that have expressions which covertly have implicit sentiments (e.g., *I love this paper so much that I made a doggy bag out of it*, where the underlined statement has an implied sentiment that is incongruous with the word *love*), and utterances that are openly expressed through sentiment words with both polarities. This study tried to improve on the work of Riloff *et al.* (2013) and, for the experimental setup, the authors used two tweet datasets, one consisting of 5208 tweets with the hash-tags *sarcastic* or *sarcasm* as the sarcastic tweets, and the hash-tags *notsarcastic* or *notsarcasm* as non-sarcastic. A total of 4170 of the tweets were sarcastic. The second dataset was the same used on the work of Riloff *et al.* (2013). The authors

also used a manually labeled discussion forum dataset, composed by 1502 forum posts of which 752 were sarcastic.

The proposed model uses lexical cues, implicit congruity, and external congruity as features. The lexical features consist of unigrams obtained using feature selection techniques and leveraging emoticons, punctuation marks, and capital words. The explicit incongruity features include the number of positive and negative words, length of contiguous sequences of positive/negative words, number of sentiment incongruities (i.e., the number of times a positive word is followed by a negative word, and vice versa), and the lexical polarity from the utterance extracted using the Lingpipe sentiment analysis system[1]. The implicit incongruity features are similar to those obtained by the process suggested by Riloff *et al.* (2013), where positive/negative sentiments and situations are extracted from the dataset (e.g., *being ignored by a friend*).

For the evaluation of the model, the authors used the aforementioned three datasets. SVM classifiers with different combinations of features were used in the experiments.

The best results were achieved using all features, having scored 61% in terms of the F1-score on the first dataset, 88.76% on the second dataset, and 64% on the discussion forum dataset. It is worth noting that the results over the second dataset were compared with those from the model suggested by Riloff *et al.* (2013). The model from Joshi *et al.* (2015) yielded better results, having improved the F1-score from 51% to 61%. An observation that can be made by examining the lower score achieved on the discussion forum dataset is that forum posts, unlike tweets, are more dependent on previous posts, since they are similar to a human conversation and they do not correspond to a single statement.

3.2.3 Word Embeddings to Predict the Literal or Sarcastic Meaning of Words

Ghosh *et al.* (2015) proposed an approach which involves detecting sarcasm by understanding if the *sense* of a word is literal or sarcastic. Their approach consisted of collecting words that can have either sense and detect if, in an utterance, those words are being employed in the literal or sarcastic sense.

To collect the words that can be literal or sarcastic, depending on context, the authors used Mechanical Turk, where turkers rephrase sarcastic messages into the possible intended meaning, substituting the part of the message that they found to be the source of the sarcasm. With this method, not only was it possible to identify the words that can have a sarcastic sense (i.e., the target words), but it was also possible to find the literal meanings of those words. The authors

[1] `http://alias-i.com/lingpipe/`

then used an unsupervised technique to obtain the opposite words/phrases for the meaning of those words, since those are the words that are most likely to give the literal meaning to the sarcastic utterance.

The dataset used for the evaluation of the model was composed by either sarcastic or literal tweets that contained the words collected by the model. With this dataset, the authors tried two different learning approaches. The first was a distributional approach, where each sense of a target word was represented as a context vector, i.e., a vector in which the target words appear in a corpus, as proxies for meaning representations. In the sarcastic/literal disambiguation task and for each word to disambiguate, two context-vectors were created representing each sense of the word. To then predict if the word on an utterance was used in a literal or sarcastic sense, a geometric technique like the cosine similarity was used, choosing the vector which obtained the maximum score. The second method was a classification approach, using an SVM model with lexical (i.e., interjections, bag-of-words, punctuation, emoticons and an online dictionary) and pattern (i.e., n-grams) features, and using a kernel which computes the semantic similarity of two tweets.

The results for the classification approach were better than those for the distributional approach (having yielded an average F1-score of 83.5%, which is better than that from the distributional approach by about 8 p.p.). The results also showed less variance with the classification approach.

3.3 Using External Context in the Detection of Irony

This section describes experiments focused on detecting irony using contextual cues, whether from information provided by the location of the utterance, or using historical information about the author of the utterance.

3.3.1 Sparse and Contextually Informed Models for Irony Detection

Wallace *et al.* (2015) carried out a study, using a discussion forum dataset, that combined lexical features with contextual features for irony detection. Specifically, the authors exploited four different types of features within different models: a sentiment feature for the inferred sentiment for a given comment, much like in the work of Riloff *et al.* (2013); the name of the forum where the comment was posted; the named entities mentioned in the comment; and, finally, a set of features that combines the two latter ones.

The use of features that rely on the name of the forum, named entities, and sentiments was considered in view of the assumption that individuals tend to write sarcastic utterances when discussing specific entities. These utterances will be sarcastic or not depending on the community

that is reading and commenting it. An example of this would be a community making positive comments on an entity that clearly has opposing beliefs (e.g., Republicans writing positive comments on the Affordable Care Act, commonly known as ObamaCare).

For the evaluation of this model, a Reddit[1] dataset was used, specifically considering subreddits with opposing beliefs. In this case, two opposing beliefs were used, namely progressive versus conservative subreddits, and Christianity versus atheism subreddits. The baseline of the experimental setup was a linear classifier that leverages unigrams and bigrams, together with features indicating grammatical cues, having a recall of 33.1% and a precision of 14.8%. The evaluation showed that the model, using all the features, was able to improve the recall of the baseline without harming the precision. The recall on the comments of the progressive/conservative subreddits improved by 4 p.p. (i.e., from 33.1% to around 37%), and for the Christianity/atheism subreddits the recall improved by 2.4 p.p. (i.e., from 33.1% to 35.5%).

3.3.2 Towards a Contextual Pragmatic Model to Detect Irony in Tweets

Karoui *et al.* (2015) tested the validity of two hypothesis which should help capture the pragmatic context needed to infer irony in tweets. The first hypothesis states that the detection of the disparity between the literal and the intended meaning of an utterance is related to the presence of negations as an internal property. The second hypothesis states that an utterance, which contains a false proposition, is ironic only if the absurdity of the true proposition can be assessed by the reader of the utterance.

To test the validity of these hypothesis, the authors proposed a three-step model. The evaluation dataset, which had tweets with different topics, was split into 9 categories: politics, sports, social media, artists, TV shows, countries or cities, the Arab Spring, and some generic topics. To capture the effect of the negation for the second hypothesis, the corpus was repartitioned into three different sets: tweets which had only negation, tweets with no negation and a single set with all the tweets of both sets.

The first step of the proposed model consists of the detection of an ironic tweet relying exclusively on the information internal to the tweet. The classifier used was an SVM model with lexical features. A few of the less common features that were used include:

- surface features that check the presence of discourse connectives that do not convey opposition (e.g., *hence*, *therefore* or *as a result*);
- sentiment features that check the presence of words that express surprise or astonishment, and the presence and number of neutral opinions;

[1] `https://www.reddit.com/`

	Ironic			Not Ironic		
	Precision	Recall	f-score	Precision	Recall	f-score
C_{Neg}	88.9%	56.0%	68.7%	67.9%	93.3%	78.5%
C_{NoNeg}	71.1%	65.1%	68.0%	67.8%	73.5%	70.5%
C_{All}	93.0%	81.6%	86.9%	83.6%	93.9%	88.4%

Table 3.3: Results of the classification procedure of Karoui *et al.* (2015) using the best combination of the lexical-based features.

	Experiment 1		Experiment 2	
Not ironic tweets for which:	All	Neg	All	Neg
Query applied	37	207	327	644
Results on Google	25	102	166	331
Class changed into ironic	5	35	69	178
Classifier accuracy	87.70%	74.46%	**87.70**%	7**4.46**%
Query-based accuracy	**88.50**%	**78.19**%	78.15%	62.98%

Table 3.4: Results of the two experiments reported by Karoui *et al.* (2015), using a first query-based method.

- sentiment shift features which check the presence of an opinion word which is in the scope of an intensifier adverb or modality;
- features based on internal context that deal with the presence or absence of personal pronouns, topic keywords, and named entities.

The results of this first step can be seen in Table 3.3, in which C_{Neg} is the negative only corpus, C_{NoNeg} is the no negation corpus and C_{All} is the corpus with both types of messages.

The second innovation of the proposed approach consists of using outside information about the subject of tweets containing negation, trying to correct the results of the classification. If the text of the tweet, with the negation removed, is found online, than the tweet is classified as ironic. To collect this information the authors query Google using its API and capture the results from sources that are reliable (e.g., Wikipedia and newspapers). The queries consisted of tweets with negations and with their text filtered (i.e., symbols, emoticons and negations were removed).

To evaluate this second step, two sets of experiments were performed. The first experiment evaluated the model on tweets with negation which were misclassified as not ironic. The second experiment evaluated the model on all tweets classified as non-ironic, whether the classification was correct or not. The results can be seen on Table 3.4, showing that this approach was able to improve the accuracy on the first experiment, but on the second experiment it substantially lowered it.

A conclusion that the authors drew from these results is that their method is not suitable for tweets which are personal or have a lack of internal context. With this in mind, two other experiments,

	Experiment 1		Experiment 2	
Not ironic tweets for which:	All	Neg	All	Neg
Query applied	0	18	40	18
Results on Google	-	12	17	12
Class changed into ironic	-	4	7	4
Classifier accuracy	87.70%	74.46%	**87.70**%	74.46%
Query-based accuracy	**87.70**%	**74.89**%	86.57%	**74.89**%

Table 3.5: Results of the two experiments from Karoui *et al.* (2015), using the query-based method which excluded tweets that were personal or had lack of context.

similar to the previous ones, were finally performed, although this time using different combinations of the relevant features. The most relevant feature for this experiment was the one based on internal context since it deals exactly with the problem of tweets with a personal subject or lack of internal context. The results are shown on Table 3.5 and they seem to indicate that internal context features are not particularly relevant for automatic detection of irony over tweets. However, an interesting observation is that internal context features can be used to detect tweets that are likely to be misclassified.

3.3.3 Contextualized Sarcasm Detection on Twitter

Bamman & Smith (2015) experimented with features derived from the local context of a message, from information about the author, from the audience, and from the immediate communication context between the author and its audience. For the experiments done by the authors, response tweets (i.e., tweets that are replies to some other tweets) were used as the evaluation dataset.

The model used four different classes of features:

Tweet features, which only use the internal context of the tweet being classified, specifically leveraging lexical-based and pattern-based cues, such as n-grams and word sentiments;

Author features, which use sentiment cues from the author's history of tweets, as well as cues from the profile of the user (e.g., gender and number of friends, followers and statuses);

Audience features, which use the similarity in interests between the author and the audience, and also the history of communication between them (e.g., number of interactions and number of references to each other) to capture the degree of familiarity between the author and the audience;

Environment features, which use lexical and pattern cues between the original tweet and the reply message.

Different combinations of features were used in the tests. The tweet features alone yielded an

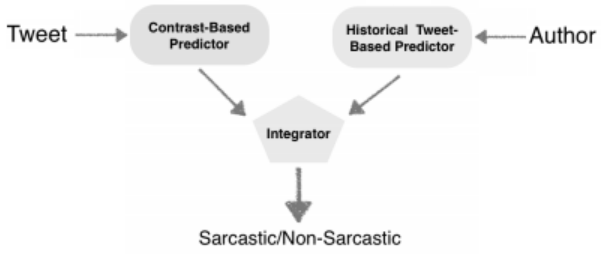


Figure 3.8: Architecture of approach suggested by Khattri *et al.* (2015).

accuracy of 75.4%, but when these features were combined with the author features, there was a gain of 9.5 p.p., pushing the accuracy of the model to 84.9%. When combining all the features mentioned above, the result was an accuracy of 85.1%.

3.3.4 Using an Author's Historical Tweets to Predict Sarcasm

In the study of Khattri *et al.* (2015), it is argued that historical text generated by an author helps with the problem of sarcasm detection in text written by that author. The study used tweets as the evaluation dataset.

The model proposed on this study consists of two components. The first component which uses both sentiment contrast, much like in the work of Riloff *et al.* (2013), and incongruity features similar to those from Joshi *et al.* (2015). This component is designated as contrast-based predictor. The other component, designated historical tweet-based predictor, identifies sentiments expressed by the author on previous tweets, and tries to match these sentiments with the tweet being classified.

The architecture of this approach also contains an integrator module that combines the features from the contrast-based predictor and makes use of historical tweets captured from the historical tweet-based predictor, as shown in Figure 3.8. The authors used four versions of this module to predict the class of the tweet:

1. Only the historical tweet-based component: In this case, the prediction is based only on the historical tweets. If the author of the tweet has not mentioned the target phrase before, the tweet is considered non-sarcastic.

2. An OR strategy: Only one of the predictors needs to return sarcastic to classify the tweet as sarcastic.

3. An AND strategy: Both predictors need to return sarcastic for the tweet to be classified as sarcastic.

4. A Relaxed-AND strategy: Similar to the previous case but if the historical predictor does not have any tweet, this version will use only the contrast-based predictor to classify the tweet.

The best results, using the dataset from Riloff *et al.* (2013), were achieved using the Relaxed-AND strategy, corresponding to a precision of 88.2% and an F1-score of 88%, which represents an improvement of 26 p.p. for the precision and 37.2 p.p. for the F1-score, over the original study from Riloff *et al.* (2013). The authors also stated that these results could be better if not for the assumption that the author has not been sarcastic about a target phrase in the past.

3.4 Using Deep Learning in the Detection of Irony

This last section describes two studies that make use of deep neural networks to detect irony in social media contents. The first study focuses on modeling the context of an author through specific embeddings, while the second study focuses on experimenting with different architectures as the classification approach.

3.4.1 Modeling Context with User Embeddings for Sarcasm Detection

Amir *et al.* (2016) proposed a neural network architecture that leverages pre-trained word embeddings and user embeddings. These last embedding vectors were automatically learned by a neural model using previous tweets from a given author. The authors created those representations by capturing relations between users and the content they produce, projecting similar users into nearby regions of the embedding space.

In this study, the authors used a CNN architecture to extract high level features from the word embeddings, which are then followed by a hidden layer which captures relations between the representations of the features captured by the CNN layers and the user embeddings.

The proposed model was evaluated on a subset from the twitter dataset used on the work of Bamman & Smith (2015), corresponding to a balanced dataset of 11,541 tweets. Besides the labeled tweets, the authors also extracted 1,000 tweets from each of the authors and from the users mentioned on the labeled tweets, with the purpose of modeling the user embeddings. The experiments considered a reimplementation of the feature set proposed by Bamman & Smith (2015) using a logistic regression classifier, and multiple versions of the neural network model (i.e., a standard CNN, or a CNN combined with the user embedding layers, leveraging pre-trained embeddings and embeddings learned by the model).

The accuracy yielded by the logistic regression classifier with the reimplemented feature set was 84.9%, while the best version of the novel architecture yielded an accuracy of 87.2%. One observation that can be taken from the results of the different neural network architectures was

that using pre-trained embeddings improved the results by 6%. Although the logistic regression classifier is only 2% worse than the neural model, the authors observed that the feature set required a lot of manual labor when compared to the work required to design the deep learning architecture.

3.4.2 Fracking Sarcasm using Neural Networks

Ghosh & Veale (2016) also proposed a neural network for the task of sarcasm detection. The authors tested several different architectures, individually and combined, which were then tuned, by adding or removing layers and changing the hyper-parameters, to achieve the best result possible. The study compared the result of the proposed neural networks with models from previous studies mentioned in this chapter, namely from Davidov *et al.* (2010) and Riloff *et al.* (2013).

The architectures used on this study involved LSTMs, CNNs and DNNs (i.e., fully connected feed-forward neural networks). The authors found that LSTMs have the capacity to remember long distance temporal dependencies and CNNs are able to capture temporal text patterns for shorter texts. The authors also found that having a fully connected layer after a LSTM layer can provide better classification results since this type of layer maps features into a more separable space.

The experiments were performed on three twitter datasets, one from the work of Riloff *et al.* (2013), another from the work of Davidov *et al.* (2010), and finally a dataset created by the authors where tweets with a sarcasm hash-tag, and other hash-tags that were also indicative of sarcasm (e.g., *#yeahright*) were labeled as sarcastic, and all others, which did not have any indication of being sarcastic, were labeled as not sarcastic. This last dataset contained 39,000 tweets, with a balanced number of sarcastic and not sarcastic tweets. For testing purposes, 2,000 of the automatically labeled tweets were also manually labeled.

On the dataset extracted by the authors, both the CNN and LSTM architectures achieved similar results, yielding a F1-score of 87.2% and 87.9%, respectively. However the combination of the CNN, LSTM and DNN architectures yielded an F1-score of 92.1%, which is significantly better than the results for the individual architectures.

On the dataset of Riloff *et al.* (2013) the combined neural network architecture achieved 88.1% in terms of the F1-score (against the original 51%), and on the dataset of Davidov *et al.* (2010) the same architecture achieved an F1-score of 90.1% (against the original 82.7%).

It is worth noticing that the good results achieved on this study did not require feature engineering, but it took a substantial amount of time to train the neural networks.

Study	Features	Dataset	Metrics		External context
			Accuracy	F1-score	
Carvalho *et al.* (2009)	lexical + patterns	article comments	-	-	-
Davidov *et al.* (2010)	lexical + patterns	tweets	-	55%	-
Davidov *et al.* (2010)	lexical + patterns	reviews	-	83%	-
González-Ibáñez *et al.* (2011)	lexical	tweets	65%	-	-
Reyes *et al.* (2013)	lexical + pattern	tweets	-	76%	-
Rajadesingan *et al.* (2015)	lexical	tweets	-	83%	-
Riloff *et al.* (2013)	patterns	tweets	-	51%	external sentiments/situations
Joshi *et al.* (2015)	lexical + incongruity	tweets	-	89%	-
Joshi *et al.* (2015)	lexical + incongruity	forum posts	-	64%	-
Ghosh *et al.* (2015)	lexical + patterns	tweets	-	84%	literal meaning of words
Wallace *et al.* (2015)	lexical	forum posts	-	-	type of thread
Karoui *et al.* (2015)	lexical	tweets	88%	-	subject of utterance
Bamman & Smith (2015)	lexical + pattern	tweets	85%	-	history of author/audience
Khattri *et al.* (2015)	lexical + incongruity	tweets	-	88%	history of author

Table 3.6: Summary of the related work analyzed in this report, regarding methods leveraging feature engineering

Study	Architecture	Dataset	Metrics		External context
			F1-Score	Accuracy	
Amir *et al.* (2016)	CNN	tweets	-	87.2%	past user tweets
Ghosh & Veale (2016)	CNN+LSTM+Dense	tweets	90.1%	-	-

Table 3.7: Summary of the related work regarding neural networks analyzed in this report.

3.5 Summary

Automatic irony detection relies heavily on the use of linguistic cues (i.e., semantic, lexical and pattern features). In general, approaches that rely more on semantic cues (Riloff *et al.* (2013), Joshi *et al.* (2015), Ghosh *et al.* (2015)) are based on getting the sense of individual words (e.g., emotional words tend to appear more on sarcastic sentences, or words with different polarities in the same sentence tend to appear on sarcastic utterances). Approaches that rely more on syntactic cues (Carvalho *et al.* (2009), Davidov *et al.* (2010), González-Ibáñez *et al.* (2011), Reyes *et al.* (2013)), are usually employed to identify patterns and expressions that tend to appear on sarcastic utterances. Pragmatic cues involving punctuation marks and emoticons are almost a standard in all irony detection models. Nonetheless, it has been noted on most previous studies that using only linguistic cues is not enough for this task. On more recent studies (Wallace *et al.* (2015), Karoui *et al.* (2015), Bamman & Smith (2015), Khattri *et al.* (2015)), contextual cues have started to be used to improve the performance of classifiers. On the year of the writing of this dissertation a few studies using neural networks to detect sarcasm have been published (e.g., Ghosh & Veale (2016) and Amir *et al.* (2016)), which show promising results with this novel paradigm on irony detection over textual messages, which does not require a substantial amount of feature engineering.

A summary of the related work that was surveyed on this report can be seen in Tables 3.6 and 3.7. Table 3.6 focuses on feature-based methods, showing which type of features they explored, the type of dataset used in the evaluation, the metrics and what kind of external context to the text has been used on the classical machine learning classifiers. Table 3.7 instead focuses on the methods exploring deep neural networks, showing the same information but, instead of summarizing the features, showing the architecture used on each of the studies.

Chapter 4

Deep Neural Networks for Irony Detection

The experiments reported in this dissertation involved two types of approaches, namely standard machine learning algorithms relying on feature engineering, and implemented through the scikit-learn (Pedregosa *et al.*, 2011) library, and deep learning algorithms relying on implementations from the Keras library[1].

4.1 Introduction

Deep learning is a class of methods that have increasingly been used on NLP tasks such as sentiment analysis. These methods have also very recently started to be used on the task of detecting irony in text.

The purpose of using these algorithms in my experiments was to verify if, as has been happening in other text classification tasks, these methods could yield better results than the standard machine learning algorithms based on extensive feature engineering. Experiments were performed with datasets from different domains, namely a discussion forum dataset (i.e., posts collected from Reddit) from the work of Wallace *et al.* (2014), two Twitter datasets from the works of Riloff *et al.* (2013) and Bamman & Smith (2015), and a Portuguese news headline dataset.

[1] https://github.com/fchollet/keras

Machine Learning Algorithms	Deep Learning Algorithms
Naïve Bayes	LSTM Stack
Logistic Regression	CNN
Support Vector Machine	Bidirectional LSTM
Naïve Bayes-SVM	CNN-LSTM

Table 4.8: The classifiers used in the experiments reported on this dissertation.

4.2 The Considered Classification Methods

This section details the different types of classification, both, featured-based methods, as well as deep learning methods. A list of the classifiers/architectures used in these experiments can be found on Table 4.8.

4.2.1 Feature-Based Machine Learning Algorithms

Baseline methods were based on common classification approaches (machine learning algorithms leveraging simple features), namely Naïve Bayes, Logistic Regression and SVM classifiers. An SVM that leverages from the results yielded by a Naïve Bayes classifier (i.e., a model inspired on the work of Wang & Manning (2012)), was also used on the experiments.

These classifiers relied on simple features like unigrams and bigrams, term frequency-inverse document frequency, or simply the number of term occurrences when building the representations (depending on the datasets). English stop words[1] were removed, but the representations kept punctuation and emoticons.

Irony has been observed to be related to the emotions expressed on an utterance (Reyes *et al.*, 2013; Riloff *et al.*, 2013), another baseline feature used on the Riloff dataset was based on the contrast of affective norm (Warriner *et al.*, 2013) associated to words within a sentence. For the featured-based classifiers, this feature corresponds to the average of the arousal, valence and dominance of all the words within sentence where the average belongs to one of three groups, low ($average < 3$) or medium ($3 \leq average \leq 7$) or high ($average > 7$), for each affective norm. For example, considering that the words *hate*, *being*, and *emotional* have each a respective valance rating of, 3, 7, and 2 and an arousal rating of 9, 7, and 8. Since the average of the valance and arousal ratings for those words are 4 and 8, respectively, the phrase *hate being emotional* belongs to the medium valence group and the high arousal group. This allows for the classifier to capture existing contrasts of low and high values of the different affective norms.

Due to the fact that the Portuguese news headline dataset consisted of Portuguese texts and

[1] `http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words`

Dataset	BoW Representation	Phrase Grams	Other Features
Reddit	term count	Unigrams & Bigrams	punctuation, emoticons
Twitter Riloff	tf-idf	Unigrams & Bigrams	punctuation, emoticons, affective norms
Twitter Bamman	term count	Unigrams & Bigrams	punctuation, emoticons
News Headlines	term count	Unigrams & Bigrams	punctuation, emoticons

Table 4.9: The different features for each of the datasets.

contains a formality (e.g., it does not make use of extra punctuation to imply irony) that is uncommon on the other social media datasets, only the bag-of-words with unigrams and bigrams was considered for the classification task over this particular dataset.

A summary of the features can be seen on Table 4.9.

4.2.2 Neural Network Architectures

In this dissertation, four main different types of layers were used in the neural network models:

- Dense Layer - a fully connected layer;
- Convolution Layer - a convolution operator for filtering neighborhoods of sequential inputs;
- Max Pooling - a form of non-linear down-sampling used after a convolution layer;
- LSTM layer - a Long-Short Term Memory unit capable of learning long distance dependencies.

The neural networks architectures used in the experiments included two variants of RNNs, namely a stack of two LSTM layers and a bidirectional LSTM. A CNN architecture and a hybrid CNN-LSTM architecture were also used. These architectures were chosen due to the good results from studies that use them in similar tasks (Ghosh & Veale, 2016; Kim, 2014; Zhou *et al.*, 2015).

All of the architectures start with an input layer where the texts are represented as vectors of the word identifiers, followed by an embedding layer. The architectures also had multiple dropout layers with increasingly smaller dropout rates, to try addressing overfitting problems with the different datasets. However, the tests revealed that the dropouts had almost no effect on the results.

The LSTM layers and the final dense layers used a *sigmoid* activation function, which is a function defined for all real input values that has a positive derivative at each point (see Equation 4.11). The convolutional layers used a rectified linear activation function (Nair & Hinton, 2010).

$$S(t) = \frac{1}{1 + e^{-t}} \tag{4.11}$$

4.2.2.1 Stack of Two LSTM

The stack of two LSTM layers is a simple sequential model where, following the input and embedding layers, we have a pair of two LSTM layers which processes the input sequence normally (i.e., in the order it is received), and ending with a dense layer which sets the output into the target output shape (i.e., a single value where 0 represents a not-ironic prediction and 1 an ironic prediction).

4.2.2.2 Bidirectional LSTM Layers

The bidirectional LSTM is an architecture running multiple layers in parallel. The model starts with the input layer and an embedding layer, which is then split into two pairs of LSTM layers running in parallel, where one of the pairs processes the input from left-to-right and the other pair processes the input from right-to-left. The output of the LSTM layers is then merged into a dense layer to obtain the correct output shape.

4.2.2.3 Convolutional Neural Network

The Convolutional Neural Network (CNN) architecture also runs in parallel, considering multiple parallel convolutional layers with different filter windows. Several combination of filter windows were tested but the results did not change enough for this to be relevant. Because of this, the experiments reported on this dissertation were made with the combination of filters with length 3, 5, and 7. After each convolutional layer the data goes through a max pooling layer and then had the output of the max pool is flattened into a shape that can be accepted by the final dense layer (i.e., we go from a three dimensional matrix to a two dimensional matrix).

4.2.2.4 CNN-LSTM

Finally, the hybrid CNN-LSTM architecture corresponds to a combination of an LSTM layer, and the CNN architecture described previously. After the initial layers, we have a CNN that is followed an LSTM Layer, with a dropout layer in between, before having the dense layer that prepares the output. This architecture was mostly based on the work done by Ghosh & Veale (2016). The architecture design for the CNN-LSTM is shown in Figure 4.9.

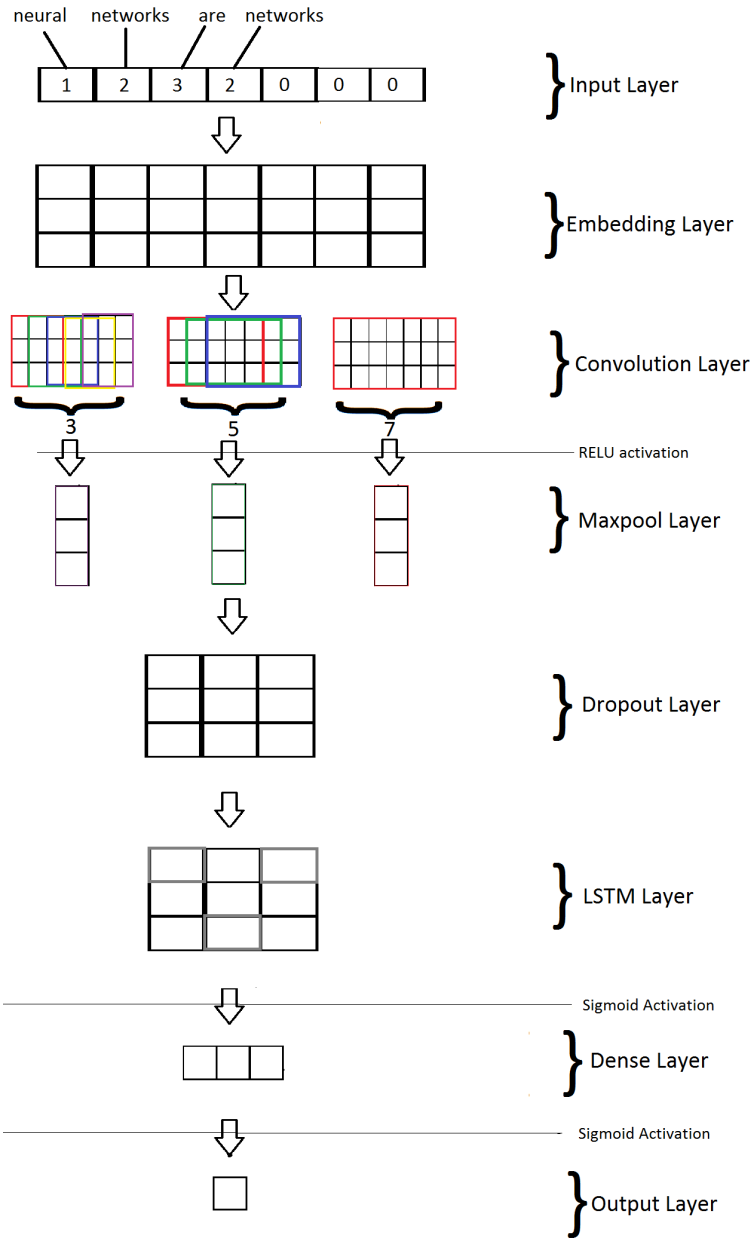


Figure 4.9: The CNN-LSTM architecture used in the experiments.

4.2.2.5 Semantic Modeling

As discussed in Chapter 2, neural networks use embeddings to give a semantic representation to the words that form the documents that are being classified.

A specific experiment used different word representations for each of the domains. Each dataset, from a different domain, had specific word embeddings trained on documents from that domain, to help the neural networks achieve the best results possible on the task of irony detection over the different domains (i.e., Reddit, Twitter, and news headlines).

However, even if no pre-trained word embedding exist, one can initialize the embeddings randomly and as the neural network is being trained, those embeddings will be adjusted according to the labeled training data. The disadvantage of this approach is that, since we are dealing with relatively small datasets, it is very difficult for the neural network to obtain good embeddings from a random initialization. Consequently, these models will not be able to achieve as good results as those using pre-trained embeddings. With this in mind, for this experiment, the neural networks leveraged from random word embeddings as the baseline in order to verify the performance gain of using different pre-trained embeddings.

For the Reddit domain, two different word embeddings were used. One of these resources was a publicly available word embedding dataset extracted from Wikipedia[1] documents and created by Mikolov *et al.* (2013a). The other was a Reddit word embedding created using the word2vec tool (Mikolov *et al.*, 2013b) on a large dataset of Reddit comments which were from the same subreddits (i.e., a specific forum for a given topic) of the labeled dataset being classified.

For the Twitter domain, two different word embedding datasets, specific to Twitter texts, were used. One of these datasets was optimized for named-entity recognition (Godin *et al.*, 2013), and the other was created for the study of Amir *et al.* (2016) on irony detection, and has already achieved good results on one of the Twitter datasets that has also been used on the work being described on this dissertation.

Finally, for the news headlines domain, a word embedding dataset with Portuguese words was created from a dataset of news articles belonging to the online newspaper *Público*[2].

Note that, when multiple pre-trained word embeddings are available, multiple embeddings can be used in a single architecture, like the one shown on Figure 4.10. Although this will slow the training of the network (i.e., many more parameters will have to be considered, it produces a small gain on the performance over the classification task.

Another experiment related to modeling word embeddings is also reported on this dissertation.

[1] https://www.wikipedia.org/
[2] https://www.publico.pt/

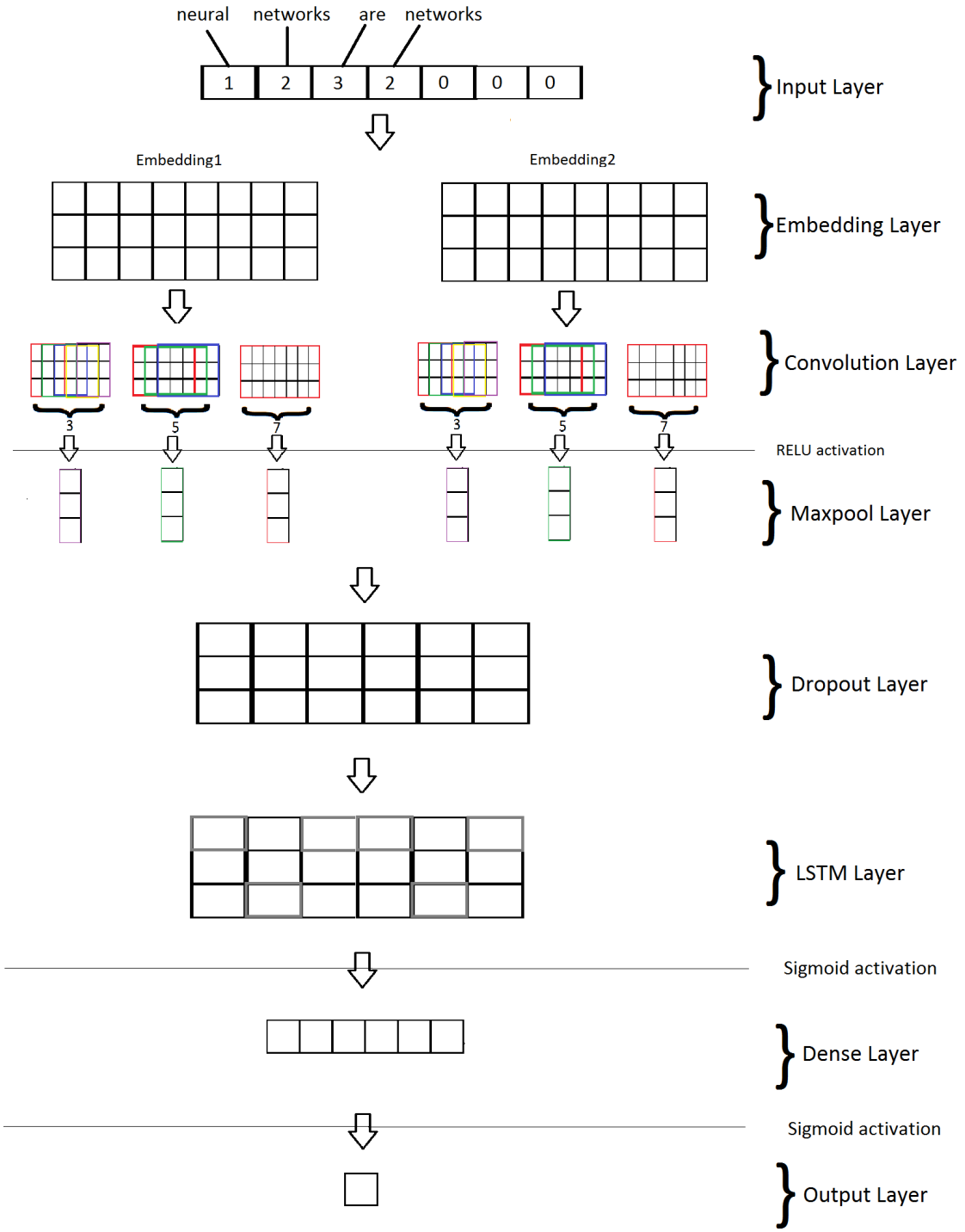


Figure 4.10: A CNN-LSTM architecture leveraging from multiple embeddings.

Previous studies found that irony has a relation to the emotions being conveyed by the text. One possible representation for those emotions is based on the affective norms of words and phrases. An existing dataset of words and their affective ratings has been created by Warriner *et al.* (2013) through a crowdsourcing methodology. However, this dataset does not contain sufficient labeled words (for the datasets being classified) to obtain a word embedding that contains enough semantic information to have any significant effect on the result of a neural network.

One attempt to increase the size of this dataset was to make use of a dataset of paraphrases (Ganitkevitch *et al.*, 2013) to find similar phrases to the words and short phrases that are included in the Warriner *et al.* (2013) dataset. The paraphrases would then be inserted into the affective norms dataset with the same affective values as the original phrase, if the paraphrase had a certain similarity rating (rated from 0 to 1, where 1 indicates a very similar phrase). Since high (> 0.75) or even medium (> 0.5) values of similarity yielded a small increase (less than 1,000) in the size of the affective norms dataset, a lower similarity of 0.25 was used, which allowed us to substantially extend the original affective norms dataset. This method was reiterated several times using the collected paraphrases to collect even more paraphrases until no more paraphrases could be found. With this, it was possible to increase the number of phrases from the original 13,915 to 135,746 labeled phrases.

To verify the viability of the paraphrasing method, the extended dataset was run against other affective norm datasets from Francisco *et al.* (2012), Bradley & Lang (2007), and Preoţiuc-Pietro *et al.* (2016).

To create the word embeddings of affective norms, the same CNN-LSTM architecture used to classify ironic texts (Figure 4.9) was used on the training of a model, that used the extended affective norms dataset as the training set, to predict the affective norms, valence and arousal, of words that were not included on the original affective norms dataset.

After obtaining the values from the predicted affective norms, the CNN-LSTM neural network was modified to incorporate the embeddings using a LSTM layer, and then merging its output into the final layer of the original architecture.

4.3 Summary

This chapter described the algorithms and architectures used in the experiments that were made in the context of my Msc research project, starting with the machine learning approaches based on feature engineering (e.g., Logistic Regression and SVM classifier), leveraging simple features

from previous work on similar text classification problems (e.g., punctuation and emoticons), followed by an approach to irony detection based on deep neural networks. Several different architectures were described, from RNNs (i.e., two stacks of LSTM layers and a bidirectional LSTM) and convolutional architectures, to hybrid approaches (the CNN-LSTM). The chapter ended with a description of the different word embeddings that were used within the aforementioned architectures, obtained from datasets with specific domains.

Chapter 5

Experimental Evaluation

This chapter presents the experimental evaluation of the different approaches that were described in the previous chapter. It starts by describing each dataset, and the experiments carried out with each dataset. It then it reports on the results of those experiments. Finally, there is a discussion about the obtained results.

5.1 Datasets

Four different datasets were used in the experiments, namely two sets of documents collected from Twitter, one from the discussion forum named Reddit and, a Portuguese news headlines dataset. A summary of the datasets used in the experiments can be found in Table 5.10.

5.1.1 Reddit Dataset

The first dataset is a labeled Reddit dataset (Wallace *et al.*, 2014). It consists of comments from several subreddits (i.e., forums of different topics) based on politics and religion. It has 1949 labeled comments and, of these, 537 were labeled as sarcastic and 1412 as not sarcastic.

Dataset	Ironic Instances	Non Ironic Instances	Total
Wallace *et al.* (2014)	537	1412	1949
Riloff *et al.* (2013)	453	1665	2118
Bamman & Smith (2015)	6478	4907	11457
Inímigo Público	4335	4314	8649

Table 5.10: The different datasets used in the experiments and their respective size.

5.1.2 Twitter Datasets

The Twitter datasets used in the experiments were described in the studies of Riloff *et al.* (2013) and Bamman & Smith (2015).

The original datasets used in those studies could not be collected in full. For instance, the dataset of Riloff *et al.* (2013) consisted of 3200 labeled tweets, but by the time my research project started it was only possible to extract 2118 of the tweets from the original dataset, since some of the data could not be extracted with the Twitter API for various reasons (e.g., the tweet was deleted or the user changed permissions). From those 2118 labeled tweets, 453 were labeled as sarcastic and 1665 were not sarcastic.

In the dataset from Bamman & Smith (2015) the same issue occurred. From the original 19,534 tweets, only 11,457 were possible to extract using the Twitter API. The labels ended being distributed into 6,478 ironic tweets and 4,907 that were not ironic.

5.1.3 Portuguese News Headline Dataset

The Portuguese News headline dataset was collected from two news sources. The first was a factual news website called *Público* from which we got the data labeled as not ironic. The source for the ironic news headlines was a satirical news website called *Inimigo Público*. The overall dataset is balanced, consisting of 4,335 ironic headlines from *Inimigo Público* and 4,313 not ironic headlines from *Público*.

5.1.4 Datasets of Affective Norms

To test the viability of the paraphrase approach over the affective norms dataset from Warriner *et al.* (2013), described in Chapter 4, other datasets labeled with the same norms were also used.

Specifically, three different datasets were used as testing data. One of these was a dataset of Facebook messages from the work of Preoţiuc-Pietro *et al.* (2016), consisting of 2,895 Facebook messages labeled with valence and arousal. Another dataset from the work of Francisco *et al.* (2012) contained texts collected from folk tales where each of the 1,397 phrases were labeled with valence, arousal, and dominance. Finally, a small set of brief texts in the English language, created by Bradley & Lang (2007), was also used, consisting of 100 phrases labeled with valence, arousal and dominance. Because not all of these datasets have the values for the dominance norm, and also because of its lesser importance, this affective norm was discarded in the experiments.

Dataset	Size	Affective Norms
Preoţiuc-Pietro *et al.* (2016)	2895	Valence & Arousal
Francisco *et al.* (2012)	1397	Valence, Arousal & Dominance
Bradley & Lang (2007)	100	Valence, Arousal & Dominance

Table 5.11: The different datasets used in the experiments and their respective size.

5.2 Methodology and Evaluation Metrics

The experimental setup consisted of performing different tests involving neural networks, and comparing their performance with the feature-based machine learning approaches.

The first set of experiments consisted of running each of the algorithms/classifiers over each of the datasets described in the previous section, comparing the results of the feature-based and neural network classifiers. For this experiment, the neural networks leveraged the publicly available word embeddings described in Chapter 4.

The second set of experiments consisted of running the neural networks leveraging from different combination of embeddings, to compare the performance of using different word embeddings on the task of irony detection. Three tests were performed for each of the Reddit and Twitter datasets. The first test consisted on using a randomly initialized word embedding. On the second test, the pre-trained word embeddings described on Chapter 4 were used individually. The third test involved using a neural network that used two word embeddings in a single architecture. For the Portuguese news headline dataset, only random embeddings and a single word embedding matrix were used.

The third experiment consisted on verifying the viability of the paraphrasing task over the affective norms dataset described in the previous chapter, later and using the affective norms as embeddings on the classification of irony using neural networks.

The metrics used for the task of classifying ironic texts, on each of the datasets, are the F1-score and accuracy, where the F1-score reported is the average of the values for the positive and negative class labels. Notice that the Reddit dataset and the Twitter dataset of Riloff *et al.* (2013) are unbalanced datasets where, if every comment was labeled as not ironic, the classifier would automatically yield an accuracy above 70%. For these two datasets, it is more important to focus on the F1-score than on the accuracy, since this metric gives more precise information on the performance over the classification task. The results reported are the average from a 5-fold validation. To verify the significance of the improvement of the performance of the neural networks against the feature-base classifiers, the mid-P McNemar statistical test (Fagerland *et al.*, 2013) was used. The improvement was considered significant for a 95%-confidence level.

Ckassufuer	Reddit		Twitter Riloff		Twitter Bamman		News Headlines	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
SVM	60%	57.9%	66%	63.1%	65%	65%	83%	83.1%
Naïve Bayes	28%	33.8%	39%	38.1%	68%	67.9%	83%	84.7%
Logistic Regression	**65%**	**68.1%**	**80%**	**81.4%**	69%	69.2%	**85%**	**84.7%**
Naïve Bayes-SVM	53%	50.2%	58%	53.8%	**70%**	**69.4%**	84%	84.4%

Table 5.12: The results of the experiments involving each of the classic learning algorithms.

For evaluating the results from the experiments performed with the affective norm representation, two metrics were used: the root mean squared error, to evaluate the deviation between the prediction and the expected value, and the Pearson correlation, to evaluate if there is any correlation (positive or negative), or simply no correlation between the expected values and the predicted results.

Both the valence and the arousal root mean squared errors and Pearson correlations were measured separately and, for comparison, both metrics were used on the results for the original affective norm dataset, and for the dataset extended with the paraphrases.

5.3 Experimental Results

The results of the experiments are presented in this section. First, the section reports on the results from the feature-based machine learning algorithms, where it is possible to compare the performance of the classifiers with and without the features described in the previous chapter. Next, the section shows the results achieved using the neural network architectures. Finally, the results yielded by the experiments involving the semantic modeling are reported.

5.3.1 Feature-based Machine Learning Algorithms

The first experiment involved running a classification approach based on classic machine learning algorithms (i.e., Naïve Bayes, Logistic Regression, SVM, and the Naïve Bayes-SVM), with just a bag-of-words representation, as mentioned in Chapter 4. The results from each of the different classifiers varied substantially depending on the datasets, as can be observed in Table 5.12. Notice that the Logistic Regression classifier achieved the best, or close to the best, overall results.

On Table 5.13, you can find the results when the extra features are added to the classifiers. The logistic regression classifier still yielded the best overall results, but, it did not exhibit a relevant improvement with the features, while the SVM classifier had a gain over the F1-score from 2 p.p. to 12 p.p. depending on the dataset. It is worth noting that the SVM classifier leveraging the

Classifier	Reddit		Twitter Riloff		Twitter Bamman		News Headlines	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
SVM	65%	64.4%	79%	80.2%	68%	67.8%	83%	83.1%
Naïve Bayes	26%	33.0%	37%	36.6%	69%	68.4%	83%	84.7%
Logistic Regression	**66%**	**68.9%**	**79%**	**80.4%**	70%	69.4%	**85%**	**84.7%**
Naïve Bayes-SVM	54%	51.0%	61%	56.8%	**70%**	**69.9%**	84%	84.4%

Table 5.13: The results of the experiments involving each of the standard machine learning classifiers with simple lexical features (i.e., emoticons and punctuation for the Reddit and Twitter datasets and also the averaged affective norms for Riloff's dataset.

Architecture	Reddit		Twitter Riloff		Twitter Bamman		News Headlines	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
Stack of two LSTMs	67%	68.8%	77%	78.1%	69%	68.8%	89%	89.0%
Bidirectional LSTM	65%	66.9%	79%	79.7%	68%	68.1%	90%	90.2%
CNN	66%	70.3%	77%	78.2%	67%	67.5%	90%	90.4%
CNN-LSTM	**67%**	**69.5%**	**80%**	**82.0%**	**70%**	**70.5%**	**91%**	**90.7%**

Table 5.14: The results from the experiments involving each neural network architecture.

affective norms feature increased the F1-score by 7 p.p. on Riloff's Twitter dataset.

5.3.2 Neural Network Approaches

With the neural network classifiers, each of the different architectures yielded quite similar results, as shown on Table 5.14. The CNN-LSTM architecture consistently yielded the best results. You can observe that the CNN-LSTM performed better than the classic machine learning classifiers, even with the added lexical features over all the datasets, yielding a 1 p.p. increase over the F1-score over the Reddit dataset and over Riloff's Twitter dataset.

5.3.2.1 Semantic Modeling Results

As described in Chapter 4, experiments involved different combinations of embeddings, according to the domain of the datasets. The idea was to test the impact the embeddings had on the neural networks.

The first results shown in Table 5.15 were obtained with the Reddit dataset, with four sets of word embeddings, namely a randomly initialized embedding matrix, a publicly available word embedding dataset from Wikipedia, word embeddings created using word2vec on a large Reddit dataset, and finally a combination of the Wikipedia and Reddit word embeddings (i.e., the vector representations on each dataset were concatenated). The improvement over the logistic regression model was quite significant (with a p-value of 0.027 over the McNemar test) having yielded an improvement of 3 p.p. on the F1-score.

Architecture	Random Embeddings		Wikipedia Embeddings		Reddit Embeddings		Wikipedia+Reddit Embeddings	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
Stack of two LSTMs	60%	60.0%	67%	68.8%	65%	66.3%	64%	65.7%
Bidirectional LSTM	62%	61.7%	65%	66.9%	65%	65.9%	66%	67.6%
CNN	65%	69.9%	66%	70.3%	67%	71.5%	66%	71.1%
CNN-LSTM	**66%**	**69.6%**	**67%**	**69.5%**	**68%**	**70.5%**	**69%**	**70.8%**

Table 5.15: The results from the experiments on the Reddit dataset using different embeddings.

Architecture	Random Embeddings		NER Twitter Embeddings		Amir Twitter Embeddings		NER+Amir Twitter Embeddings	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
Stack of two LSTMs	70%	68.9%	77%	78.1%	79%	80.4%	78%	78.6%
Bidirectional LSTM	72%	71.6%	79%	79.7%	78%	79.8%	78%	79.2%
CNN	*	*	77%	78.2%	**80%**	**81.7%**	**81%**	**82.8%**
CNN-LSTM	**76%**	**79.9%**	**80%**	**82.0%**	80%	81.6%	81%	82.7%

Table 5.16: The results from the experiments on the Riloff Twitter dataset using different embeddings.

Architecture	Random Embeddings		NER Twitter Embeddings		Amir Twitter Embeddings		NER+Amir Twitter Embeddings	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
Stack of two LSTMs	**67%**	**67.4%**	69%	68.8%	67%	67.0%	69%	69.4%
Bidirectional LSTM	59%	55.4%	68%	68.1%	67%	66.9%	69%	69.1%
CNN	53%	59.0%	67%	67.5%	69%	68.7%	70%	69.9%
CNN-LSTM	64%	64.4%	**70%**	**70.5%**	**70%**	**69.8%**	**71%**	**71.3%**

Table 5.17: The results from the experiments on the Bamman Twitter dataset using different embeddings.

Architecture	Random Embeddings		*Público* Embeddings	
	F1-score	Accuracy	F1-score	Accuracy
Stack of two LSTMs	87%	87.2%	89%	89.9%
Bidirectional LSTM	**87%**	**87.3**%	90%	90.2%
CNN	50%	50.1%	90%	90.4%
CNN-LSTM	56%	55.7%	**91%**	**90.7%**

Table 5.18: The results from the experiments on the news headline dataset using two different embeddings.

Dataset	Test Set	Root Mean Squared Error		Pearson Correlation	
		Valence	Arousal	Valence	Arousal
Warriner	Facebook	1.0694	2.1218	0.2233	0.0897
	EmoTales	1.3517	1.6058	0.1110	0.0176
	ANET	2.6399	2.4634	0.1738	0.5054
	All	1.2322	1.9840	0.1714	0.1039
Warriner + Paraphrases	Facebook	1,0534	2,0230	0.1724	-0.0258
	EmoTales	1,3025	1,8232	0.0201	0.0164
	ANET	2,6448	3,0203	0.0931	-0.0276
	All	1,2064	1,9966	0.1272	-0.0157

Table 5.19: The results from the tests performed on the affective norms considering paraphrases and using only an affective norm architecture.

Dataset	Test Set	Root Mean Squared Error		Pearson Correlation	
		Valence	Arousal	Valence	Arousal
Warriner	Facebook	1,8257	2,2884	0.2794	0.1592
	EmoTales	2,0909	1,4479	0.1279	0.0902
	ANET	2,9790	2,1590	0.5615	0.4506
	All	1,9526	2,0561	0.2349	0.0875
Warriner + Paraphrases	Facebook	1,0349	2,0721	0.4473	0.0791
	EmoTales	1,7539	1,3648	0.1924	0.0313
	ANET	2,4324	2,8937	0.5555	0.3445
	All	1,2055	2,0049	0.3704	0.0604

Table 5.20: The results from the tests performed on the affective norms considering paraphrases using an architecture leveraging the affective norms and Googles' pre-trained word embeding.

Over the Riloff dataset (see Table 5.16), the results using the two pre-trained embeddings individually were not much different between each other. However, combining the two yielded a slightly better performance, i.e., an improvement of 1 p.p. over the F1-score (with a p-value of 0.004 on the McNemar test), over both the other neural networks leveraging from a single word embedding and the best performing feature-based machine learning classifier.

Table 5.17 shows the results obtained over the Bamman Twitter dataset. In this case, the CNN-LSTM performed better on all tests except when using randomly initialized embeddings. Even though it was not possible to obtain the same results as on the study of Amir *et al.* (2016), it is possible to compare the results from using word embeddings individually and combined. When combining the two word embeddings, the model performed better by 1 p.p. over the F1-score (with a p-value lower than 0.001 on the McNemar Test).

Finally, on Table 5.18, you can see the last tests using the Portuguese news headlines. Notice that using pre-trained embeddings improved the results by 6% over the F1-score, when compared to the logistic regression classifier.

As mentioned on Section 5.2, to test the usage of affective norms paraphrasing and prediction results two experiments were performed. The results from the first experiment, involving only

Architecture	Reddit		Twitter Riloff		Twitter Bamman		News Headlines	
	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy
CNN-LSTM	66%	67.1%	80%	81.0%	68%	67.9%	-	-

Table 5.21: The results from the experiments using an affective norm representation for irony detection, with the CNN-LSTM architecture.

(a) **(b)**

Figure 5.11: Results of the (a) Logistic Regression and (b) SVM classifier, with and without the set of extra features.

the use of the affective norms by themselves, can be seen on Table 5.19. The results from the second experiment, using an architecture leveraging both from the affective norms and a Google pre-trained embeddings, can be seen on Table 5.20. The test sets present on these tables are denoted as following: the dataset from the work of Preoţiuc-Pietro *et al.* (2016) is denoted simply as *Facebook*. The dataset from the work of Francisco *et al.* (2012) is denoted as *EmoTales*. Finally, the dataset from the work of Bradley & Lang (2007) is denoted as *ANET*. The test set denoted as *All* combines all the previous datasets in a single test set.

Observe that even though the architecture using both the affective norms and the Google word embeddings yielded a error of 2.4 for the valence rating and 2.9 for the arousal rating, it also yielded a better correlation, yielding 0.55 and 0.34, respectively.

After finishing the experiments with modeling of the affective norms, the network that performed best (i.e., the CNN-LSTM) on the task of irony detection was used to test the use of the affective norms as representations to be used on the classification procedure. The results from this experiment, which can be seen on Table 5.20 determined that there was no gain on using a word embedding with the affective norms.

5.4 Discussion

The main motivation behind the work reported on this dissertation was to make a comparison between the performance of feature-based machine learning algorithms, like logistic regression

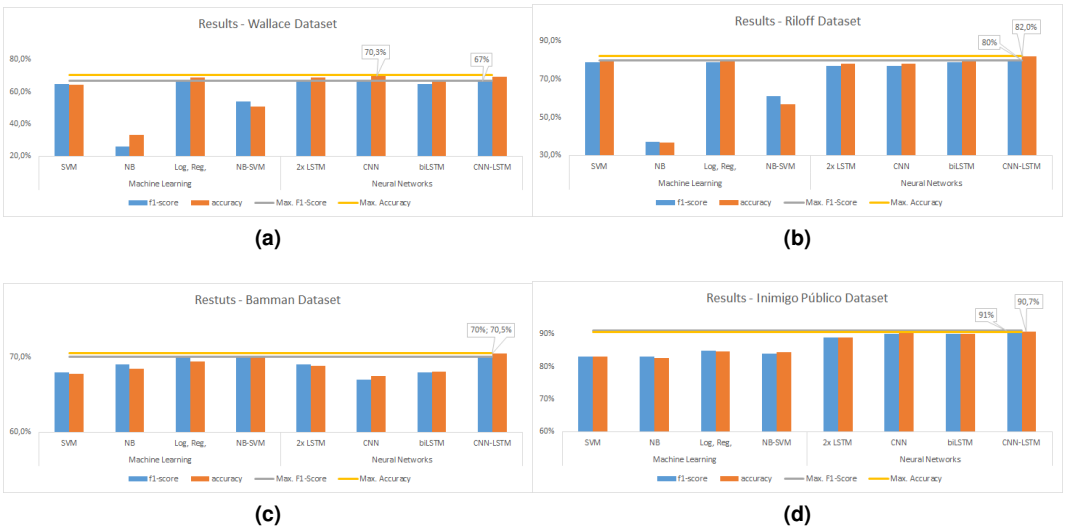


Figure 5.12: Results of the neural networks on the (a) Reddit, (b) Riloff and (c) Bamman, and (d) Portuguese datasets.

models and SVMs, and the deep learning algorithms that have been appearing on tasks involving semantic interpretation of texts. To be able to make a broader comparison, datasets from slightly different domains were used, one from a social media discussion forum named Reddit, two datasets from the social media networking service named Twitter, and a dataset consisting of satirical and factual news headlines.

The first point to be made is that in terms of the classification approaches using feature-based machine learning classifiers, logistic regression models consistently yielded the best results, while SVMs yielded the second best results. However, when adding more lexical features, the logistic regression showed little or no improvement, and observing the results from the SVM, the inclusion of those same features allowed the SVM to perform much better, making it able to compete with the performance of the logistic regression classifier. This shows that when handling a

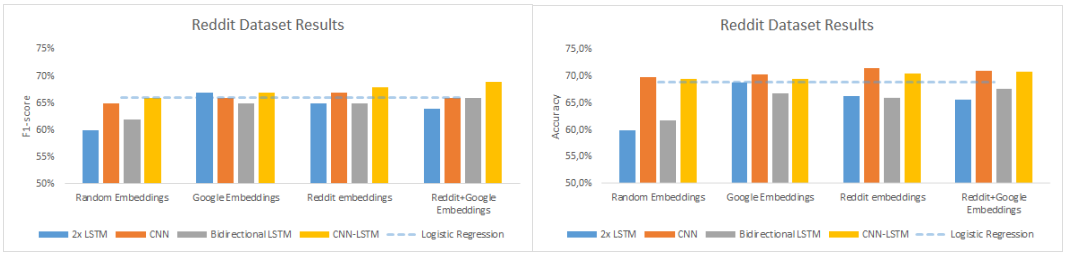
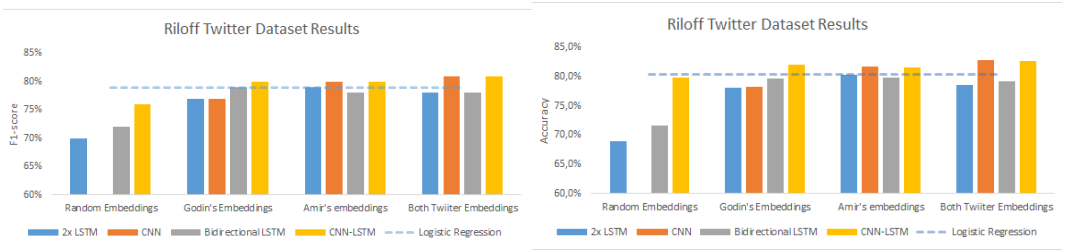
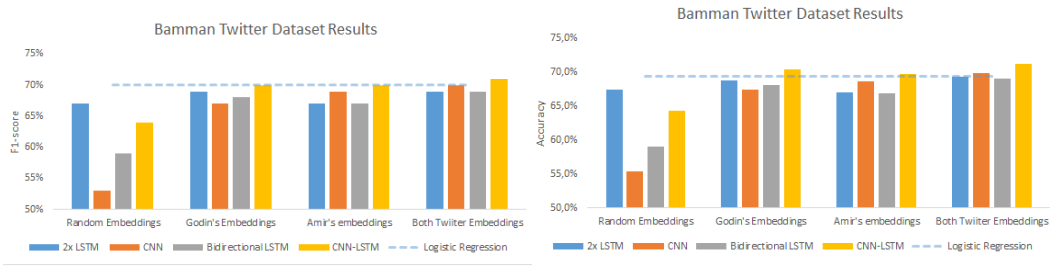


Figure 5.13: Results of the neural networks leveraging different embeddings, comparing to the logistic regression classifier, over the Reddit dataset.



(a) (b)

Figure 5.14: Results of the neural networks leveraging different embeddings, comparing to the logistic regression classifier, over the Twitter Riloff dataset.



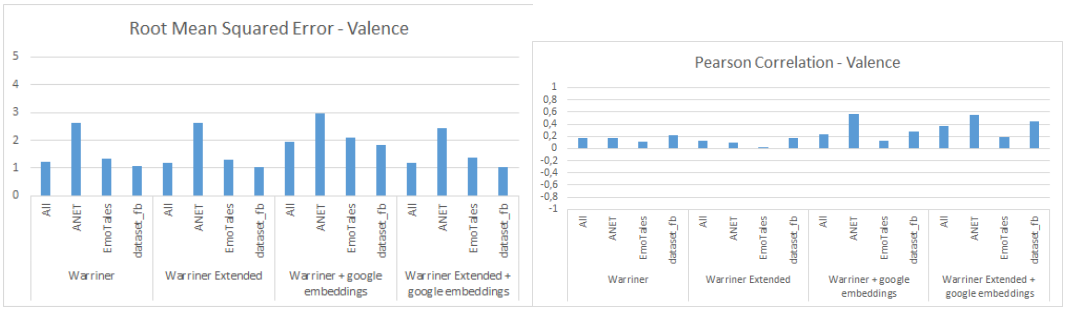
(a) (b)

Figure 5.15: Results of the neural networks leveraging different embeddings, over the Bamman Twitter dataset.

large set of features, as most of the previous studies addressed on Chapter 3, an SVM classifier will be more likely to outperform the other feature-based classifiers on this particular task.

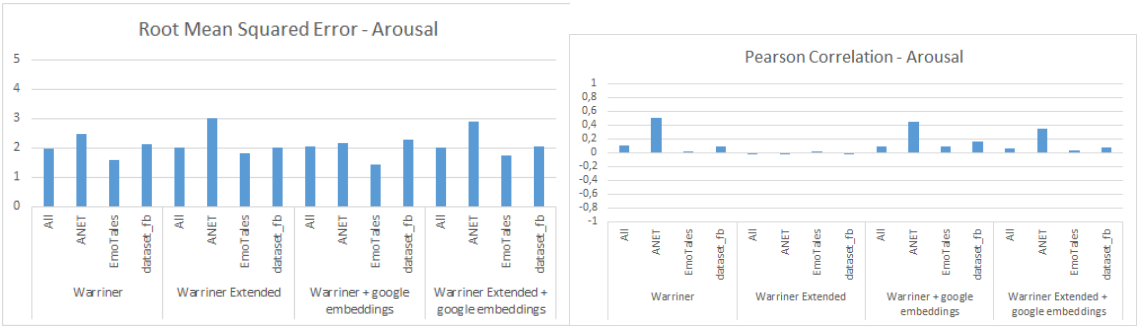
The second point that can be made, this time from the results of the second set of experiments, is that neural networks are able to perform as well as any of the classic machine learning algorithms, as can be seen in Figure 5.12. Even though these neural networks only used word embeddings that were not created specifically for the particular task of irony detection or even for the domain of the dataset being used on this work, these models are able to, outperform the best feature-based classifier used on the first set of tests.

The main feature that neural networks require to perform decently on any text-based classification task are appropriate embeddings. The better the word representations from the embeddings, the better the performance of the classifier. A simple word embedding matrix can be created using word2vec (Mikolov *et al.*, 2013b) on a large dataset. With some tuning of the available parameters (e.g., context window and number of dimensions of the embeddings), it is possible to improve the performance of deep learning classifiers (Figure 5.13). Even though these embeddings can take some time to create and optimize in order to obtain good results, with the tools that are publicly available today, this does not require much of an implementation effort which is one of the major



(a) (b)

Figure 5.16: Results of each of the test sets using a neural network that predicts the value of the valence emotional norm.



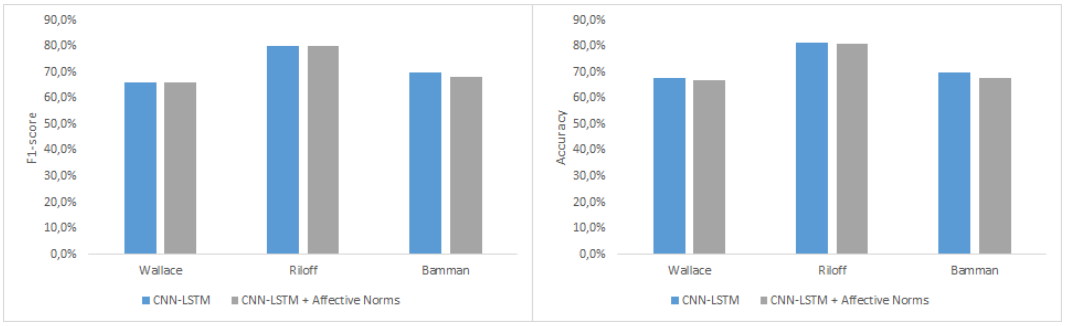
(a) (b)

Figure 5.17: Results of each of the test sets using a neural network that predicts the value of the arousal emotional norm.

issues when engineering features for the classic machine learning classifiers.

With this in mind, the third set of experiments consisted of verifying the impact of using different word embeddings on neural networks. Using random embeddings with the neural networks performed quite poorly. However, this was to be expected since we are dealing with datasets that are too small for the neural networks to adjust the weights well enough to obtain a good representation. Using individual pre-trained word embeddings, it was possible to outperform the feature-based machine learning classifiers, specifically using the CNN-LSTM architecture. As anticipated, with the word embeddings that are more specific to the domain, the classifiers performed better than with word embeddings that were created for another domain or task (Figure 5.13).

Combining multiple word embeddings is another way to improve the results of the neural networks, as can be observed from the results of the experiments performed on this dissertation. Even though the embeddings used in these experiments only considered the words from their



(a) **(b)**

Figure 5.18: Results of the CNN-LSTM architecture with and without leveraging the affective norms of valence and arousal.

datasets, they still complemented each other and consequently improved the results from the classifiers.

Another approach attempted to use embeddings based on affective norms, but the results from these experiments did not show any improvement in the classification task. The first problem observed from this approach was the lack of information for many of the words in the affective norms dataset made available from the study of Warriner *et al.* (2013). Although it was possible to increase the number of words of the dataset using paraphrases, this did not alter the outcome of the classification.

Chapter 6

Conclusions ans Future Work

This chapter contains a brief summary of the key findings of my MSc research project, presenting the conclusions that are possible to make from the different experiments that were performed, and discussing the importance of the findings derived from these results.

The chapter also describes possible future work based on the outcomes of the experiments and observations made in this study.

6.1 Conclusions

This dissertation described three set of experiments involving neural networks, with the objective of verifying the performance of this approach on the task of irony detection. It was expected that neural networks would outperform the previously used feature-based machine learning classifiers, as it has been happening on other NLP tasks of NLP.

The first conclusion that can be made from the results of the experiments is that, from the different machine learning algorithms, both feature-based (i.e., naïve Bayes, logistic regression, SVM and naïve Bayes-SVM) and neural network approaches (i.e., two stacks of LSTM layers, bidirectional LSTM layers, a CNN, and a CNN-LSTM), the CNN-LSTM is the best classifier on the task of irony detection over datasets with different domains, outperforming any of the other architectures on all the different datasets, including the feature-based classifiers that are most commonly used on the task at hand.

From the second set of experiments using different word embeddings, it was possible to conclude that even though, as expected, word embeddings more specific to the domain of the dataset achieve better results, it is possible to improve the performance of the neural networks by combining different word embeddings, even if not created using a dataset from the same domain as

the one of the datasets being classified.

The third experiment, related to using affective norms to detect irony, showed that using a 2-dimensional vector space representation of affective norms is insufficient to allow for an improvement on the performance on neural networks.

The findings of this research study show that there are two viable approaches that correspond to different paradigms for the task of detecting irony in text. The first approach has been commonly used over the last years and is based on the design of features, where each feature requires a lot of manual labor. The second approach, based on neural networks, is a more recent approach that leverages on word embeddings to classify documents.

Neural networks have the advantage of not requiring the same amount of manual labor to obtain the same results. However, these models do require the use of other resources, they need time (needed to train a neural network) and also better hardware (to be able to train the neural networks over shorter training times). It is also worth noting that it is difficult to predict the outcome of a neural network since it can involve the use of any combination of layers, embeddings and hyperparameters. Nonetheless, feature-based machine learning algorithms, that have been used on previous studies, have been requiring larger and more complex features to obtain better results on the classification of text according to irony, consequently requiring more design and implementation time, to only slightly improve the results.

Using neural networks seems to require less time to implement and design than methods based on features that would still be outperformed by a simple neural network leveraging pre-trained word embeddings that can be found publicly available. This makes deep learning an interesting new approach, although there are still improvements that can be made for the task of irony detection.

6.2 Future Work

The use of neural networks on the task of irony detection is very recent. There are many ideas that can be used to improve the performance of deep neural classifiers on this task, specifically on what regards the creation of word representations (i.e., embeddings).

A possibility to give continuity to this research project is to use other algorithms to create embeddings besides word2vec, such as the algorithm from the work of Pennington *et al.* (2014), and combine the resulting embeddings into a single architecture (Zhang *et al.*, 2016).

Another possible work that can be performed is the creation of embeddings specific to a user (i.e., user embeddings) instead of words. Information of the user has been shown to improve the

performance of classifiers over the task of irony detection. Because of this, a user embeddings would follow the logic of Bamman & Smith (2015) where the features took into consideration the users historical data and his audience. This can also be done to create embeddings specific to a user, as it has been shown on the work of Amir *et al.* (2016).

Bibliography

AMIR, S., WALLACE, B.C., LYU, H., CARVALHO, P. & SILVA, M.J. (2016). Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of the Computational Natural Language Learning*.

BAMMAN, D. & SMITH, N.A. (2015). Contextualized sarcasm detection on Twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*.

BRADLEY, M.M. & LANG, P.J. (2007). Affective norms for English Text (ANET): Affective ratings of text and instruction manual. Tech. rep.

BURFOOT, C. & BALDWIN, T. (2009). Automatic satire detection: Are you having a laugh? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and of the International Joint Conference on Natural Language Processing*.

CARVALHO, P., SARMENTO, L., SILVA, M.J. & DE OLIVEIRA, E. (2009). Clues for detecting irony in user-generated contents: Oh...!! it's "so easy" ;-). In *Proceedings of the International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*.

CORTES, C. & VAPNIK, V. (1995). Support-vector networks. *Machine Learning*, **20**.

DANFORTH, C. & DODDS, P.S. (2010). Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. *Journal of Happiness Studies*, **11**.

DAVIDOV, D., TSUR, O. & RAPPOPORT, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the International Conference on Computational Linguistics*.

FAGERLAND, M.W., LYDERSEN, S. & LAAKE, P. (2013). The mcnemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional. *BioMed Central Medical Research Methodology*, **13**.

FRANCISCO, V., HERVÁS, R., PEINADO, F. & GERVÁS, P. (2012). Emotales: creating a corpus of folk tales with emotional annotations. *Language Resources and Evaluation*, **46**.

GANITKEVITCH, J., VAN DURME, B. & CALLISON-BURCH, C. (2013). PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

GHOSH, A. & VEALE, T. (2016). Fracking sarcasm using neural network. In *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

GHOSH, D., GUO, W. & MURESAN, S. (2015). Sarcastic or not - word embeddings to predict the literal or sarcastic meaning of words. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

GODIN, T., SUTSKEVER, I., CHEN, K., CORRADO, G.S. & DEAN, J. (2013). Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations.

GONZÁLEZ-IBÁÑEZ, R., MURESAN, S. & WACHOLDER, N. (2011). Identifying sarcasm in twitter: A closer look. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

HOCHREITER, S. & SCHMIDHUBER, J. (1997). Long short-term memory. *Neural Computation*, **9**.

IPEIROTIS, P.G. (2010). Analyzing the amazon mechanical turk marketplace. *Association for Computing Machinery Crossroads*, **17**.

JOSHI, A., SHARMA, V. & BHATTACHARYYA, P. (2015). Harnessing context incongruity for sarcasm detection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and of the International Joint Conference on Natural Language Processing*.

KAROUI, J., FARAH, B., MORICEAU, V., AUSSENAC-GILLES, N. & HADRICH-BELGUITH, L. (2015). Towards a contextual pragmatic model to detect irony in tweets. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and of the International Joint Conference on Natural Language Processing*.

KHATTRI, A., JOSHI, A., BHATTACHARYYA, P. & CARMAN, M.J. (2015). Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

KIM, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

MIKOLOV, F., VANDERSMISSEN, B., DE NEVE, W., WALLE, R.V.D. & DEAN, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*.

MIKOLOV, T., CHEN, K., CORRADO, G. & DEAN, J. (2013b). Efficient estimation of word representations in vector space. *Computing Research Repository*, **abs/1301.3781**.

NAIR, V. & HINTON, G.E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*.

PANG, B. & LEE, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, **2**.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M. & DUCHESNAY, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**.

PENNINGTON, J., SOCHER, R. & MANNING, C.D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

PREOŢIUC-PIETRO, D., SCHWARTZ, H.A., PARK, G., EICHSTAEDT, J., KERN, M., UNGAR, L. & SHULMAN, E. (2016). Modelling valence and arousal in facebook posts. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

QUINLAN, J.R. (1986). Induction of decision trees. *Machine Learning*, **1**.

RAJADESINGAN, A., ZAFARANI, R. & LIU, H. (2015). Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

REYES, A., ROSSO, P. & VEALE, T. (2013). A multidimensional approach for detecting irony in twitter. *Language Resources and Evaluation*, **47**.

RILOFF, E., QADIR, A., SURVE, P., SILVA, L.D., GILBERT, N. & HUANG, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*.

RISH, I. (2001). An empirical study of the naive bayes classifier. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Empirical Methods in Artificial Intelligence*.

RUMELHART, D.E., HINTON, G.E. & WILLIAMS, R.J. (1988). Neurocomputing: Foundations of research. chap. Learning Representations by Back-propagating Errors.

SEVERYN, A. & MOSCHITTI, A. (2015a). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

SEVERYN, A. & MOSCHITTI, A. (2015b). Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the International Workshop on Semantic Evaluation*.

STRAPPARAVA, C. & VALITUTTI, A. (2004). WordNet-Affect: An affective extension of WordNet. In *Proceedings of the International Conference on Language Resources and Evaluation*.

TEPPERMAN, J., TRAUM, D.R. & NARAYANAN, S. (2006). "yeah right": sarcasm recognition for spoken dialogue systems. In *Proceedings of Interspeech*.

THELWALL, M., BUCKLEY, K. & PALTOGLOU, G. (2012). Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, **63**.

WALLACE, B.C. (2015). Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, **43**.

WALLACE, B.C., CHOE, D.K., KERTZ, L. & CHARNIAK, E. (2014). Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

WALLACE, C.B., CHOE, K.D. & CHARNIAK, E. (2015). Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

WANG, S. & MANNING, C.D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

WARRINER, A.B., KUPERMAN, V. & BRYSBAERT, M. (2013). Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, **45**.

YU, H.F., HUANG, F.L. & LIN, C.J. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, **85**.

ZHANG, Y., ROLLER, S. & WALLACE, B.C. (2016). Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

ZHOU, C., SUN, C., LIU, Z. & LAU, F.C.M. (2015). A c-lstm neural network for text classification. *Computing Research Repository*, **abs**/**1511.08630**.