An Exploration of Sarcasm Detection Using Deep Learning

BY

EDOARDO SAVINI
B.S., Computer Engineering, Politecnico di Torino, Torino, Italy, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Cornelia Caragea, Chair and Advisor
Erdem Koyuncu
Elena Maria Baralis, Politecnico di Torino

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Prof. Cornelia Caragea, for her constant support, help and patience throughout the time I spent at UIC. She stimulated my work with her ideas and motivation and made an important contribution to my academic and personal growth.

I would also like to express my gratitude to the other members of my thesis committee, Prof. Elena Baralis and Prof. Erdem Koyuncu, for their interest and their feedback and advices on this work.

A very special thanks to Lynn Thomas and Jenna Stephens for their prompt assistance to solve any problems that could affect an international student at UIC.

I want to thank my family for supporting me and giving me the chance to live this amazing experience.

I thank my friend Alessandro for sharing this whole journey with me till the end, and also the amazing Rafiki's Squad for giving me the best times I had in this permanence in the US: thanks to my party buddy Gabriele, the bomber Arturo and also to that funny useless panda Davide.

A special thanks to all the people that made these last two years unforgettable, especially to Alessandra, Marco and Valerio who, despite the distance, never stopped supporting me and cheering me up.

ES

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
2	**RELATED WORK**		4
	2.1	Content Feature-Based Analyses	4
	2.2	Context-based models	5
	2.3	Deep Learning for Sarcasm Detection	6
	2.4	Multitasking	7
3	**NEURAL NETWORK MODELS**		9
	3.1	LSTM	9
	3.2	BiLSTM	10
	3.3	BiLSTM+Attention	11
	3.4	CNN	12
	3.5	CNN+LSTM	14
4	**WORD EMBEDDINGS**		15
	4.1	GloVe	15
	4.2	FastText	16
	4.3	ELMo	16
	4.4	Concatenation of ELMo with non-contextual embedding	17
5	**DATA**		18
	5.1	Sarcasm V2 Corpus	18
	5.2	SARC	19
	5.3	Crawled Dataset	20
6	**EXPERIMENTS**		23
	6.1	Base Model	23
	6.2	Implementation	25
	6.3	Experiments on Sarcasm V2 Corpus	26
	6.3.1	Experiments on Validation Set	26
	6.3.1.1	LSTM	26
	6.3.1.2	BiLSTM	28
	6.3.1.3	CNN	29
	6.3.1.4	Analysis on the Validation Set	30
	6.3.2	Evaluation on the Test set	30
	6.4	SARC analysis	31

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

6.4.1 SARC Training 31
6.5 Multi-Tasking Approach 32
6.5.1 Sentiment Analysis 34
6.5.2 Finding the most performing auxiliary task 35
6.5.3 Multitask Training phase 36
6.6 Experiments on the Crawled Dataset 37

7 RESULTS 39
7.1 Results on Sarcasm Corpus V2 39
7.2 Results on Sarcasm Corpus V2 with Multitasking 42
7.3 Results on SARC 44
7.4 Results on SARC with MultiTasking 46
7.5 Comparison with Baseline methods on SARC 47
7.6 Best Model Prediction 49
7.6.1 Comparisons with Reality 52

8 CONCLUSION AND FUTURE WORK 54

CITED LITERATURE 56

LIST OF TABLES

TABLE		PAGE
I	TOTAL NUMBER OF TWEETS CRAWLED	21
II	CONTENT OF THE FINAL CRAWLED DATASET FOR PREDICTIONS	22
III	EXPERIMENT ON VALIDATION SET WITH LSTM ENCODER	27
IV	EXPERIMENT ON VALIDATION SET WITH BILSTM ENCODER	28
V	EXPERIMENT ON VALIDATION SET WITH CNN ENCODER	29
VI	RESULTS ON SARCASM V2 CORPUS (WITHOUT MULTITASKING)	41
VII	RESULTS USING MULTITASKING ON SARCASM V2 CORPUS	43
VIII	RESULTS ON A SMALLER TRAINING SET OF SARC (WITHOUT MULTITASKING)	45
IX	RESULTS USING MULTITASKING ON A SMALLER TRAINING SET OF SARC	46
X	COMPARISON WITH THE BASELINES OF MAIN BALANCED SARC DATASET	49
XI	SARC BEST MODEL PREDICTIONS WITH 0.5 THRESHOLD	50
XII	SARC BEST MODEL PREDICTIONS WITH 0.6 THRESHOLD	51

LIST OF FIGURES

FIGURE		PAGE
1	Configuration of an LSTM cell.	9
2	Example of a Bidirectional LSTM network	11
3	Example of a Convolutional Neural Network for image processing .	13
4	Base Model for the experiments.	23
5	Two Models configuration.	24
6	Multi-tasking framework.	33

LIST OF ABBREVIATIONS

BiLSTM	Bidirectional Long-Short Memory
CNN	Convolutional Neural Network
ELMo	Embeddings from Language Models
GloVe	Global Vector for word representations
IAC	Internet Argument Corpus
LSTM	Long-Short Memory
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NN	Neural Network
SARC	Self-Annotated Reddit Corpus
SVM	Support Vector Machine

SUMMARY

Sarcasm detection plays an important role in Natural Language Processing as it has been considered one of the most challenging task in sentiment analysis and opinion mining applications. Our work aims to recognize sarcasm in social media sites, microblogs and discussion forums, exploiting the potential of Deep Learning tools such as Deep Neural Network and Word Embeddings. In this thesis, we (a) develop multiple types of neural models and analyze their efficiency when combined with word embeddings; (b) create a new multitasking framework that exploits the strong correlation between sarcasm and sentiment detection (c) test the performances of our models on two pre-labelled datasets; (d) compare our results with other state-of-the-art models; (e) apply our models on real word data to evaluate the efficiency of their prediction. We then discuss on the benefits of our research in the field of sarcasm detection and sentiment analysis, and put the basis for some future researches.

CHAPTER 1

INTRODUCTION

In recent years, Internet has become the main source to communicate and share information. In particular, social media sites, microblogs, discussion forums, online reviews have become more and more popular. They represent a way for people to express their own opinion with no inhibition, and to look for some advice. Many companies take advantage of these sites' popularity to share their products and services, provide assistance and understand costumer needs.

For this reason, social media have developed into one of the the main domains for Natural Language Processing (NLP) research, especially in the area of Sentiment Analysis. Analyzing people sentiment could be useful to comprehend their behaviour, monitor customer satisfaction and increase sales revenue. However, this opinion mining task appears to be very challenging, due to the dense presence of figurative languages in social communities like Reddit[1] or Twitter[2].

Our research focuses on a recurrent sophisticated linguistic phenomenon (and a form of speech act) that makes use of figurative images to implicitly convey contempt through incongruity [20] between text and context, the sarcasm. Its highly figurative nature has caused

---

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

[2] `https://twitter.com/`

sarcasm to be defined as one of the main challenges in sentiment analysis and opinion mining applications.

While many previous works on this task have focused more on approaches based on feature engineering, that use distant supervision and Support Vector Machines to extract lexical cues recurrent in sarcasm, we propose to continue the path followed by Amir et al. (2016), Ghosh and Veale (2016) and Joshi et al. (2016) [1; 13; 21] in attempting to automatically detect sarcasm harnessing the potentials of deep neural networks combined with word embeddings to capture both semantic and syntactic features in sarcastic utterances.

The first phase of our work consists on experimenting with five basic neural network models (BiLSTM, LSTM, CNN, BiLSTM+Attention, CNN-LSTM) combined with different typologies of pre-trained word embeddings (GloVe, FastText, ELMo) and their concatenations, to find the best configuration to predict sarcasm.

We study the performances of our basic models on two datasets of different sizes, collected from Internet Argument Corpus (IAC)[3] and Reddit. Our purpose is to analyze the efficiency of Deep Learning elements in the sarcasm detection task and find a neural framework able to accurately predict sarcasm in many types of social platforms, from discussion forums to microblogs, without recurring to manual feature engineering or user embeddings (context-based features). We also attempt to develop a multitasking framework for sarcasm detection, joining an auxiliary sentiment detection task to our main model.

---

[3] `https://nlds.soe.ucsc.edu/iac2`

In the last part of our research we apply our best models on real-word data. We create our own data collections with tweets coming from the cities of Chicago, San Francisco and Philadelphia, concerning eight different topics in which sarcasm is recurrent (Abortion, Creation, Health, Homophobia, Obama, Racism, Terrorism, Trump). We analyze the percentage of sarcastic tweets to evaluate our model and make sociological analyses on how geographical circumstances, such as climate and political tendencies, could affect people attitude towards certain arguments.

CHAPTER 2

RELATED WORK

Many studies and researches have been conducted in the past years having sarcasm as subject, not only in linguistic field, but also in psychology and cognitive science [32; 15; 25; 43; 47; 16]. However, experiments on automatic sarcasm detection represent a recent field of study.

2.1 Content Feature-Based Analyses

One of the first analyses has been made by Tepperman et al. (2006) [45] who aimed to recognize sarcasm in speech using prosodic, spectral, and contextual cues. Their research attracted the interest of many sentiment analysis experts.

The first investigations made on text were focused on discovering lexical indicators and syntactic cues that could be used as features for sarcasm detection. In fact, at the beginning, sarcasm recognition was considered as a simple text classification task.

Kreuz and Caucci (2007) [24] noted that interjections, punctuation symbols, intensifiers and hyperboles play a fundamental role for the research. Also Carvalho et al. (2009) [7] found that oral and gestural expressions such as emoticons and other keyboard characters are more predictive of sarcasm.

Tsur et al. (2010) [46] found that exclamations like *"yay!"* or *"great!"* are good indicators on Amazon product reviews, and Davidov et al. (2010) [10] exploited syntactic patterns such as sarcasm hashtags to train classifiers using a semi-supervised technique.

González-Ibánez et al. (2011) [17] performed Support Vector Machines and logistic regression on tweets using also emoticons and sarcastic hashtags as features. They also found positive and negative emotions to be strongly correlated with sarcasm. This theory was deepened by Riloff et al. (2013) [40] which developed a bootstrapping algorithm based on the opinion that sarcasm consists on a contrast between Positive Sentiment and a Negative Situation (e.g. *"I love being ignored"*). Their work was resumed by Joshi et al. (2015) [20] who used a similar method to predict implicit and explicit incongruity features.

While the previous works focused on unigrams, Lukin and Walker (2017) [28] proposed a bootstrapping method too, based on the expansion of sarcastic N-grams cues, such as *"no way"*, *"oh really"*, etc. N-grams were considered also by Liebrecht et al. (2013) [26] and Ptáček et al. (2014) [38] to create classifiers to analyze Dutch, Czech and English tweets.

2.2 Context-based models

However, sarcasm can not be considered only as a purely lexical and syntactic phenomenon. In fact, Wallace et al. (2014) [48] showed that many of the classifiers previously described fail when dealing with sentences where context is needed. For example, in political posts, a general knowledge of the political situation regarding the time and place in which a certain post was written is crucial to understand its meaning.

A more semantic approach was performed by Joshi et al. (2015) [20] who used a method similar to Riloff et al. (2013) [40] to predict implicit and explicit incongruity features. They expanded the context of a discussion forum post including the parent post (called *"elicitor"*) in the discussion thread.

Rajadesingan et al. (2015) [39] and Bamman and Smith (2015) [2] extracted contextual features (user profile information) from the history tweets of the same author. Also, Khattri et al. (2015) [22] tried to discover sarcasm looking for contrast between users and specific entities along their tweets history.

2.3 Deep Learning for Sarcasm Detection

In order to detect this kind of semantic information from a sarcastic statement, some researchers experimented also a new approach using Deep Learning techniques. The advantage of adopting neural networks is in their ability to induce features automatically, allowing them to capture long-range and subtle semantic characteristics that are hard to find with manual feature engineering.

Amir et al. (2016) [1] use Convolutional Neural Networks to capture user embeddings and utterance-based features. They managed to discover homophily scanning user's historical tweets. Also Joshi et al. (2016) [21] proposed different kinds of word embeddings (Word2Vec, GloVe, LSA), augmented with other features on word vector-based similarity, to apprehend context in phrases with no sentiment words. Poria et al. (2016) [37] developed a framework based on pre-trained CNN to retrieve sentiment, emotion and personality features for sarcasm recognition.

Zhang et al. (2016) [49] created a bi-directional gated recurrent neural network with a pooling mechanism to automatically detect content features from tweets and context information from history tweets. They demonstrated the efficiency of neural networks over manual feature engineering.

Ghosh and Veale (2016) [13] proposed a concatenation of Convolutional Neural Network, Long-Short Term Memory Network and Deep Neural Network (CNN-LSTM-DNN) that outperformed many state-of-art methods based on text features. We created a similar framework for one of our main network models (Section 3.5). They exploited the same framework one year later [14] to capture the mood of the users along time, in order to increase the accuracy of their framework.

Tay et al. (2018) [44] used a Multi-dimensional Intra-Attention Recurrent Network (MIARN) to detect word-to-word and long range dependencies.

Hazarika et al. (2018) [18] proposed a framework able to detect contextual information with user embedding created through user profiling and discourse modeling from comments on Reddit. Their model reaches the state-of-art in one of the datasets (SARC) we consider for the experiments. Nevertheless, our evaluation without user embedding information outperforms their basic model.

2.4 Multitasking

For as regards the multitask learning [6], our model relies on the work made by Cohan et al. (2019) [9] for citation intent classification. They showed that the performances of a main classification task can be improved by being biased from the training of some secondary tasks. We adapted their code in order to make it work for the sarcasm classification, having the sentiment analysis as secondary task.

We discovered that this mechanism was only applied recently on sarcasm detection task by Majumder et al. (2019) [29]. Following the same intuition we had regarding a correlation

between sentiment classification and sarcasm detection, they created a GRU-based classifier with attention mechanism and applied it on the dataset by Mishra et al. (2016) [31], which contained about a thousand samples of sentences with both sarcastic and sentiment tags. Their mechanism shares the GRU model between the two tasks and exploits a neural tensor network to fuse sarcasm and sentiment-specific word vectors. They outperform the state-of-the-art previously obtained by Mishra et al. (2016) [31] with a CNN-based model.

Other relevant works in NLP on this matter were made by Chen et al. (2017) [8] and Liu et al. (2017) [27]. The first obtained relevant results in the field of Chinese Word Segmentation. They use 3 different segmentation criteria (with a shared-private layer configuration) as single tasks, and integrate their knowledge in a multitasking framework to extract criteria-invariant (with the shared layer) and criteria-specific (with the private layer) features. The second ones use an adversarial multitask framework for text classification in which private and shared features are disjoint. They introduced orthogonality constraints to remove feature redundancy and avoid noise data.

CHAPTER 3

NEURAL NETWORK MODELS

In this chapter we describe briefly the main features of the networks we have employed for our experiments. The main models used for our evaluations are: LSTM, BiLSTM, CNN, CNN+LSTM, BiLSTM+Attention.

3.1 LSTM

The Long Short-Term Memory (LSTM) network is a variation of the "classic" Recurrent Neural Network (RNN), proposed by the German researchers Sepp Hochreiter and Juergen Schmidhuber [19] to reduce the vanishing gradient problem. Figure 1 displays an example of an LSTM cell.

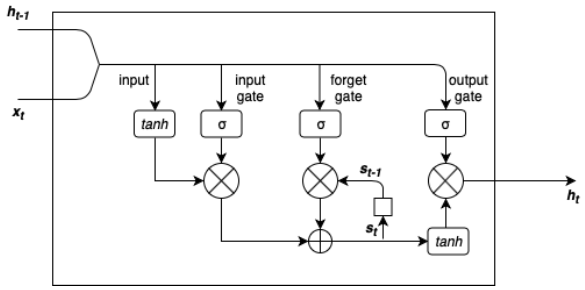

Figure 1. Configuration of an LSTM cell.

The main feature of this network is the presence of a loop, that makes it able to capture data information in its states and propagate it over the time. This characteristic has caused LSTM to be widely used in the last years to classify and process sequential data, especially in applications as time series prediction, audio and video composition, and speech processing.

For our experiment we use a single-layer LSTM encoder with a dropout rate of 0.2 and h cells on the hidden/output layer (usually $h = 100$). The model we use is a Seq2VecEncoder that transforms an input sequence of shape [*batch_size*, *sentence_length*, *embedding_dim*] into a vector in the form [*batch_size*, h].

3.2 BiLSTM

A Bidirectional LSTM is a network developed to retrieve more information from the same input data. As a simple LSTM scans an input only in one direction to predict new data capturing the old one, the bidirectional framework runs the input in two directions, forward and backward, allowing to store more information both from past and future.

The model has been really exploited recently especially for sentence prediction tasks because, thanks to its peculiarity, BiLSTM is able to detect more contextual information. This makes it a powerful tool for sarcasm detection.

For our experiment we model a single-layer BiLSTM Seq2VecEncoder with a dropout rate of 0.2 and h cells on the hidden layer (usually $h = 100$). As the hidden cells are scanned twice, the output data will have size equal to [*batch_size*, $2 * h$].

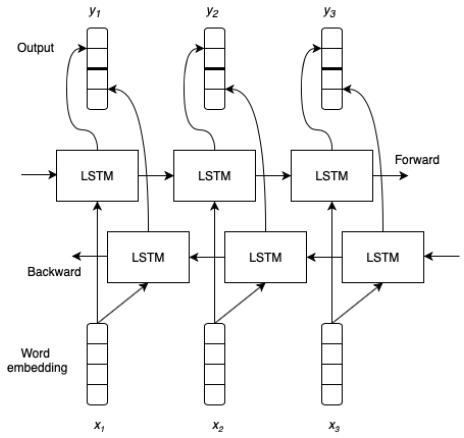

Figure 2. Example of a Bidirectional LSTM network

3.3 BiLSTM+Attention

Another approach we have implemented consists on taking every output of the hidden states of our BiLSTM and joining them through a simple attention mechanism [9] in order to obtain a single contextual vector representing the full sentence.

The BiLSTM encoder, in this instance, is a Seq2Seq encoder, which means that takes an input sequence of dimensions [*batch_size*, *sentence_length*, *embedding_dim*] and shapes it into an output of size [*batch_size*, *sentence_length*, $2 * h$]. This data is then transformed into a vector of shape [*batch_size*, $2 * h$] by our attention framework, which consists on a concatenation of dot-product and softmax operation.

In particular, given an input $d = \{d_1, d_2, ..., d_{2h}\}$ with $d_i \in \Re^{(sentence\_length, 2h)}$, the output vector y is computed as:

$$y = \sum_{i=1}^{sentence\_length} \gamma_i d_i \tag{3.1}$$

where:

$$\gamma_i = \text{softmax}(w^T d_i) \tag{3.2}$$

and w is a vector of parameters initialized with Xavier normalization, having zero mean and variance $Var(w_i) = 1/2h$, that embodies the query vector for dot-product attention.

3.4 CNN

A Convolutional Neural Network is a deep learning network developed for the image classification task but used also for text classification (sentence prediction) [50].

It consists on performing on an input data a series of convolutions and sub-sampling (pooling) operations, in order to analyze only relevant information (e.g. borders and shapes of an image) and simplify the initial data. This overcomes the problem of over-fitting data that could affect a multi-layer perceptron (MLP) network.

As it can be seen in the Figure 3, the CNN may exploit a fully connected layer only after the execution of the other operations, when the data has become more simple and informative.

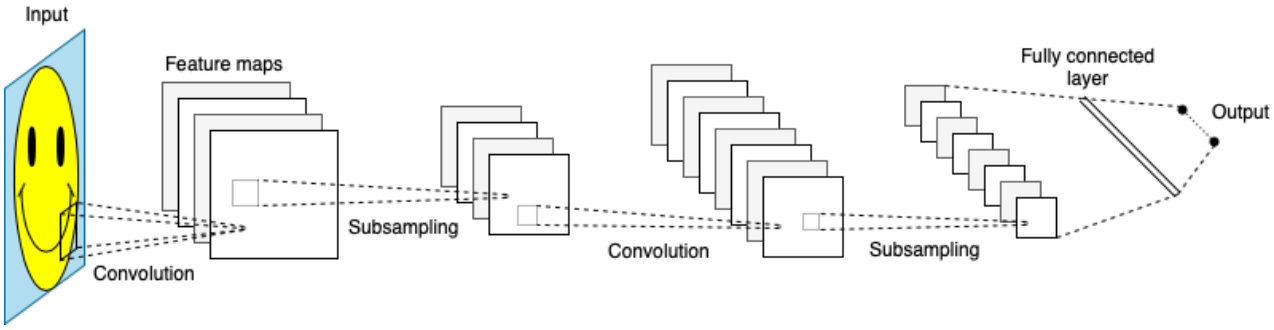

Figure 3. Example of a Convolutional Neural Network for image processing

For our experiment we use a Seq2Vec CNN encoder with a single combination of one convolution layer, with 150 filters of size 5, and one max-pooling layer, with kernel size and stride equal to 2, that aggregates the results of each convolution, outputs their max value and concatenates them into a single vector of size 150. Before the max-pooling operation, a ReLu activation function is applied on the results of each convolution.

Given the configuration of our model, its output should be of size:

$$out = N\_convolutions * N\_filters. \tag{3.3}$$

In this way, changing the number of filters may vary the output size. To address this problem, we add a projection layer that projects the collected features into a vector of fixed dimensions.

3.5 CNN+LSTM

We also build a model similar to the one developed by Ghosh and Veale (2016) [13], in which the output of the CNN is passed as input to an LSTM layer. For our implementation we follow the directions described by Brownlee (2017) [5].

In this instance, the CNN encoder takes an input of shape [*batch_size*, *sentence_length*, *embedding_dim*, 1] and creates an output of shape [*batch_size*, *sentence_length*/2, *N_filters*], that feeds our Seq2Vec LSTM network, with the same features as the one described previously.

CHAPTER 4

WORD EMBEDDINGS

As context plays a relevant role in sarcasm detection, the choice of the right word embedding becomes one of the most important tasks in our experiment. To obtain a detailed view of their performance and analyze their differences, we execute a lot of experiments with different typologies of pre-trained word embeddings and their concatenations, applied to the networks described in the previous chapter.

4.1 GloVe

GloVe (Global Vectors for word representations) is an unsupervised learning algorithm developed by Pennington et al. (2014) [35] that generates vector representations of words from their co-occurence information. This implies that the vectors contain information on how frequently words appear together in corpora of big size.

GloVe is a count-based model, which means that the vectors are built performing a dimensionality reduction from a co-occurence counts matrix. The matrix has words as rows and some contexts as columns and, for every word, it counts how often the word appears in some context along the corpus. For our experiment we consider 50, 100 and 300-dimensional word vectors trained on a collection of 6B tokens from Wikipedia[1] and Gigaword[2].

---

[1] `https://www.wikipedia.org/`

[2] `https://catalog.ldc.upenn.edu/LDC2003T05`

4.2 FastText

FastText [3] is an open-source, free, lightweight library that allows users to learn word embeddings and text classifiers, created by Facebook's AI Research (FAIR) lab. It is different from the previous embeddings as it treats each word as composed of n-grams. Each vector of a word is composed by the sum of its character n-grams.

The model represents an extension of the classic Word2Vec, faster in training with large corpora, able to compute word representations also for rare words and for words that do not appear in the training data ("out-of-vocabulary" words). It also supports 157 different languages. In our research we use 300-dimensional word vectors pre-trained on Common Crawl[3] (600B tokens).

4.3 ELMo

ELMo (Embeddings from Language Models) [36] is a deep contextualized word representation developed by the Allen Institute of Artificial Intelligence. It differs from the other models because it represents each word vector as a function of the sentence to which it belongs. Indeed, they are computed from the hidden states of a 2-layers bidirectional Language Model. This approach consistently increases the contextual information in the embeddings, making it appropriate for tasks such as sarcasm prediction and sentiment analysis.

---

[3] `http://commoncrawl.org`

We generally employ vectors trained on the 1 Billion Word Benchmark[4] (approximately 800M tokens of news crawl data from WMT 2011[5]) having output size of 1024. For the last part of our experiments we also tested the model trained on a dataset of 5.5B tokens (from Wikipedia (1.9B) and monolingual news crawl data from WMT 2008-2012 (3.6B)) as it turned out to obtain better performances.

4.4 Concatenation of ELMo with non-contextual embedding

We investigate also a mixed approach, concatenating ELMo with the other previously described embedding models, in order to obtain word embeddings with both contextualized and non-contextualized features.

In particular, we focuse on both the combinations with 300-dimensional GloVe and FastText Embeddings. For example, given a sentence of length l, we encode it as $x = \{x_1, x_2, ..., x_l\}$ where:

$$x_i = [x_i^{GloVe}, x_i^{ELMo}] \text{ or } x_i = [x_i^{FastText}, x_i^{ELMo}] \tag{4.1}$$

and $x_i \in \Re^d$. In the case of 300-dimensional non-contextual vectors, the value of d is equal to 1024+300 = 1324.

---

[4] `https://www.statmt.org/lm-benchmark/`

[5] `http://www.statmt.org/wmt11/translation-task.html`

CHAPTER 5

DATA

To evaluate our models, we focuse attention on labelled datasets on sarcasm that, in addition to the sole sarcastic statement (that we call *"Response"*), contain also the parent comment (*"Quote"*) that elicits the sarcastic utterance.

We perform our studies on the Sarcasm V2 Corpus[1], a dataset created by Oraby et al. (2017) [34]. Then, given the limited size of the dataset, we test our models also on a large-scale self-annotated corpus for sarcasm, SARC[2] [23]. In addition, we crawl some comments from Twitter in order to evaluate the accuracy of our final model on a real-word data distribution.

5.1 Sarcasm V2 Corpus

Sarcasm V2 is a dataset released by Oraby et al. (2017) [34]. It is a highly diverse corpus of sarcasm developed using syntactical cues and crowd-sourced annotation. It contains 4692 lines containing both Quote and Response sentences from dialogue examples on political debates from the Internet Argument Corpus (IAC 2.0). The data is collected and divided in three categories: General Sarcasm (Gen, 3260 comments), Rhetorical Questions (RQ, 582 comments) and Hyperbole (Hyp, 850 comments).

---

[1] `https://nlds.soe.ucsc.edu/sarcasm2`

[2] `http://nlp.cs.princeton.edu/SARC/`

As our model is supposed to work also with real-word data, we use only the Gen Corpus for our experiments.

5.2 SARC

The Self-Annotated Reddit Corpus (SARC) was introduced by Khodak et al. (2017) [23]. It contains more than a million sarcastic and non-sarcastic statements retrieved from Reddit, with some contextual information such as author details, score and parent comment (corresponding to the Quote text in Sarcasm V2).

Reddit is a social media site in which users can communicate on topic-specific discussion forums called *subreddits*, each titled by a post called *submission*. People can vote and reply to the submissions or to their comments, creating a tree-like structure. This guarantees that every comment has its "father". The main feature of the dataset is the fact that sarcastic sentences are directly annotated by the authors themselves, through the inclusion of the marker *"/s"* in their comments. This method provides reliable and trustful data. Another important aspect is the fact that almost every comment is made of one sentence.

As SARC dataset has many variants (Main Balanced, Main Unbalanced and Pol), in order to make our analyses more consistent with the Sarcasm V2 Corpus, we run our experiments only on the first version of the Main Balanced dataset, composed of an equal distribution of both sarcastic (505413) and non-sarcastic (505413) statements (Total training set size: 1010826). They also provide a balanced test set of 251608 comments for model evaluation.

5.3 Crawled Dataset

To prove the efficiency of our model, we create our own collection of sentences. The Corpus has been collected from the Twitter platform through their designated API. The decision to use Twitter is given to the fact that we wanted our model to work efficiently on multiple platforms. In order to retrieve relevant sarcastic data for our experiments, we studied many preexisting sarcastic datasets and, after further analyses, we found eight main topics for which sarcasm is mainly recurrent: Politics (Trump, Obama), Healthcare, Creationism, Abortion, Terrorism, Racism and Homophobia. We then selected three cities from different parts of the United States: Chicago (Center), San Francisco (West Coast), Philadelphia (East Coast) to evaluate how some circumstances, like climate or political tendencies, could affect people's behaviour with respect to a certain topic.

Finally we collected thousands of tweets on each one of these topics, written in the surroundings of the cities listed above between January and May 2019. As we can see from Table I, given that some arguments are more recurrent than others on social media, for every topic in every city we crawled a different number of tweets.

TABLE I: TOTAL NUMBER OF TWEETS CRAWLED

	Chicago	**Philadelphia**	**San Francisco**
Abortion	10268	6208	6694
Creation	3208	1578	6999
Health	50000	50000	50000
Homophobia	31033	13420	17092
Racism	43951	30853	34443
Terrorism	29130	17888	16320
Trump	50K	36185	45780
Obama	2273	1280	1222

Once all the data have been collected and cleaned, we chose the criteria according to which analyse them and make predictions. At first, we set the maximum size of the collections to 50.000. Then, to make our analysis more consistent, for every topic we selected the minimum number of crawled tweets between the 3 cities and cut down the collections of the other two cities to that size. At the end of this procedure our final crawled dataset had the configuration shown in Table II.

TABLE II: CONTENT OF THE FINAL CRAWLED DATASET FOR PREDICTIONS

	Chicago	**Philadelphia**	**San Francisco**
Abortion	6208	6208	6208
Creation	1578	1578	1578
Health	50000	50000	50000
Homophobia	13420	13420	13420
Racism	30853	30853	30853
Terrorism	16320	16320	16320
Trump	36185	36185	36185
Obama	1222	1222	1222

CHAPTER 6

EXPERIMENTS

6.1 Base Model

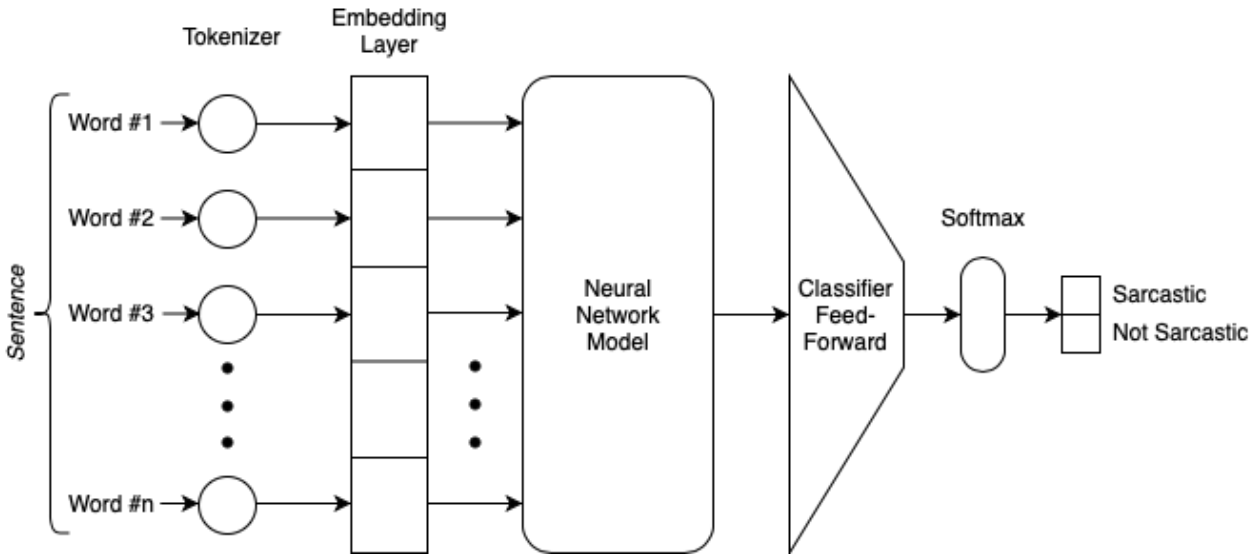

Figure 4. Base Model for the experiments.

The first experiments were made on the Sarcasm V2 dataset, since it is way smaller and faster to run than the SARC one. Our network framework is represented in the Figure 4.

As in the datasets the sarcastic and non-sarcastic comments (Response) are collected with their own parent comment (Quote), we attempted three different network configurations to study how contextual information from the quote can affect sarcastic predictions:

- Two Models (Quote Encoder + Response Encoder): the purpose of this network is to analyze both inputs separately and unify them through the feedforward network, in order to maintain the context features in the Quote (Figure 5);

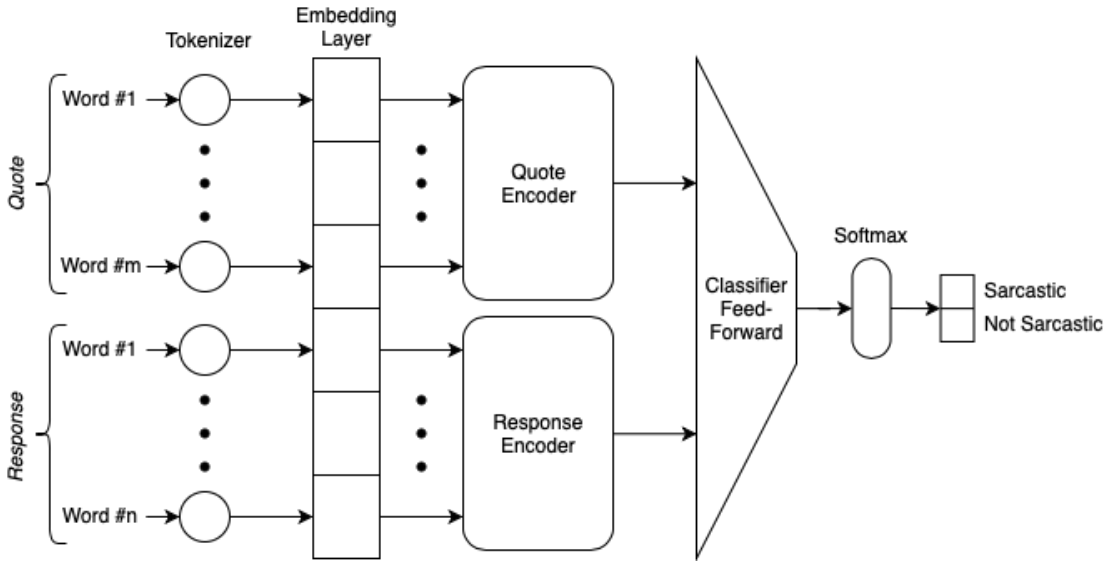

Figure 5. Two Models configuration.

- Quote_Response (Q+R): it consists on one model which takes a single input that joins quote and response separated only by a white-space. Its configuration is the same as in Figure 4, having *Sentence = Quote + " " + Response*.
- Response Only: it has a single model having only the Response as input (*Sentence = Response*). Even though this structure does not take into account the context information in the Quote, it is supposed to be useful to the network in order to capture the subtle patterns of sarcastic utterances.

6.2 Implementation

To obtain a reliable and good performing model, a supervised learning has been studied on the two sarcasm datasets. We implement our models using the AllenNLP library [12] on Python 3.6. To perform our experiments we use the AWS Platform, in particular we run EC2 instances (Ubuntu Deep Learning AMI) having one GPU on a **Pytorch** environment. We experiment with almost all the possible combinations of neural network models (Encoders) and word embeddings described in the chapters 3 and 4.

On each iteration, the input sentence is split into words, each word is embedded with a word embedding and then passed as input to the encoder. The internal parameters of the encoders may vary from one experiment to another. The encoded result of the model goes through a Feed Forward Network (also called Multi Layer Perceptron, MLP), having as input dimension the output size of the model (in the Two Models configuration it will be equal to the sum of the output sizes of both Quote and Response encoders), h hidden nodes, ReLU [33] activation function and a Dropout rate [42] of 0.2 between the input and the hidden layer. We then apply

a Softmax function [4] on the result, compute the class probability and output the label with highest value.

We iterate over the datasets with a batch of size equal to 16. The model parameters are tuned with AdaGrad optimizer [11] having gradient clipping threshold set to 5.0. For every epoch we compute F-Measure, Precision, Recall, Accuracy and Loss. The training is stopped once the validation accuracy ceases to grow after some consecutive epochs (the patience is generally set to 5).

6.3 Experiments on Sarcasm V2 Corpus

The experiments on Sarcasm V2 Corpus can be divided in two phases. In the first step we choose our best network configuration and in the second step we perform the effective evaluation of the dataset.

6.3.1 Experiments on Validation Set

In this part of the experiment we rely on the results obtained on the validation set to study the efficiency of our three model configuration. We split the Sarcasm V2 dataset creating two balanced subsets (randomly shuffled) having dimension of 80% for training and 20% for validation set. Then we experiment only with the basic Network models, LSTM, BiLSTM and CNN and evaluate their results.

6.3.1.1 LSTM

The LSTM model we applied in this experiment is a single layer Seq2Vec encoder with hidden size of 100 or more cells and a dropout rate of 0.2. The input size corresponds on the

dimension of the embedding applied on it. Table III describes also the hidden and input size of the feed-forward network.

TABLE III: EXPERIMENT ON VALIDATION SET WITH LSTM ENCODER

Config	**Embedding Type**	**Embedding Dim**	**Hidden Size**	**FF input**	**FF hidden**	**Acc**	**F**	**P**	**R**
Two Models	GloVe	50	100	200	100	0.7450	0.7573	0.7214	0.7969
Q + R	GloVe	50	100	100	50	0.7435	0.7337	0.7616	0.7077
Resp Only	GloVe	50	100	100	50	0.7742	0.7692	0.7853	0.7538
Two Models	GloVe	100	100	200	100	0.7604	0.7746	0.7302	0.8246
Q + R	GloVe	100	100	100	50	0.7358	0.7354	0.7354	0.7354
Resp Only	GloVe	100	100	100	50	0.7650	0.7733	0.7457	0.8031
Two Models	GloVe	300	100	200	100	0.7435	0.7458	0.7380	0.7538
Q + R	GloVe	300	100	100	50	0.7419	0.7308	0.7625	0.7015
Resp Only	GloVe	300	100	100	50	**0.7834**	**0.7867**	0.7738	0.8
Two Models	FastText	300	100	200	100	0.7450	0.7573	0.7214	0.7969
Q + R	FastText	300	100	100	50	0.7650	0.7678	0.7575	0.7785
Resp Only	FastText	300	100	100	50	0.7788	0.7778	0.7802	0.7754
Two Models	ELMo	1024	200	400	200	0.7742	0.7699	0.7834	0.7569
Q + R	ELMo	1024	400	400	200	0.7819	0.7848	0.7731	0.7969
Resp Only	ELMo	1024	400	400	200	0.7803	0.7797	0.7809	0.7785
Two Models	ELMo+GloVe	1024+100	200	400	200	0.7742	0.7879	0.7418	**0.84**
Q + R	ELMo+GloVe	1024+100	400	400	200	0.7604	0.7570	0.7666	0.7477
Resp Only	ELMo+GloVe	1024+100	400	400	200	0.7588	0.7464	**0.7857**	0.7108
Two Models	ELMo+FastText	1024+300	200	400	200	0.7727	0.7744	0.7674	0.7815
Q + R	ELMo+FastText	1024+300	400	400	200	0.7634	0.7701	0.75	0.7914
Resp Only	ELMo+FastText	1024+300	400	400	200	0.7680	0.7769	0.7493	0.8067

6.3.1.2 BiLSTM

The BiLSTM model has the same features as LSTM, the only difference is that the Multi Layer Perceptron takes twice the hidden size (forward + backward LSTM) as input.

TABLE IV: EXPERIMENT ON VALIDATION SET WITH BILSTM ENCODER

Config	**Embedding Type**	**Embedding Dim**	**Hidden Size**	**FF input**	**FF hidden**	**Acc**	**F**	**P**	**R**
Two Models	GloVe	50	100*	200+200	200	0.7512	0.7699	0.7150	0.8338
Q + R	GloVe	50	100*	200	100	0.7327	0.7500	0.7035	0.8030
Resp Only	GloVe	50	100*	200	100	0.7650	0.7740	0.7443	0.8062
Two Models	GloVe	100	100*	200+200	200	0.7419	0.7399	0.7445	0.7354
Q + R	GloVe	100	100*	200	100	0.7143	0.7513	0.6643	**0.8646**
Resp Only	GloVe	100	100*	200	100	0.7665	0.7725	0.7522	0.7938
Two Models	GloVe	300	100*	200+200	200	0.7465	0.7660	0.7105	0.8308
Q + R	GloVe	300	100*	200	100	0.7358	0.7410	0.7257	0.7569
Resp Only	GloVe	300	100*	200	100	0.7634	0.7781	0.7317	0.8308
Two Models	FastText	300	100*	200+200	200	0.7512	0.7699	0.7150	0.8338
Q + R	FastText	300	100*	200	100	0.7604	0.7600	0.76	0.76
Resp Only	FastText	300	100*	200	100	0.7680	0.7743	0.7530	0.7969
Two Models	ELMo	1024	200*	400+400	200	**0.7757**	**0.7853**	0.7521	0.8215
Q + R	ELMo	1024	400*	800	400	0.7588	0.7625	0.75	0.7754
Resp Only	ELMo	1024	400*	800	400	0.7665	0.7625	0.7746	0.7508
Two Models	ELMo + GloVe100	1024+100	200*	400+400	200	0.7680	0.7802	0.7403	0.8246
Q + R	ELMo + GloVe100	1024+100	400*	800	400	0.7650	0.7552	**0.7867**	0.7262
Resp Only	ELMo + GloVe100	1024+100	400*	800	400	0.7680	0.7615	0.7825	0.7415
Two Models	ELMo + FastText	1024+300	200*	800	400	0.7696	0.7845	0.7358	0.84
Q + R	ELMo + FastText	1024+300	400*	800	400	0.7496	0.7606	0.7296	0.7945
Resp Only	ELMo + FastText	1024+300	400*	800	400	0.7527	0.7736	0.7143	0.8436

6.3.1.3 CNN

The CNN encoder is set with only one filter of size equal to 5. We fix the output dimension of the network thanks to an internal fully connected layer (projection layer) already implemented in the AllenNLP library.

TABLE V: EXPERIMENT ON VALIDATION SET WITH CNN ENCODER

Config	**Embedding Type**	**Embed. Dim**	**N Filters**	**Output Dim**	**FF input**	**FF hidden**	**Acc**	**F**	**P**	**R**
Two Models	GloVe	50	100	100	100+100	100	0.7435	0.7504	0.7297	0.7723
Q + R	GloVe	50	130	100	100	50	0.7066	0.7170	0.6914	0.7446
Resp Only	GloVe	50	130	100	100	50	0.7081	0.6975	0.7227	0.6738
Two Models	GloVe	100	100	100	100+100	100	0.7266	0.7236	0.7304	0.7169
Q + R	GloVe	100	130	100	100	50	0.7174	0.7437	0.6794	0.8215
Resp Only	GloVe	100	130	100	100	50	0.7435	0.7618	0.7101	0.8215
Two Models	GloVe	300	100	100	100+100	100	0.7327	0.7290	0.7382	0.72
Q + R	GloVe	300	130	100	100	50	0.7327	0.7356	0.7267	0.7446
Resp Only	GloVe	300	130	100	100	50	0.7665	0.7765	0.7437	0.8123
Two Models	FastText	300	130	100	100+100	100	**0.7972**	**0.8002**	0.7629	**0.8615**
Q + R	FastText	300	130	100	100	50	0.7389	0.7578	0.7056	0.8185
Resp Only	FastText	300	130	100	100	50	0.7926	0.7874	**0.8065**	0.7692
Two Models	ELMo	1024	100	100	100+100	100	0.7680	0.7722	0.7574	0.7877
Q + R	ELMo	1024	400	400	400	200	0.7558	0.7512	0.7643	0.7385
Resp Only	ELMo	1024	400	400	400	200	0.7527	0.7440	0.7697	0.72
Two Models	ELMo + GloVe100	1024+100	100	100	100+100	100	0.7680	0.7716	0.7589	0.7846
Q + R	ELMo + GloVe100	1024+100	400	400	400	200	0.7389	0.7592	0.7034	0.8246
Resp Only	ELMo + GloVe100	1024+100	400	400	400	200	0.7711	0.7773	0.7558	0.8
Two Models	ELMo + FastText	1024+300	130	100	100+100	100	0.7696	0.7774	0.7507	0.8062
Q + R	ELMo + FastText	1024+300	130	100	100	50	0.7435	0.7443	0.7431	0.7454
Resp Only	ELMo + FastText	1024+300	130	100	100	50	0.7557	0.7686	0.7313	0.8098

6.3.1.4 Analysis on the Validation Set

As we can see from the tables, the Q+R model has lower performances than the others. Probably, analyzing the context phrase and the sarcastic utterance together in the same string, our model was not able to capture many contextual information, making the whole sentence less dense of sarcastic content and thus harder to classify. So, Q+R was the first framework that we discarded before proceeding to our next experiments.

As the predictions had to be done on Twitter comments, which do not have any context information such as a quote or a parent comment, and given that according to our results the Two Models and Response Only structures have similar performances, we chose to continue our research using the Response Only framework. This choice was also favored by the fact that the Two Models framework needs more memory to be run causing an increase in the training time. Once the best structure has been chosen we proceeded to the next step of our experiment.

6.3.2 Evaluation on the Test set

The next step of our research consists on the evaluation of the dataset. Oraby et al. (2017) [34] performed a Supervised Learning using SVM and, as the Sarcasm V2 dataset has small dimensions, they executed a 10-fold cross-validation on the data to obtain the state-of-art metrics shown in Table VI.

For our approach, we randomly divided the Gen Dataset into 90% training and 10% test set. Then we split the temporary training set into 80% training and 20% validation set. We performed this procedure three times using a different random seed, obtaining three sets of data. It is important to note that all the subdivisions were maintained balanced (i.e. with the same

number of sarcastic and non-sarcastic data). We ran the same model over the three created sets and computed the mean values of the metrics we obtained through the 3 executions. We set a fixed value of random seed, pytorch seed and numpy seed to compare the results on the same footing.

6.4 SARC analysis

While on the Sarcasm V2 Corpus we were able to perform many experiments with different network models, for the SARC dataset, which is hundreds times bigger, we used a different approach to deal with its magnitude (An epoch with ELMo embeddings can last more than 7 hours).

As Khodak et al. (2017) [23] provided also a balanced test set for the training task, we only had to create our own validation set. We firstly removed some noise data from our training. About 40 empty comments were found and deleted, equivalent to a really small percentage of the dataset, a negligible quantity that cannot affect our models' performance. We then divided our original training set into 80% training and 20% validation. Both collections have been shuffled and maintained balanced.

6.4.1 SARC Training

Basing on the results obtained on Sarcasm V2, we performed our evaluation with the best performing configurations from previous experiment. We tested our models using a smaller sample of 100.000 lines as training set (keeping untouched validation and test set), in order to obtain significant results in less time. We then selected the framework with the highest accuracy

value and performed the experiment with the whole dataset to compare our performance with the baseline.

6.5 Multi-Tasking Approach

Another method studied to improve the accuracy of our models is the Multi-Tasking approach. Basing on the work done by Cohan et al. (2019) [9] for Citation Intent classification, we aimed to combine our base models with another auxiliary secondary task in order to enhance the performance of the classification.

Given that the SARC dataset does not contain a lot of relevant information to improve sarcasm detection (Ups, Down and Score are not strongly correlated with sarcasm) and considering that, as we stated in Chapter 2, emotion and sentiment play an important role for detecting sarcasm, we tried to add another useful information to the dataset: the Sentiment Label.

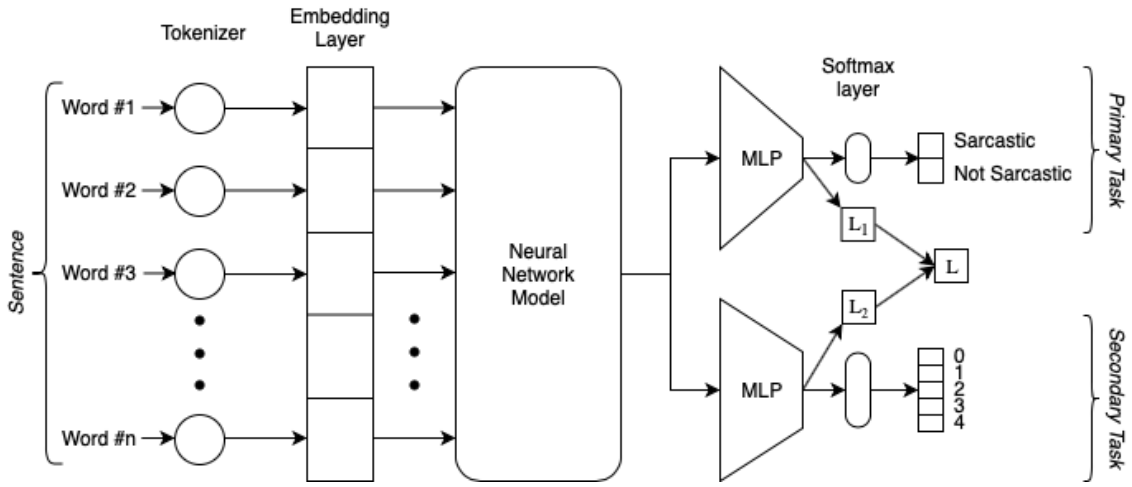

Figure 6. Multi-tasking framework.

Figure 6 shows the multitasking framework we used for our experiment, having sentiment detection as secondary task. We can see that both tasks share the same model and embedding, but they have their own Multi Layer Perceptrons. However, in our model, we separated the embeddings and the models' parameters too, because our CUDA did not have enough memory to run both the tasks with ELMo embeddings. So, we used ELMo embeddings (and their concatenations) only for the primary task.

The configuration of our framework allows the secondary task to affect the training on the first task when computing the loss of the model. In fact, expressing Ω_1 as the sarcastic labeled

dataset and Ω_2 as the sentiment labeled dataset, the total loss L of our framework is computed with the formula:

$$L = \sum_{(x,y)\in\Omega_1} L_1(x,y) + \lambda \sum_{(x,y)\in\Omega_2} L_2(x,y), \tag{6.1}$$

where L_i is the Cross Entropy loss computed for each sentence in the database Ω_i, between the desired output y and the output x predicted by our network. The index $i \in \{1,2\}$ distinguishes the primary task ($i = 1$) from the auxiliary ($i = 2$) one. The term λ is a hyper-parameter set to 0.06 to limit the impact of the secondary task on the training parameters. In every epoch, our algorithm computes the gradient of L for each batch and propagates it to tune our model parameters through the AdaGrad optimizer.

6.5.1 Sentiment Analysis

In order to add a sentiment label to our dataset, we employed the pre-trained model for sentiment analysis available on Stanford NLP website[1]. The algorithm, conceived by Socher et al. (2013) [41], scans every text content one sentence at the time, assigning to each one an integer score from 0 to 4, that corresponds to the labels: Very Negative(0), Negative(1), Neutral(2), Positive(3), Very Positive(4).

The main challenges we faced in this operation were: the massive size of SARC dataset, the presence of noises and the treatment of lines composed of more than one sentence. To overcome the first problem, we converted the original csv file into a text file, which is easier to handle,

---

[1] `https://nlp.stanford.edu/sentiment/`

keeping for each line only the sentences to be analyzed (Response comments), their sarcasm label and their number of line in the dataset (in order to track any possible errors).

The second issue was the most challenging one. As the dataset is very big and retrieved from a social network like Reddit, it is possible to find in it comments more than 1000 characters long, and sometimes without any punctuation marks to split the sentences, as well as comments without any character. For as regards the empty comments, we decided to delete all of them since their exclusion does not affect the balance in labeling (about 40 comments out of more than a million is a negligible amount for the results of our study). The long posts, on the other hand, were threaten differently. Whenever it was possible, we split them into multiple sentences, labeled each sentence and then taken their mean score as final label.

There were also some sentences without punctuation marks (e.g one, two or three words repeated hundreds of times or receipts or technical specifications). In the first case we computed the sentiment score just analyzing the same word repeated only 5 times while, in the second, we assigned a Neutral (2) score to all the receipts or try to analyze pieces of the whole sentence.

6.5.2 Finding the most performing auxiliary task

In order to increase the effectiveness of the multitasking framework we focused our attention on the secondary task. We believe that finding the best possible configuration for the auxiliary task could improve the overall performance of the whole model. We attempted to run all our networks with 300-dimensional GloVe vectors or FastText embedding on the SARC dataset, with the same training configuration described in Section 6.2, having the Average F-score as

stopping criteria. We decided to choose this metric instead of accuracy to take more into account those labels who appear less frequently (i.e. Very Negative(0), Very Positive(4)).

After some attempts we figured out that the BiLSTM model with FastText embedding appeared the best configuration overall. We avoided to evaluate the models with ELMo embedding as our GPU did not have enough memory to handle 2 tasks with ELMo embeddings in a such huge dataset.

6.5.3 Multitask Training phase

Once all the data has been labeled and the best auxiliary task network has been picked, we proceeded to the training phase. Also in this case, we ran our models on the smaller training set and applied the best configuration to the original SARC dataset. The batch size for the multitasking approach was set to 8 (for both the tasks) to prevent any GPU out-of-memory errors. The auxiliary task model encodes the input tokens with FastText word representations and passes them through a single-layer BiLSTM network with hidden dimension size of 100 for each direction, to capture sentiment information about the whole sentence. The output of the BiLSTM model is sent to a fully connected single-layer MLP with 20 hidden nodes, ReLU activation and a Dropout rate of 0.2 between the input and the hidden layer. The output, having size equals to the number of labels (i.e. 5), is modified by a softmax layer to obtain class probabilities. The class with the highest probability represents the predocted sentiment label. The main task's configuration, on the other hand, depends on the model used and has usually the same parameters described in Chapters 3 and 4.

We applied the multitasking approach also to the Sarcasm V2 Corpus and we obtained our best state-of-the-art performance.

6.6 Experiments on the Crawled Dataset

As we have pointed out in Chapter 2, in the task of sarcasm detection also punctuation, capital letters and abbreviations take on significant importance. For example, the phrase *"I'm so happy!"* may assume sarcastic connotation if written as *"I'm SO happy."*. The capital *"SO"* and the presence of the full stop in place of the exclamation point may drastically change the meaning of the sentence. Another example may be *"You're sooooooo funny"*, in which the prolongation of the word *"so"* helps understanding more the purpose of the statement. In addition, nowadays, even the use of emoticons is widespread and, as demonstrated by Carvalho et al. (2009) [7] and González-Ibánez et al. (2011) [17], they contain relevant information for sarcasm detection.

For all these reasons we decided to avoid stemming and lemmatizing. In our preprocessing phase on our crawled dataset from Twitter, we removed all usernames, links (and the word "via" that usually precedes them), and considered only the tweets with 5 or more words. The only exception was represented by Trump's username which was replaced by the word "Trump" itself to preserve the meaning of the sentences.

Once all the data was collected and cleaned, we selected the best model from the previous experiments in order to make some predictions and comparisons. For this prediction task, we decided to use the best models found on the SARC dataset. As it was trained on more data,

we thought it could be able to capture more information and detect more sarcastic utterances on different topics than the model trained on the Sarcasm V2 Corpus.

Given that our predictors produce a set of class probabilities as output, we examined our results considering two threshold values for the sarcastic label, 0.5 and 0.6.

CHAPTER 7

RESULTS

In this section we show and analyze the results obtained for the experiments described in the previous section, discussing how the features of the multitasking framework affect the performance of the sarcasm detection task.

7.1 Results on Sarcasm Corpus V2

Oraby et al. (2017) [34] run their experiment using only two types of features: Word2Vec (W2V) and N-grams. On the rightmost column of Table VI we show their state-of-art results, obtained with Word2Vec [30] embeddings trained on Google News as features and F-measure computed on the Sarcastic label.

The results on the table show the mean metrics for every couple of NN Model and Word Embedding. All the experiment were run with a batch size equal to 16. Given the size of the dataset, we consider F1-measure as the most significant metric for the evaluation. Also, the previous state-of-the-art by Oraby et al. (2017) [34] reports only the values of F1, Precision and Recall. It is important to mention that CNN and CNN-LSTM models have the same configuration and parameters (i.e. number of convolutions, number of filters, filter sizes, input dimension) for the convolutional framework. Also the BiLSTM used in the Attention model has the same configuration applied for the simple BiLSTM network. Whereas, the only difference

between the BiLSTM and LSTM framework is the output dimension, which is twice the size in BiLSTM.

We can notice from the results in Table VI that almost all our experiments outperform the pre-existent state-of-art for the Sarcasm Corpus V2. This outcome demonstrates the effectiveness of Deep Learning neural network models for the sarcasm detection task. In particular, we can observe that simple BiLSTM networks obtain good performances with non-contextual embeddings (GloVe100, FastText), while CNN-LSTM works efficiently with all the kinds of embeddings, except for the simple FastText. However, FastText embeddings outperform GloVe embedding when applied to all the other models.

The best performing model in our evaluation is the BiLSTM framework with the Attention mechanism having as input the concatenation of contextual (ELMo) and non-contextual (GloVe300) embedding. It consists on a single-layer BiLSTM with an input size of 1324 (1024(ELMo) + 300(GloVe)), an hidden size of 100 cells and a dropout rate of 0.2, which is scanned twice in opposite directions creating an output of size 200. This output feeds a MLP with the same features and activation functions described in the Section 6.2 and hidden size h equal to 20.

TABLE VI: RESULTS ON SARCASM V2 CORPUS (WITHOUT MULTITASKING)

Embedding	**Metrics**	**BiLSTM**	**LSTM**	**CNN**	**Attention**	**CNN-LSTM**	**Oraby et al. (2017)**
GloVe50	Acc	0.7321	0.7362	0.7065	0.7536	**0.7556**	
	F1	0.7381	0.7259	0.7045	0.7566	**0.7598**	0.74
	Prec	0.7256	**0.7536**	0.7098	0.7481	0.7458	0.71
	Rec	0.7526	0.7035	0.7014	0.7669	**0.7751**	0.77
GloVe100	Acc	**0.7658**	0.7515	0.7219	0.7515	0.7515	
	F1	**0.7632**	0.7508	0.7201	0.7576	0.7529	0.74
	Prec	**0.7715**	0.7531	0.7254	0.7425	0.7500	0.71
	Rec	0.7566	0.7485	0.7198	**0.7751**	0.7587	0.77
GloVe300	Acc	0.7372	0.7372	0.7168	**0.7485**	0.7474	
	F1	0.7236	0.7361	0.7265	0.7426	**0.7507**	0.74
	Prec	0.7619	0.7383	0.7006	**0.7647**	0.7458	0.71
	Rec	0.6912	0.7342	0.7566	0.7260	**0.7587**	0.77
FastText Crawl	Acc	**0.7791**	0.7597	0.7607	0.7638	0.7495	
	F1	**0.7748**	0.7620	0.7558	0.7654	0.7430	0.74
	Prec	**0.7887**	0.7556	0.7718	0.7645	0.7615	0.71
	Rec	0.7628	**0.7689**	0.7423	0.7689	0.7301	0.77
ELMo	Acc	0.7720	0.7740	0.7648	0.7669	**0.7781**	
	F1	0.7603	0.7716	0.7646	0.7639	**0.7731**	0.74
	Prec	0.7996	0.7782	0.7662	0.7787	**0.7911**	0.71
	Rec	0.7280	0.7689	**0.7710**	0.7566	0.7566	0.77
ELMo + GloVe100	Acc	0.7597	0.7607	0.7607	0.7597	**0.7740**	
	F1	0.7577	0.7521	0.7604	0.7554	**0.7736**	0.74
	Prec	0.7619	**0.7814**	0.7641	0.7743	0.7783	0.71
	Rec	0.7546	0.7260	0.7587	0.7403	**0.7710**	0.77
ELMo + GloVe300	Acc	0.7658	0.7781	0.7669	**0.7822**	0.7689	
	F1	0.7667	0.7764	0.7641	**0.7844**	0.7703	0.74
	Prec	0.7659	**0.7824**	0.7734	0.7765	0.7684	0.71
	Rec	0.7689	0.7710	0.7566	**0.7935**	0.7751	0.77
ELMo + FastText	Acc	0.7720	0.7597	0.7648	**0.7791**	0.7679	
	F1	0.7666	0.7574	0.7511	**0.7798**	0.7749	0.74
	Prec	0.7867	0.7639	**0.7937**	0.7766	0.7527	0.71
	Rec	0.7526	0.7526	0.7219	0.7832	**0.7996**	0.77

7.2 Results on Sarcasm Corpus V2 with Multitasking

Table VII shows the results when applying the multitasking approach to the frameworks used in the previous experiments. The configuration and the parameters of the models we evaluated have remained unchanged. We only modified them adding the auxiliary task for detecting sentiment and setting the batch size of both the tasks to 8, in order to prevent CUDA-out-of-memory errors.

TABLE VII: RESULTS USING MULTITASKING ON SARCASM V2 CORPUS

Embedding	Metrics	BiLSTM	LSTM	CNN	Attention	CNN-LSTM	Oraby et al. (2017)
GloVe50	Acc	0.7362	0.7505	0.7219	**0.7556**	0.7393	
	F1	0.7416	0.7541	0.7185	**0.7636**	0.7483	0.74
	Prec	0.7272	**0.7437**	0.7273	0.7398	0.7290	0.71
	Rec	0.7566	0.7648	0.7117	**0.7894**	0.7710	0.77
GloVe100	Acc	**0.7546**	0.7536	0.7229	0.7413	0.7454	
	F1	0.7539	0.7549	0.7264	**0.7595**	0.7445	0.74
	Prec	**0.7572**	0.7518	0.7158	0.7083	0.7492	0.71
	Rec	0.7566	0.7587	0.7403	**0.8200**	0.7423	0.77
GloVe300	Acc	0.7382	**0.7607**	0.7280	0.7434	0.7485	
	F1	0.7308	**0.7594**	0.7206	0.7425	0.7515	0.74
	Prec	0.7534	**0.7639**	0.7411	0.7437	0.7425	0.71
	Rec	0.7137	**0.7607**	0.7014	0.7423	0.7607	0.77
FastText Crawl	Acc	**0.7597**	0.7536	0.7444	0.7577	0.7515	
	F1	0.7581	0.7539	0.7401	**0.7602**	0.7487	0.74
	Prec	**0.7630**	0.7533	0.7524	0.7530	0.7535	0.71
	Rec	0.7566	0.7546	0.7301	**0.7689**	0.7464	0.77
ELMo	Acc	0.7679	**0.7781**	0.7628	0.7781	0.7751	
	F1	0.7767	**0.7895**	0.7756	0.7820	0.7787	0.74
	Prec	0.7490	0.7511	0.7355	**0.7678**	0.7636	0.71
	Rec	0.8078	**0.8323**	0.8262	0.7975	0.7955	0.77
ELMo + GloVe100	Acc	0.7556	0.7536	**0.7699**	0.7587	0.7546	
	F1	0.7640	0.7533	**0.7792**	0.7692	0.7582	0.74
	Prec	0.7430	**0.7523**	0.7492	0.7438	0.7475	0.71
	Rec	0.7894	0.7566	**0.8119**	0.7996	0.7710	0.77
ELMo + GloVe300	Acc	0.7648	0.7771	0.7710	**0.7771**	0.7618	
	F1	0.7734	0.7818	0.7803	**0.7842**	0.7736	0.74
	Prec	0.7521	**0.7671**	0.7495	0.7629	0.7351	0.71
	Rec	0.7996	0.7975	**0.8139**	0.8078	0.8200	0.77
ELMo + FastText	Acc	0.7597	0.7526	0.7638	**0.7730**	0.7495	
	F1	0.7588	0.7634	0.7777	**0.7843**	0.7566	0.74
	Prec	0.7598	0.7326	0.7346	**0.7517**	0.7364	0.71
	Rec	0.7587	0.7996	**0.8262**	0.8221	0.7791	0.77

From these results, we can notice that the multitasking method increases the performances of the simplest models, such as CNN and LSTM, up to a 2.8% on F-measure but it does not work well on the CNN-LSTM network. This may be caused by the complexity of the CNN-LSTM network with respect to the simple BiLSTM structure of the sentiment detection task. Our framework has also bad performances with FastText, while generally improves the efficiency of all the other embeddings. The BiLSTM with attention model, on the other hand, does not noticeably vary in performance, but reaches the highest values for most of the embeddings.

However, the new state-of-the-art model that we developed for Sarcasm V2 Dataset consists on a primary task with a single-layer LSTM network, having only ELMo contextual embedding as input (size = 1024), a hidden layer of 100 cells which is passed as input on the MLP network with an hidden size h equal to 20. Our model outperforms the previous state-of-the-art by about 5% in all the metrics and highlights the importance of context in the sarcasm detection task.

7.3 Results on SARC

Table VIII shows the results, expressed in term of accuracy, obtained running our experiment on SARC dataset with a training set size reduced to 100.000 comments and considering only the embeddings of high dimensions. We also run our experiments with a version of ELMo (ELMo 5.5B) pre-trained on a dataset of 5.5B tokens from Wikipedia and news crawldata from WMT 2008-2012.

TABLE VIII: RESULTS ON A SMALLER TRAINING SET OF SARC (WITHOUT MULTITASKING)

	BiLSTM	**LSTM**	**CNN**	**Attention**	**CNN-LSTM**
Glove300	0.70539	0.70493	0.68801	0.70508	0.69627
FastText Crawl	0.71763	0.71322	0.71309	0.72061	0.71357
Elmo	0.72884	0.72717	0.72250	0.73049	0.72026
Elmo 5.5	0.73111	0.73268	0.72482	0.73134	0.72656
Glove 300 Elmo	0.73046	0.73188	0.72626	0.73207	0.72412
Glove 300 Elmo 5.5B	0.73049	0.73201	0.72684	0.73431	0.72689
FastText Elmo	0.73346	0.73355	0.72990	0.73446	0.72721
FastText Elmo 5.5B	0.73383	0.73459	0.73102	**0.73503**	0.73102

From these results we can notice that the LSTM-based models (LSTM, BiLSTM, BiLSTM with attention) perform generally better than the CNN-based model (CNN, CNN-LSTM). In particular, the BiLSTM with Attention model appears to outperform all the other models for almost every typology of embedding (except for GloVe 300-dimensional vectors and ELMo 5.5B). In addition, the version of ELMo trained on 5.5B tokens gives better results than the version trained 1 Billion Word Benchmark in all the instances. From the results we can also observe that the concatenation of FastText embeddings with ELMo 5.5B embedding seems the most efficient embedding overall.

The best model for this experiment is a BiLSTM encoder with 300 cells in its hidden size, trained with the concatenation of ELMo 5.5b and FastText, that feeds a Multi Layer Perceptron with an hidden size of 200 neurons.

7.4 Results on SARC with MultiTasking

Table IX shows the outcomes of the same experiment obtained using a Multitasking approach and a batch size equal to 8.

TABLE IX: RESULTS USING MULTITASKING ON A SMALLER TRAINING SET OF SARC

	BiLSTM	**LSTM**	**CNN**	**Attention**	**CNN-LSTM**
Glove300	0.70543	0.70438	0.68718	0.70552	0.69476
FastText Crawl	0.71935	0.72061	0.71133	0.71851	0.71481
Elmo	0.72829	0.73130	0.72483	0.73161	0.72318
Elmo 5.5	0.72908	0.73301	0.72632	0.73196	0.72593
Glove 300 Elmo	0.73001	0.73177	0.72502	0.73025	0.72308
Glove 300 Elmo 5.5	0.73137	0.73046	0.72561	0.72998	0.72563
FastText Elmo	0.73471	0.73352	0.73105	0.73256	0.72641
FastText Elmo 5.5	**0.73583**	**0.73580**	0.73167	0.73575	0.72833

We can observe that in this experiment LSTM and BiLSTM still achieve good performances with the concatenation of contextual and non-contextual embeddings. In addition, also in these instances the ELMo version trained on 5.5B tokens is slightly more efficient than the other one. In every single model, the concatenation of ELMo 5.5B with FastText outperforms all the other embeddings combinations from 1% to 3% in accuracy.

As in the Sarcasm V2 Corpus, also for this dataset the CNN-LSTM model does not improve its performances with the multitasking approach. We can notice that the Multitasking framework has slightly better results than the basic framework for most of the FastText embeddings, the ELMo embeddings and their concatenation.

Also for this dataset, the best performing model overall is obtained using the Multitasking approach. Our best model main task consists on a BiLSTM encoder, trained with ELMo 5.5B and FastText, with 300 cells in its hidden layer, that feeds a MLP with an hidden size equal to 200. The auxiliary task uses a BiLSTM as well, trained with FastText embeddings, with 100 hidden cells and a MLP with 20 neurons in its hidden layer We kept this configuration, ran the same experiment on the whole SARC dataset and examined its performance with respect to other SARC's baseline models.

7.5 Comparison with Baseline methods on SARC

We compared our best model with almost the same state-of-the-art networks and baselines examinated by Hazarika et al. (2018) [18] on the Main Balanced version of the SARC dataset :

- Bag-of-words: a model that uses an SVM classifier having comment's word counts as input features.

- CNN: a simple CNN that can only model the content of a comment.
- CNN-SVM: model developed by Poria et al. (2016) [37] that exploits a CNN to model the content of the comments and other pre-trained CNNs to extract from them sentiment, emotion and personality features. All these features are joined and passed to an SVM to perform classification.
- CUE-CNN: method proposed by Amir et al. (2016) [1] that also models user embeddings combined with a CNN thus forming the CUE-CNN model.
- Bag-of-Bigrams: previous state-of-the-art model for this dataset, by Khodak et al. (2017) [23], that uses the count of bigrams in a document as vector features.
- CASCADE (ContextuAl SarCAsm DEtector): method proposed by Hazarika et al. (2018) [18] that uses user embeddings to model user personality and stylometric features, and combines them with a CNN to extract content features. We propose the results from both the versions, with and without personality features, in order to emphasize the efficiency of our model even though it does not employ any user personality feature.

TABLE X: COMPARISON WITH THE BASELINES OF MAIN BALANCED SARC DATASET

Models	**Accuracy**	**F1**
Bag-of-words	0.63	0.64
CNN	0.65	0.66
CNN-SVM [37]	0.68	0.68
CUE-CNN [1]	0.70	0.69
Bag-of-Bigrams[23]	0.758	N/A
CASCADE [18](no personality features)	0.68	0.66
CASCADE [18]	**0.77**	**0.77**
Our MultiTask BiLSTM	**0.764**	**0.763**

It can be observed that our model exceeds the previous state-of-the-art by Khodak et al. (2017) [23] by 0.6% and ouperforms all the other models that do not use personality features (Bag-of-words, CNN, CASCADE) by 6-8%, even the CNN-SVM and the CUE-CNN that model user embeddings. Its accuracy is only 0.6% lower than the current state-of-the-art (CASCADE with personality features). We believe that upgrading our framework with structures like user embeddings, to take into account personality features or other contextual information, could outperform the actual state-of-the-art.

7.6 Best Model Prediction

We used the previously described model also to predict the tweets in our crawled dataset. The results are shown in Table XI and Table XII.

TABLE XI: SARC BEST MODEL PREDICTIONS WITH 0.5 THRESHOLD

Threshold = 0.5			
	CHI	**PHI**	**SF**
Abortion	38.27	38.38	**39.27**
Creation	**30.48**	27.69	28.83
Health	39.17	**39.68**	39.31
Homophobia	46.92	**48.26**	43.41
Obama	37.73	**39.85**	39.36
Racism	49.25	**50.71**	49.95
Terrorism	**44.47**	43.42	44.26
Trump	30.99	31.77	**33.19**

TABLE XII: SARC BEST MODEL PREDICTIONS WITH 0.6 THRESHOLD

Threshold=0.6			
	CHI	**PHI**	**SF**
Abortion	28.96	29.03	**30.51**
Creation	**22.56**	20.22	19.90
Health	30.12	**30.32**	30.11
Homophobia	35.52	**37.00**	32.72
Obama	27.41	**29.38**	**29.38**
Racism	39.77	**41.01**	40.73
Terrorism	34.35	33.30	**34.78**
Trump	21.9	22.35	**24.14**

We can notice that increasing the threshold value of the sarcastic label class probability from 0.5 to 0.6 causes a loss of 10% of sarcastic sentences detected. In this way, phrases like: *"FOX News and Donald Trump, all lies, all the time."* which were predicted as sarcastic, probably because of their noticeable criticism, are not considered and the statistics become more accurate. From these results, Philadelphia appears to be the most sarcastic city of the ones we chose.

Taking into account the single topics we can notice that Homophobia and Racism are the most sarcastic arguments in tweets. Going manually through the classified sentences we noticed that for those topics, the number of false positives is relevant. For example, the sentence: *"It was only a matter of time before white gay privilege became the problem"* is classified as sarcastic

with a class probability of 90% by our classifier. This mistake is probably due to the fact that the word *"gay"* is commonly used to convey contempt and thus perceived as an highly sarcastic expression. The same consideration can be done regarding Racism. In fact, the tweet *"You are a racist dog. Your words are empty as your brain is"* is recognized as sarcastic with the 70% of class probability but it appears to be just a taunt, with no sarcastic meaning. Probably the fact that the SARC dataset does not treat deeply the themes of homophobia and racism could increase the classification errors in this task. In fact, for topics related to the politics (i.e. Trump, Obama), which are more featured in SARC, the percentage of sarcastic statements looks more realistic and we did not notice any relevant recurrent mistake in the predictions.

7.6.1 Comparisons with Reality

From the results we obtained on the model trained on the SARC dataset, we can notice some patterns which are coherent with the reality.

For example, from the results, Chicago appears to be 2% less sarcastic than the other cities on Obama topic. This instance may find an explanation in the reality: as Obama lived in Chicago, it is highly possible that the inhabitants of Chicago are very fond and thus, less sarcastic towards him.

In addition, our statistics support some evidence that could be confirmed by the results of the last elections: San Francisco's sarcastic rate on Trump is about 2% higher (700 tweets) than the other cities. We can also see, from a more extended point of view, that Philadelphia appears to be the most sarcastic city in the US, which could be an index to represent the criticism and the pessimistic view of the East Coast cities. However, given the amount of false

positive encountered, in particular for topics like Terrorism or Homophobia, this assumption should not be considered axiomatic.

CHAPTER 8

CONCLUSION AND FUTURE WORK

Sarcasm is a complex phenomenon which is hard to understand even to humans. In our work we showed the efficiency of using neural network with Word Embeddings to predict it accurately.

We demonstrated how sarcastic statements can be recognized automatically with a good accuracy even without recurring to further contextual information such as users' historical comments or parent comments. We explored a new multitasking framework to exploit the correlation between sarcasm and the sentiment it conveys. Except for few peculiar instances (e.g. CNN-LSTM network), the addition of a new sentiment detection task to our configuration improved moderately the effectiveness of our models. However, we strongly believe that further upgrades could be done focusing more on the sentiment detection task. In fact, the Stanford model we used is not able to predict accurately some statements. For example, the sentence *"I love being ignored"* is wrongly predicted as Positive.

We also think that further studies in considering also the parent comments in our approach could obtain greater results. However, we obtained state-of-the-art performances on the Sarcasm V2 Corpus and our Multitask BiLSTM model outperforms all the previous baselines that do not exploit user embeddings for the SARC dataset. Additionally, most of the prediction on tweets are coherent with reality, confirming the efficiency of our model.

We believe that our models could be used as baselines for new researches and we expect that enhancing them with contextual data, such as user embeddings, new state-of-the-art performances can be reached.

CITED LITERATURE

1. Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*, 2016.

2. David Bamman and Noah A Smith. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

3. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

4. John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

5. Jason Brownlee. Cnn long short-term memory networks. `https://machinelearningmastery.com/cnn-long-short-term-memory-networks/`, 2017. [Online; Accessed 22/07/2019].

6. Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

7. Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM, 2009.

8. Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*, 2017.

9. Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. Structural scaffolds for citation intent classification in scientific publications. In *NAACL-HLT*, 2019.

10. Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on*

CITED LITERATURE (continued)

computational natural language learning, pages 107–116. Association for Computational Linguistics, 2010.

11. John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

12. Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

13. Aniruddha Ghosh and Tony Veale. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 161–169, 2016.

14. Aniruddha Ghosh and Tony Veale. Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 482–491, 2017.

15. Raymond W Gibbs. On the psycholinguistics of sarcasm. *Journal of Experimental Psychology: General*, 115(1):3, 1986.

16. Raymond W Gibbs Jr, Raymond W Gibbs, and Herbert L Colston. *Irony in language and thought: A cognitive science reader*. Psychology Press, 2007.

17. Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, pages 581–586. Association for Computational Linguistics, 2011.

18. Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. Cascade: Contextual sarcasm detection in online discussion forums. *arXiv preprint arXiv:1805.06413*, 2018.

19. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

20. Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Associa-*

CITED LITERATURE (continued)

tion for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 757–762, 2015.

21. Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection? *arXiv preprint arXiv:1610.00883*, 2016.

22. Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. Your sentiment precedes you: Using an authors historical tweets to predict sarcasm. In *Proceedings of the 6th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 25–30, 2015.

23. Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm. *CoRR*, abs/1704.05579, 2017.

24. Roger J Kreuz and Gina M Caucci. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4. Association for Computational Linguistics, 2007.

25. Roger J Kreuz and Sam Glucksberg. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of experimental psychology: General*, 118(4):374, 1989.

26. Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

27. Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017.

28. Stephanie Lukin and Marilyn Walker. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. *arXiv preprint arXiv:1708.08572*, 2017.

29. Navonil Majumder, Soujanya Poria, Haiyun Peng, Niyati Chhaya, Erik Cambria, and Alexander F. Gelbukh. Sentiment and sarcasm classification with multitask learning. *CoRR*, abs/1901.08014, 2019.

30. Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

CITED LITERATURE (continued)

31. Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. Predicting readers' sarcasm understandability by modeling gaze behavior. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

32. Douglas Colin Muecke. *Irony and the Ironic*. Routledge, 2017.

33. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

34. Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. Creating and characterizing a diverse corpus of sarcasm in dialogue. *arXiv preprint arXiv:1709.05404*, 2017.

35. Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

36. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

37. Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*, 2016.

38. Tomáš Ptáček, Ivan Habernal, and Jun Hong. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, 2014.

39. Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM, 2015.

40. Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, 2013.

CITED LITERATURE (continued)

41. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

42. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

43. Frank Stringfellow Jr. *Meaning of Irony, The: A Psychoanalytic Investigation*. SUNY Press, 1994.

44. Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*, 2018.

45. Joseph Tepperman, David Traum, and Shrikanth Narayanan. " yeah right": Sarcasm recognition for spoken dialogue systems. In *Ninth International Conference on Spoken Language Processing*, 2006.

46. Oren Tsur, Dmitry Davidov, and Ari Rappoport. Icwsma great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

47. Akira Utsumi. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics*, 32(12):1777–1806, 2000.

48. Byron C Wallace, Laura Kertz, Eugene Charniak, et al. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516, 2014.

49. Meishan Zhang, Yue Zhang, and Guohong Fu. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, The 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460, 2016.

50. Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.