uc3m | Universidad Carlos III de Madrid

Dual Degree on Computer Engineering and Business Administration

2018-2019

Final Degree Project [TFG]

"Sentiment Analysis in Unstructured Textual Information with Deep Learning"

Alberto García Hernández

Tutor

Miguel Ángel Patricio Guisado

Madrid, 17th June 2019

Summary

This document analyses the current State-of-the-Art algorithms in the fields of Natural Language Processing and Sentiment Analysis. It continues with a step-by-step explication of the development process of pre-processing techniques and neural networks architectures that allow to perform sentiment predictions (predicting rating stars) on Amazon.com customer reviews. An accuracy comparison has been made between 4 different models to check their performance.

The second part of the project has been the development of a demo web application to show the potential of a Product Analytics Tool, which allows to perform sentiment predictions of any product on Amazon website. This app scrapes the reviews, loads the previously trained model and makes the predictions, generating different insights such as the most positive and negative features of the product based exclusively on the most reliable and objective data, customer reviews. The source code of the app can be found here:

https://github.com/albergar2/SA_Project

At the end of the document an appendix has been added providing information and estimates of the cost and tasks required to replicate this project in a professional environment.

Key Words:

- Sentiment Analysis
- Deep Learning
- Amazon reviews
- Feature extraction
- Web application.

Index

Figures Index

Section 1

Introduction

1.1 Contextualization

Sentiment Analysis is a field within Natural Language Processing (NLP) which focuses on identifying and extracting opinions from an unstructured text. According to the studies developed by Anant Jhingrans from IBM Research (Jhingrans, A. 2018), nearly 85% of the world's data is unstructured (not organized in a determined manner). Advanced Sentiment Analysis techniques are based on Machine Learning and Deep Learning algorithms that help companies to make sense of their data, to extract actionable insights, therefore making teams more competent and automating different business processes. Sentiment Analysis can be understood as a text mining technique used to identify and extract subjective insights from unstructured text to help businesses or other organizations to monitor the social sentiment related to their products, brand or services.

The main research about Sentiment Analysis revolves around polarity classification, which aims to classify a sentence's opinion as expressing positive, negative or neutral emotions. Depending on the nature of the problem, the algorithms can be applied at a different level of scope: document, sentence or sub-sentence level.

Research about Sentiment Analysis has experienced a good amount of progress for the last 10 years and algorithms and its text analysis' ability has improved remarkably. So far, researchers on the field have been focusing on five types of analysis:

- Polarity Classification: positive, neutral and negative emotions.
- Fine-Grained Classification: very positive, positive, neutral, negative and very negative emotions.
- Emotions detection: happiness, frustration, anger, sadness, excitement, etc.
- Aspect-Based: features about the concept the sentiments refer to.
- Intent: detecting what people mean rather than what people say.

Despite of the progress on the field, there are some persistent limitations that have not been solved yet and researchers still struggling with: Subjectivity and Tone, Context and Polarity, Irony and Sarcasm, Emojis and Defining Neutral (objective texts, irrelevant information, wishes).

Focusing on a practical approach, the main applications of Sentiment Analysis from which companies and users can benefit from, are:

- Social Media Monitoring: analyze people's reactions around a topic.
- Brand Monitoring: analyze mentions from different sources like social media, news, blogs or forums to obtain a broad view of the brand.
- Customer Feedback: Net Promoter Score (NPS) classifies individuals as promoters, passives or detractors. Sentiment Analysis can aggregate NPS and other surveys in order to respond quicker to customers' shifts.
- Customer Support: detect disgruntled customers and surface those tickets to the top, gaining deep insight into what's happening.
- Workforce Analytics and Voice of Employee: discover employee concerns from surveys, keywords and behavior, ensuring they feel heard and valued.
- Product Analytics: analyze large quantities of product feedback surveys, social media and online mentions about a product, keeping constant tabs on the features people like and don't like about the product itself.
- Market Research and Analytics: analyze product reviews vs competitors reviews, generate weekly/monthly/daily reports, analyze market reports and business journals for trends, social media analysis for real-time happenings.

The different applications identified above are fields in progress at the moment, getting useful insights from customer reviews, tracking the advertisement's influence on purchasing effect as well as real-time analysis on social media are just a few tools that are going to improve a lot in the years to come. Understanding real-time perception from customers with a high accuracy score will be taken from granted in the future by any company. This document has focused on Product Analytics, the most promising area of study oriented to businesses as it's development will be able to reproduce human level accuracy on millions of customer interactions about a company's product or brand by analyzing user's sentiment and extracting the product's main features they are talking about. The key steps on performing Product Analytics are:

1. Identify product's features and aspects the customers care about.
2. Predict users' intentions, sentiments and reactions about these features.

1.2 Goals and objectives

The main goal this project pursues is to obtain a Sentiment Analysis model trained with unstructured text using deep learning techniques to solve a polarity classification problem, extending the scope to a simple aspect-based sentiment analysis, which in combination will produce a Product Analytics web application. The model has been trained with a dataset built upon 4 million Amazon.com products' reviews obtained from the research and work of Xiang Zhang, PhD student at New York University (X. Zhang's Google Drive, 2019).

The aim of the model is to achieve an accuracy value on the test dataset close to State-of-the-Art, using current methodologies and deep learning algorithms. Once the model has been successfully trained, it will be loaded into a user-friendly demo web app that will allow the user to perform a sentiment analysis on any Amazon.com product by introducing its URL link, showing the great potential of a Product Analytics deep learning platform.

1.3 Legal Framework

This section examines the key aspects that must be taken into consideration when developing a project as the one described in this document. It's important to revise the current legislation related to the software licenses and the General Data Protection Regulation (GDPR). As far as software licenses, the training section of the project has been developed on Kaggle servers, owned by Google and free to use. The web app has been developed using PyCharm Free Community Edition IDE and Dash by Plotly Framework, both free to use.

The dataset used was obtained from Xiang Zhang's Google Drive, which was open to the public. On the other hand, the web app is loading customer reviews from any product from Amazon.com, however, the method used to do this is basically loading each review's web page and reading the HTML file, therefore data which is open and not restricted by GDPR regulation since it's not personal data (such as name, IP, email, phone number, etc).

1.4 Socio-Economic Environment

The field of Sentiment Analysis jointly with NLP and Artificial Intelligence are main research topics both in academia and industry. These topics will evolve in the next few years, integrating new languages as well as new algorithms and modified neural networks that will increase the accuracy of the current methods.

The results achieved in this project could be used on the private sector for multiple applications. The most straightforward tool the industry will benefit from will be a Product Analytics tool similar to the one developed for this project to extract the key product's features as well as the customer sentiment from product reviews. On the other hand, there are multiple variations of this tool that can be oriented to develop tools able to predict sentiments on customer' feedback to improve NPS scores or Social Media Monitoring to track customer reaction's to marketing campaigns, releases of new products or news about the company's brand or industry.

Solutions as the ones identified above could make a competitive strength for a company by improving the service given to customers and being flexible to adapt its products and marketing campaigns to respond as fast as possible to changes in trends on the industry. According to Domo's Data Never Sleeps 6.0 report (Domo, 2018), in 2020, on average each person in the planet will generate 1.7MB of data per second, therefore companies investing in technology to get insights of this unstructured and low-quality data will obtain a significant advantage over their competitors and will increase its revenues.

On the other hand, there are social applications of this technology, being an example of it the ones implemented on social media. Companies like Facebook and Twitter analyze every post looking for inappropriate content that enacts violence, drug abuse or pornography, in order to censor it and stop it's spread over the internet. These approaches are not related to revenues but to social wellness.

1.5 Document structure

This document will start with an overview of Sentiment Analysis' State-of-the-Art to explore the current techniques, approaches and accuracies obtained from the world's best researchers on the field of NLP. Once the current trends and techniques to solve similar problems have been identified, the document will explain and justify the algorithms used and decisions taken to develop the goal solution. On the following sections a results analysis focused on the accuracy and the perform of the output from the web app will be showed. A final section placed at the Annexes analyzes the project implementation costs and tasks in case a company wants to build a similar Product Analytics tool. As a conclusion, the document will recap the results described on this document and state future work guidelines to anyone interested in working on the field and improving the solutions described.

Section 2

State-of-the-Art

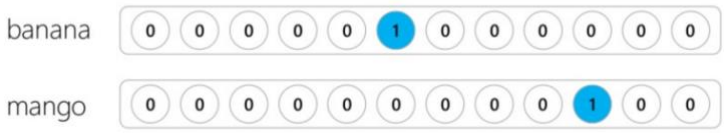
Current research trends on sentiment analysis rely on unsupervised algorithms to perform feature extraction, using pre-trained word embeddings as the most common approach. Text pre-processing is usually more important than the neural network used on terms of accuracy.

2.1 Word Embeddings

At the moment, computers are not able to understand the meaning and concepts behind words. Word vectors are the standard approach to represent text in a way a computer can use it, mapping words and phrases to vectors of real numbers, which constitutes the previous step to the training stage as neural networks must work with numbers rather than words.

Traditional word vectors try to capture the appearance frequency for each word on each document. Bag of Words (BoW) links each word given in the vocabulary to a one-hot encoded vector depending on whether the word appears on the document. This technique doesn't encode any information related to the actual meaning of the word and gives the same weight to every word, even though it is proved that in most NLP problems there are some words which are more relevant than others.

Figure 1: BoW vector representation

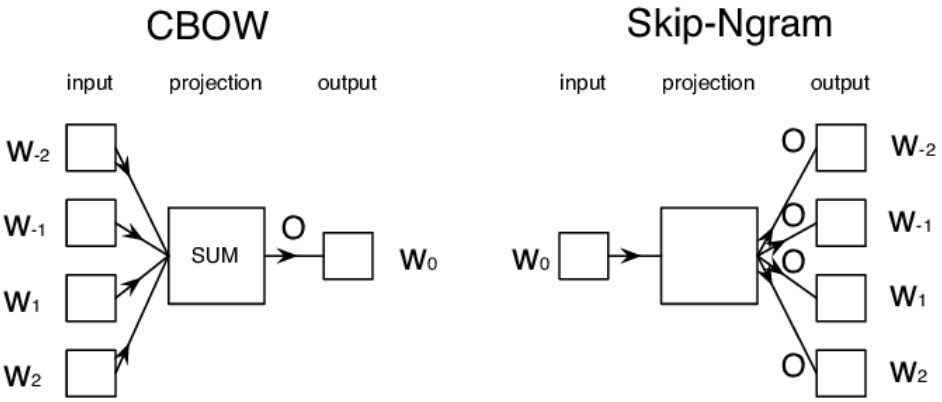


Source: D. Karani (2018)

Neural embeddings are the next step on the evolution of word embeddings. Word2Vec is a predictive model developed by Google (Google Code Archive, 2019) which have been trained using English Wikipedia Corpus to predict word vectors representation and to capture the meaning of the word by taking into consideration surrounding words. This predictive model can use two different architectures:

- Continuous bag-of-word (CBOW): the model is trained with the surrounding words and it aims to predict the given word.
- Skip-Gram: the model is trained with single words and aims to predict its surroundings.

Figure 2: Neural Embeddings Architectures – CBOW vs Skip-Gram



Source: D. Karani (2018)

The real alternative to Word2Vec is GloVe (Pennington et al., 2014), a word embedding developed by Stanford which uses a count-based model (instead of predictive model as Word2Vec) and a neural methodology to break down the co-occurrence counts matrix by doing a dimensionality reduction, therefore building more expressive and dense word vectors. It has been proved that GloVe is faster to train than Word2Vec, but researchers haven't found any relevant differences on results.

Neural word embeddings are able to include into their vectors most of the information related to the vocabulary words, the most frequent surrounding words, meanings, synonyms or antonyms by taking into consideration from 50 to 300 dimensions for each of word in the vocabulary. As a result, we can perform operations with words similar to those done with numbers. For example, given three words *king*, *man* and *woman*, we could perform the following operation using their word vectors, and we will obtain a new word:

$$(king - man) + woman = queen$$

Similar words (even if their relationship is that they are antonyms of each other) will be placed into a space from 50 to 300 dimensions close to each other, on contrast to radically different words with nothing in common which will be placed far away from each other. Google has developed a web app to explore word embeddings trained by Google's Wor2Vec team that can be found on the following link:

http://projector.tensorflow.org/

Continuing with the example about the word *king*, the example below shows the results obtained from Word2Vec and Google's web app for word embeddings on representing the closest 50 neighbors. Since each neighboring word has 300 dimensions, a PCA has been used to reduce their dimensionality into a 2D visual representation. The example shows that the closest words to *king* are *kingdom*, *battle*, *prince*, *crown* and *regent*. It is notable how Word2Vec has captured words' closeness on south region (English rulers), east region (roman numbers used to identify kings who have chosen the same name) and west region (family kinship).

Figure 3: *King* Word2Vec example

Source: Projector Tensorflow. Embedding Projector – Visualization of high dimensional data

2.2 State-of-the-art datasets

In order to analyze the results obtained by different techniques, a replicable environment built upon benchmark datasets is compulsory. At the moment, there are different State-of-the-art datasets with different granularity used to compare the performance between different algorithms.

- Stanford Sentiment (Socher et al., 2013): movies reviews from IMDB which contain 5 levels of sentiments (SST-Fine / SST-1) (strong negative, negative, neutral, positive and strong positive). A binary sentiment version is also available (SST-Binary / SST-2).
- OpeNER (Agerri et al., 2013): hotel reviews and opinions, which include annotated sentiment holders, targets and phrases.
- SenTube (Uryupina et al., 2014): comments extracted from YouTube about tablets (SenTube-T) and automobiles (SenTube-A), usually from commercial videos. They have been annotated for negative, neutral and positive.
- SemEval (Nakov et al., 2013): tweets collected in 2013 and annotated on three levels of sentiment.

2.3 Related experimental results

The following table developed by Barnes et al. (2016), highlights the State-of-the-Art accuracy on sentiment analysis' task on every dataset stated above. Barnes and his colleagues compared the accuracy of different algorithms across the set of datasets on word embeddings' dimensions ranging from 50 to 600. Marked in bold are the best performing experiment for each dataset, while marked on blue are the results reported by other researchers. The last column shows the average across all the dataset for every algorithm and dimensionality combination, stating that as overall, BiLSTM is the most successful.

Figure 4: Algorithm accuracy comparison across state-of-the-art datasets

	Model	Dim.	SST-fine	SST-binary	OpeNER	SenTube-A	SenTube-T	SemEval	Macro-Avg.
Baselines	BOW		40.3	80.7	77.1[4]	60.6[5]	66.0[5]	65.5	65.0
	AVE	50	38.9	74.1	59.5	62.0	61.7	58.1	59.0
		100	39.7	76.7	67.2	61.5	61.8	58.8	60.9
		200	40.7	78.2	69.3	60.6	62.8	61.1	62.1
		300	41.6	80.3[3]	76.3	61.5	64.3	63.6	64.6
		600	40.6	79.1	77.0	56.4	62.9	61.8	63.0
State-of-the-Art Methods	RETROFIT	50	39.2	75.3	63.9	60.6	62.3	58.1	59.9
		100	39.7	76.7	70.0	61.4	62.8	59.5	61.7
		200	41.8	78.3	73.5	60.0	63.2	61.2	63.0
		300	42.2	81.2[3]	75.9	61.7	63.6	61.8	64.4
		600	42.9	81.1	78.3	60.0	65.5	62.4	65.0
	JOINT	50	35.8	70.6	72.9	65.1	**68.1**	66.8[6]	63.2
		100	34.3	70.8	67.0	64.3	66.4	60.1	60.5
		200	33.7	72.3	68.6	**66.2**	66.6	58.4	61.0
		300	36.0	71.6	70.1	64.7	67.6	60.8	61.8
		600	36.9	74.0	75.8	63.7	64.2	60.9	62.6
	LSTM	50	43.3 (1.0)	80.5 (0.4)	81.1 (0.4)	58.9 (0.8)	63.4 (3.1)	63.9 (1.7)	65.2 (1.2)
		100	44.1 (0.8)	79.5 (0.6)	82.4 (0.5)	58.9 (1.1)	63.1 (0.4)	67.3 (1.1)	65.9 (0.7)
		200	44.1 (1.6)	80.9 (0.6)	82.0 (0.6)	58.6 (0.6)	65.2 (1.6)	66.8 (1.3)	66.3 (1.1)
		300	45.3[1] (1.9)	81.7[1] (0.7)	82.3 (0.6)	57.4 (1.3)	63.6 (0.7)	67.6 (0.6)	66.3 (1.0)
		600	44.5 (1.4)	**83.1** (0.9)	81.2 (0.8)	57.4 (1.1)	65.7 (1.2)	67.5 (0.7)	66.5 (1)
	BILSTM	50	43.6 (1.2)	82.9 (0.7)	79.2 (0.8)	59.5 (1.1)	65.6 (1.2)	64.3 (1.2)	65.9 (1.0)
		100	43.8 (1.1)	79.8 (1.0)	82.4 (0.6)	58.6 (0.8)	66.4 (1.4)	65.2 (0.6)	66.0 (0.9)
		200	44.0 (0.9)	80.1 (0.6)	81.7 (0.5)	58.9 (0.3)	63.3 (1.0)	66.4 (0.3)	65.7 (0.6)
		300	**45.6**[1] (1.6)	82.6[1] (0.7)	**82.5** (0.6)	59.3 (1.0)	66.2 (1.5)	65.1 (0.9)	**66.9** (1.1)
		600	43.2 (1.1)	83 (0.4)	81.5 (0.5)	59.2 (1.6)	66.4 (1.1)	**68.5** (0.7)	**66.9** (0.9)
	CNN	50	39.9 (0.7)	81.7 (0.3)	80.0 (0.9)	55.2 (0.7)	57.4 (3.1)	65.7 (1.0)	63.3 (1.1)
		100	40.1 (1.0)	81.6 (0.5)	79.5 (0.9)	56.0 (2.2)	61.5 (1.1)	64.2 (0.8)	63.8 (1.1)
		200	39.1 (1.1)	80.7 (0.4)	79.8 (0.7)	56.3 (1.8)	64.1 (1.1)	65.3 (0.8)	64.2 (1.0)
		300	39.8[2] (0.7)	81.3[2] (1.1)	80.3 (0.9)	57.3 (0.5)	62.1 (1.0)	63.5 (1.3)	64.0 (0.9)
		600	40.7 (2.6)	82.7 (1.2)	79.2 (1.4)	56.6 (0.6)	61.3 (2)	65.9 (1.8)	64.4 (1.5)

Source: (Barnes et al., 2016)

2.3.1 Barnes et al. (2016) Experiment: Models

1. BOW: regularized logistic regression on a Bag of Words representation (count-based model) of dimensionality ranging from 50 to 600. This approach is commonly used as standard baseline on text classification tasks. Marked in blue are the results reported by Uryupina et al. (2014) on SenTube and Lambert (2015) on OpeNER.

2. AVE: regularized logistic regression classifier performed over the average of word embeddings vectors trained with skip-gram algorithm (Mikolov et al., 2013). Best results have been reported by Faruqui et al. (2015) on SST-binary.

3. RETROFIT: regularized logistic regression on average word embeddings vectors from Mikolov et al. (2013) in a combination with PPDB-XL lexicon, a lexicon database that helps NLP algorithms to be more robust against language expressions'' variability. This approach was taken by Faruqui et al. (2015) who improved AVE's results on SST-binary.

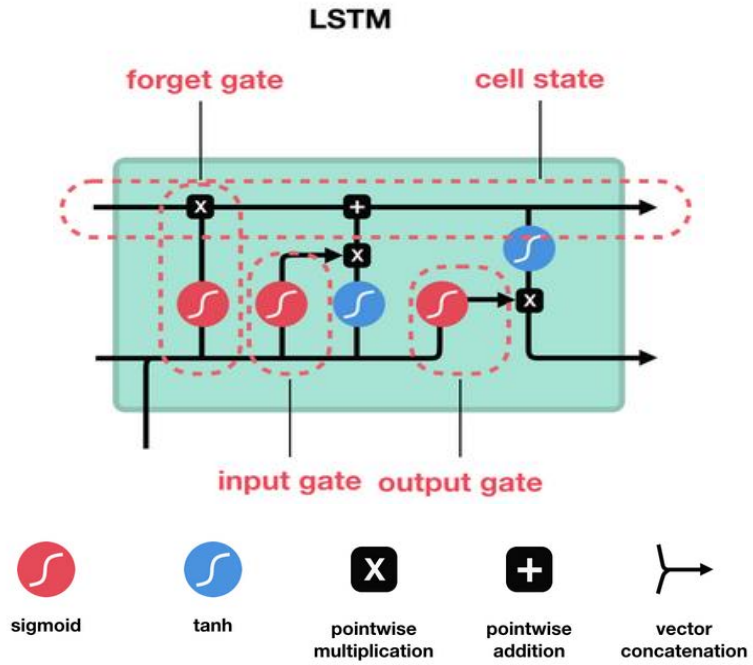
4. JOINT: this approach was performed by Tang et al. (2014) with word embeddings trained on sentimental meaning rather than the original semantic meaning from Mikolov et al. (2013). Concatenating the minimum, maximum and average sentimental word embedding vectors for each sentence a training a linear SVM (Support Vector Machine – a non-probabilistic binary linear classifier) on these, Tang et al. (2014) obtained the best result on SemEval dataset.

5. LSTM (Long Short-Time Memory): this algorithm was first introduced by Hochreiter and Schmidhuber (1997) and it is a kind of Recurrent Neural Network. As on every model, words are transformed into word embeddings vectors through an embedding layer. Then each word vector is fit into the network by the *input gate* on the *cell state*, which is the belt that goes through the LSTM neuron, modifying the vector as it advances through different regulated gates (*input gate, output gate, forget gate*) which add or subtract information of the cell. Each word will modify in a different amount the cell state, which gives this kind of networks the name of *recurrent*, as they take into consideration previous words in the sentence to learn to predict the output. Best results have been obtained by Tai et al. (2015) on SST-fine and an improved version trained on 600 dimensions rather than 300 as Tai et al. (2015) did by Barnes et al. (2016).

6. BiLSTM (Bidirectional Long Short-Term Memory): designed over the same architecture than LSTM, the main feature is that two LSTM networks are run and their outputs concatenated. The first LSTM network will receive the original sequence of word embeddings vectors, while the second LSTM network will receive the input on inverse order, therefore the model will be more concerned about the sentimental meaning of the sentence, generally improving LSTM results. BiLSTM has established State-of-the-Art accuracy values on several

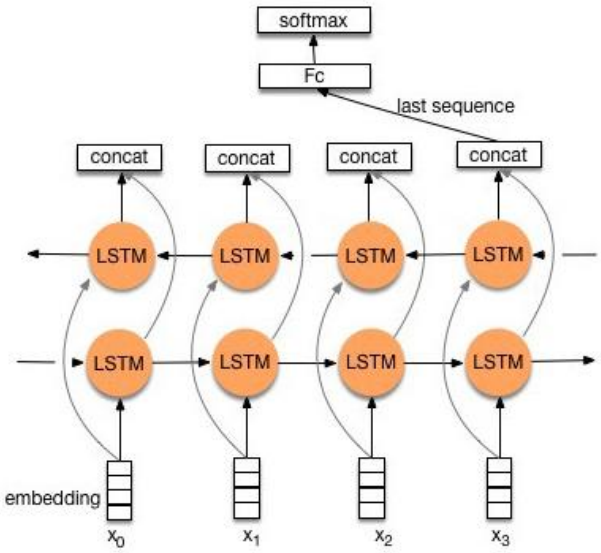
datasets and has been ranked overall as the best network over the full set of datasets by Barnes et al. (2016). This network has achieved overall best results on SST-fine and SST-binary by Tai et al. (2015), and on OpeNER and SemEval by Barnes et al. (2016).

Figure 5: LSTM Diagram



Source: M. Nguyen (2018)

Figure 6: BiLSTM Diagram



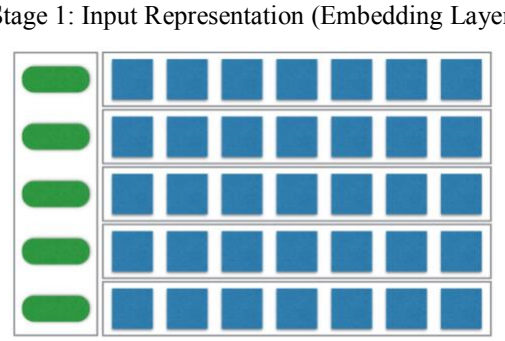
Source: Ceshine Lee (2017)

7. CNN (Convolutional Neural Network): CNN are feature extractors from a given input focused on the fields of Computer Vision and Image Classification. However, in recent years they have been adapted to one dimension vectors, being capable to solve NLP tasks. CNN models take word embedding's vectors to perform convolutions over the sentence's words in order to learn whether the word is meaningful for the task. One of the biggest weakness of CNN when applied to NLP is their lack to take into consideration the surroundings and semantic meaning of text (where LSTM and BiLSTM outperforms). Best results following this approach have been obtained by Kim (2014).

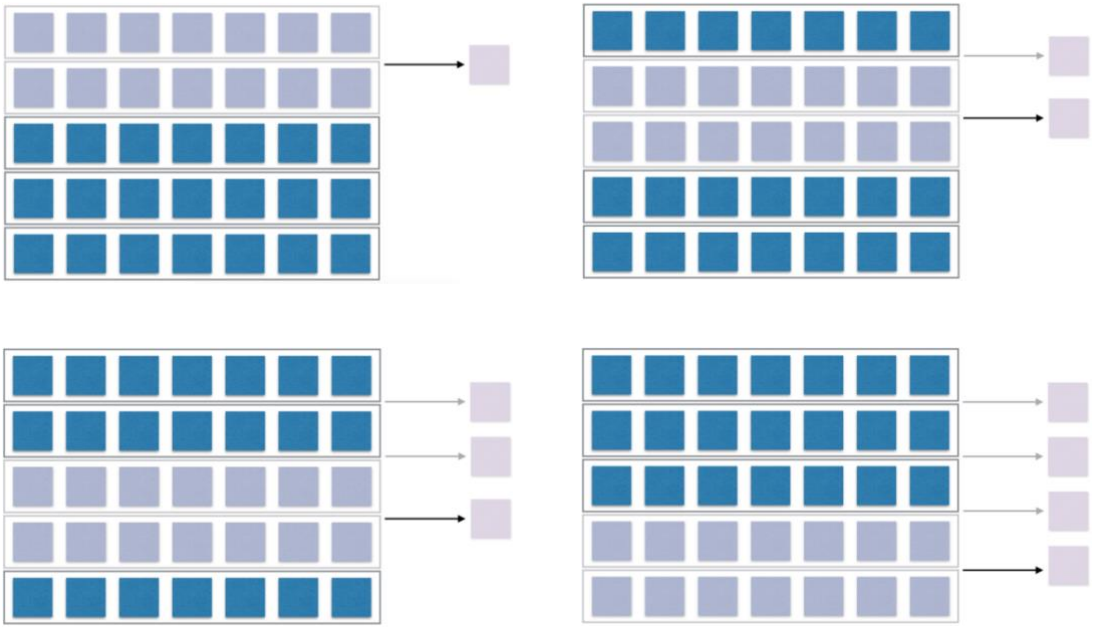
   The following image illustrates these concepts where a window size of 2 has been chosen for simplicity (convolution kernel).

   - Stage 1: original words are represented in green, which are transformed by the embedding layer into vectors represented in blue squares.
   - Stage 2: taking the first two words' vectors (window size) the network performs the multiplication of vectors' weights to obtain an output. This process is repeated with the next word until every word of the sentence has been convoluted. A CNN network has to specify how many neurons the network has (filters) and each of them will perform similar convolutions for each sentence. At the end of the convolution process, the output will be a one dimensional array of length *number of words* for each of the filters.
   - Stage 3: for each array obtained, a *max-pooling* process will be performed to reduce its dimensionality by a factor determined by a parameter, generating an output that can be introduced into a function to obtain a final prediction.

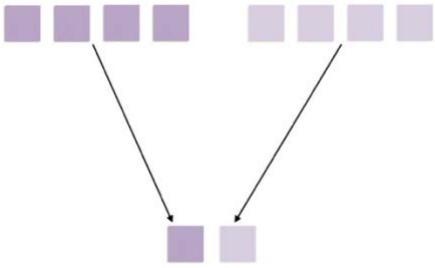
Figure 7: Convolutional Neural Network Workflow



Stage 2: Filter Convolution



Stage 3: Max-Pooling



Source: Debajyoti Datta, Data Scientist at X.ai

2.3.2 Advanced Models

Both NLP and Sentiment Analysis are evolving really fast on recent years, and year after year researchers manage to improve State-of-the-Art accuracy with complex configurations, redesigned algorithms and different combinations of them. These advanced models are usually tested on SST-fine and SST-binary as both datasets have highlighted over the others during the last years. On the figure below are illustrated the results obtained by several researchers doing some modifications to the networks and algorithms explained in the previous section. This comparison has been developed by Young et al. (2018).

Figure 8: Algorithm accuracy comparison on Stanford Sentiment Treebank

Paper	Model	SST-1	SST-2
Socher et al (2013)	Recursive Neural Tensor Network	45.7%	85.4%
Kim (2014)	Multichannel CNN	47.4%	88.1%
Kalchbrenner et al (2014)	DCNN with k-max pooling	48.5%	86.8%
Le and Mikolov (2014)	Paragraph vector	48.7%	87.8%
Tai et al (2015)	Constituency Tree-LSTM	51.0%	88.0%
Yu et al (2017)	**Tree-LSTM with refined word embeddings**	**54.0%**	**90.3%**
Yu et al (2017)	Bi-LSTM with refined word embeddings	49.7%	88.6%
Yu et al (2017)	CNN with refined word embeddings	48.8%	87.9%
Chen et al (2017)	BiLSTM + CNN	48.5%	88.3%

Source: (Young et al., 2018).

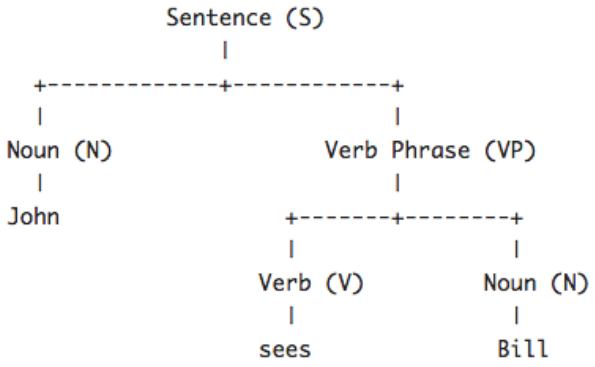
These models have performed complex transformations, additions or combinations to obtain State-of-the-Art accuracy values and it is out of the scope of this document to make a deep analysis of each of them, however, a high level clarification of some of them has been done on this document:

- Socher et al. (2013) and Tai et al. (2015) both developed recursive networks relying on constituency-based parse trees that represent the syntactic structure of the sentence following a phrase structure grammar. The difference between both is the sentence modelling and preprocessing, remarking the importance of this steps. Tree structures performed much better than linear LSTM showed on the previous section, which implies that tress structures capture better the syntactical features of sentences.

- Yu et al. (2017) established the State-of-the-Art by proposing a refined word embedding vectors combined with sentiment lexicons on a Tree LSTM structure. They also developed similar refinement to BiLSTM and CNN networks, achieving great improvement.
- Kim (2014) and Kalchbrenner et al. (2014) proposed a CNN approach where the former is a modified version of its own previous work and the last is a sequence model with interweaving convolutional a max-pooling layers.

- Chen et al. (2017) opted for a sequence model of a BiLSTM layer combined with a CNN layer. Performing a simple refinement this approach scored high on the State-of-the-Art ranking.

Figure 9: Consistency Parse tree



Source: Sebastian Ruder, NLP Progress

To conclude this section, it is notable the improvement related to tree structures as well as preprocessing and refinement techniques, that can lead to substantial improvements. Neural networks are unquestionably the choice to make on Sentiment Analysis tasks focusing the attention on different variations of LSTM, BiLSTM and CNN networks.

Section 3

Research and development

3.1 Data Exploration

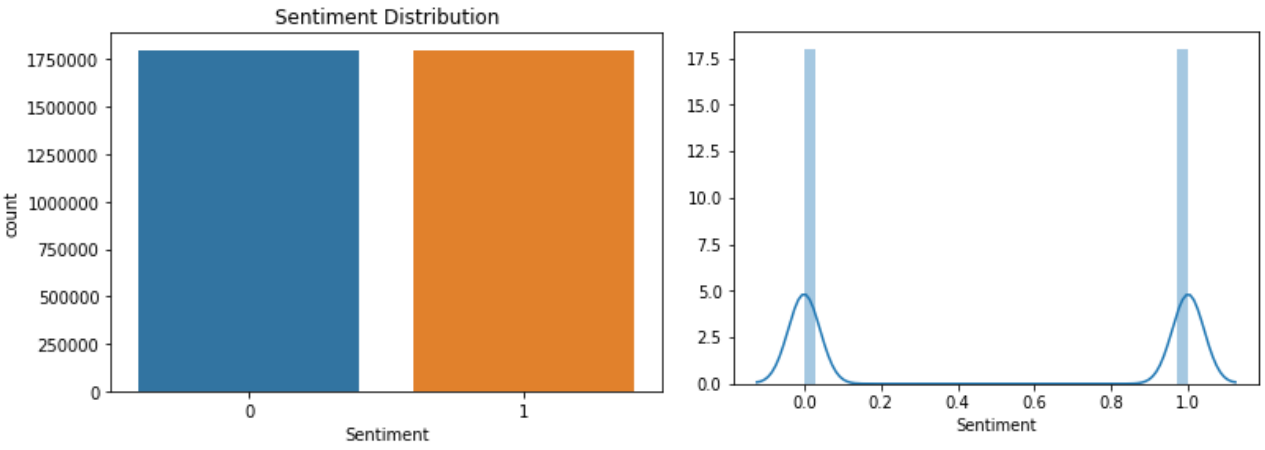
As a good practice for any data science project, a data exploration has been performed to get to know more information about the dataset as well as extracting insights and distributions. The full dataset contains 3.60 million reviews for training and 400k reviews for testing, labelled as positive or negative. These reviews come from different Amazon's products, being labelled negative those with 1 and 2 stars and positive those with 4 and 5 stars. As a result, every review with 3 stars, which highlights by its neutrality, has been omitted. The following figure shows the structure of the dataset.

Figure 10: Dataset's structure overview

	Sentiment	Title	Review
0	1	Stuning even for the non-gamer	This sound track was beautiful! It paints the ...
1	1	Excellent Soundtrack	I truly like this soundtrack and I enjoy video...
2	0	Buyer beware	This is a self-published book, and if you want...
3	0	mens ultrasheer	This model may be ok for sedentary types, but ...
4	1	More great playing	Larry's work for the Muse label in the late 80...

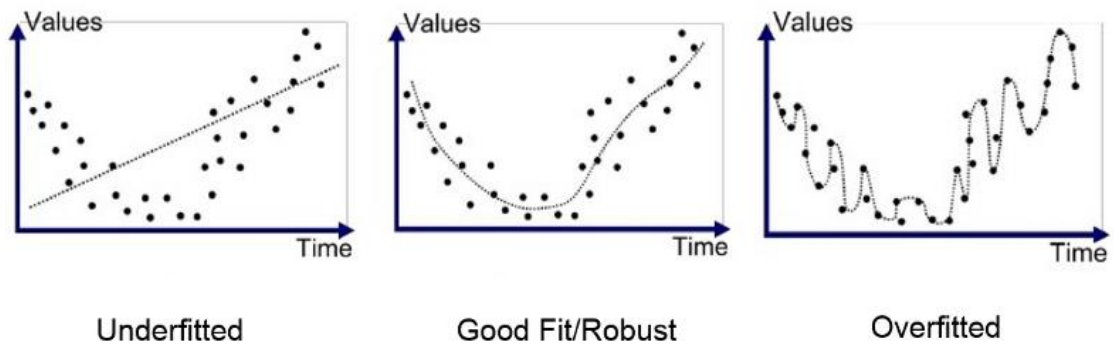
It is very important to know whether the dataset's target label is balanced, to avoid training the model with this kind of bias. The figure below illustrates the label distribution across the dataset, resulting in a perfectly balanced set ready to work with.

Figure 11: Sentiment Distribution



Hidden correlations between variables and the target label can end up generating an overfitting problem. Overfitting arises when during the training process, the model adjusts too much to the training dataset instead of generalizing for any data, which leads to underperform on the test dataset. Let's make the hypothesis that any review on the training dataset longer than 50 words is negative. During the training process the model will realize about this and stop learning about the word embeddings or sentiments and just focusing on the review's length. Once the training has been completed, the model must be tested with unseen data, which now doesn't have this hidden correlation (there is no link between negative reviews and review's length). The model will just check the review's length to make predictions, that won't be accurate given the circumstances.

Figure 12: Overfitting example



Source: A. Bahande, 2018

In order to identify the existence of this kind of hidden correlations, a feature extraction has been done to check whether they are correlated with the target label.

- Review Length: count of the number of words in the review.
- Avg_word_length: average number of letters a word has on each review.
- Upper: count of the number of upper letters each review has.

Figure 13: Extracted Features Distributions

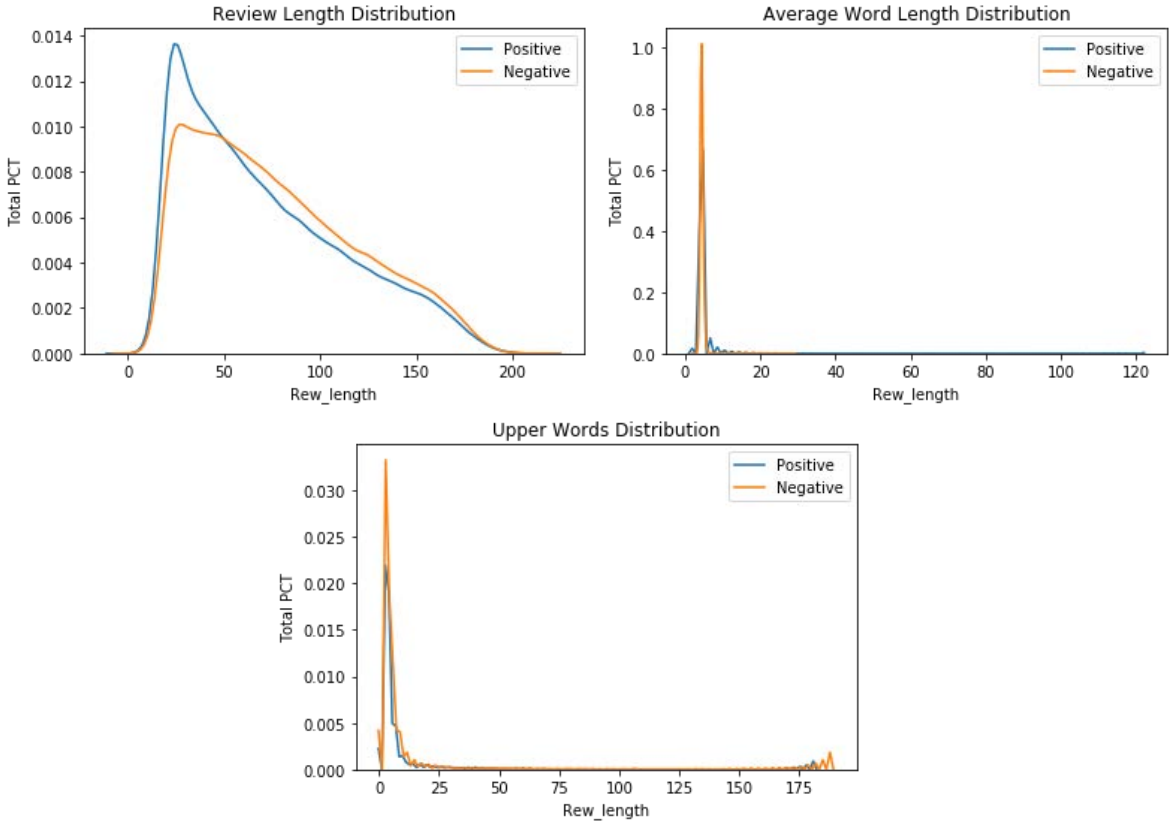


Figure 14: Extracted Features - Correlation Matrix

	Index	Sentiment	Review_length	Avg_word_length	Upper
0	Sentiment	1.000000	-0.072032	0.014307	-0.017613
1	Review_length	-0.072032	1.000000	0.005374	0.085556
2	Avg_word_length	0.014307	0.005374	1.000000	-0.012229
3	Upper	-0.017613	0.085556	-0.012229	1.000000

The strongest correlation with the Sentiment tag appears to be on the Review Length feature with a strength of -0.07 confirming the hypothesis stated before, longer reviews tend to be slightly more negative. This parameter must be taken into consideration when training the model to avoid overfitting.

3.2 Data Pre-Processing

Data pre-processing is crucial on NLP tasks since different approaches can lead to totally different results when trained the same neural network. Because all of these, usually cleaning and preprocessing is done manually and has to be adapted to the specific problem we aim to solve, meaning sometimes trial and error procedures. This subsection explains the cleaning and preprocessing techniques performed that led to the best results, leaving to the Results Analysis Section the empirical results on performance and accuracies.

Due to the size of the dataset and all the transformations and operations required to solve the problem, a regular laptop is not enough to run all within a reasonably time. As a result, the code implemented has been run on Kaggle framework which gives free access to Google's servers to run code with a compilation time limit of 9 hours. This limit has been overpassed in several occasions during the development of the project, so the code has been optimized to run as fast as possible and split in chunks that were reassembled later. The next figure summarizes the steps taken on the Cleaning and Pre-processing process.

Figure 15: Cleaning and Preprocessing Workflow

1. Simple Preprocessing: Once the data has been loaded in a tabular pandas dataframe and the review's title has been concatenated with the review itself, *gensim* library has been imported to have access to its *simple_preprocessing* function, which converts each review into a list of lowercase words in a pretty optimized way. *Gensim* library is focused on topic modelling (statistical models that extract the hidden topics in documents), statistical semantics and semantic structure analysis, areas of study that need too preprocessing techniques for text.

2. Stop-Words: the list of words is filtered to remove English stop-words, which are those words commonly used and that doesn't give any information ("*the", "a", "an", "in",* etc.*)*. This stop-words have been downloaded from *NLTK* (*Natural Language Tool Kit*), a library focused on NLP tasks that makes preprocessing linguistic data considerably easier by providing tools for text classification, tokenization, semantic reasoning, tagging, stemming and parsing. The existence of these tools is essential to make sense of word semantics and meanings and they have been key for researchers to focus their efforts mainly on English language.

3. POS-Tagging: a great optimization of the cleaning process presented on this document is to keep exclusively sentimentally meaningful syntactical categories. This means to drop any word that belongs to syntactical categories with no sentiment associated to them, such as determinants or pronouns. *NLTK* has a tool available called *Part of Speech Tagging* which identifies the syntactical category for each word on the sentence, allowing a filter process to keep the most meaningful categories:
    - Adjectives
    - Nouns
    - Adverbs
    - Verbs

4. Tokenizer: once the cleaning is done, the text must be manipulated to fit neural networks' requirements, which means transforming words into numbers with which they can operate. The first step is to tokenize each word by giving every different word a number that represents it unmistakably adding an extra category,

UNK (unknown), for words that may appear and are not on the tokenizer's index table. Words whose occurrence is lower than 5 times have been omitted to optimize the code and avoid training the model with non-frequent words. After that, every review is encoded looking up to the tokenizer's index table (vocabulary). Every neural network has a strong requirement that must be satisfied, each training example (reviews on this project) must have the same length. To achieve this, a padding technique has been used fixing the length of every sentence to the number calculated with the following formula:

Equation 1: Max Length Padding Formula

$$Max\ Length\ Padding = \bar{x} - Z \cdot \sigma$$

Where $\bar{x}$ is the average reviews' word length, σ is its standard deviation and Z is the Z's score for the desired percentile under a normal distribution

Assuming the sentence length across the dataset follows a normal distribution, the *z-value* will fix the percentile that determines the reviews that will be fully considered. To give an example, a *z-value* of for the 80% percentile will result in a *max_length* value great enough to embrace 80% of all reviews, which will have a review's length lower or equal than *max_length*. Reviews with a lower length will be filled with zeros at the end to fit the *max_length* variable. The remaining 20% of reviews will be truncated to *max_length*, losing some information.

5. At this moment, words have been mapped to numbers, but each word is not related by any mean with any other word. Pre-trained neural word embeddings have been used to transform those tokens into meaningful tokens, leaning towards a 100 dimensions of GloVe since is more optimized and runs faster than Word2Vec. Therefore, each word token will be transformed into a 100 dimensions word vector that will be given as input with the other vectors of the review to the first layer of the neural network.

3.3 Neural Networks Implemented

Once data has been cleaned and preprocessed it is ready to be fitted into a neural network model. In order to choose an appropriate model that will lead to good results a four model comparison has been made choosing vanilla versions of LSTM, BiLSTM and CNN as well as a modified CNN version that without increasing its complexity has improved the accuracy of vanilla CNN. The election of the following architectures has been done to compare the performance of vanilla networks similar to those developed by Tai et al. (2015) and Kim (2014). Despite of its proven performance, tree structures' networks have not been tested since there wasn't a stable version of Tensorflow or substitutes capable of dealing with specific architectures at the moment this project was developed.

All the models below have been implemented using Keras Library over a TensorFlow build and run on Kaggle servers.

3.3.1 Long Short-Term Memory (LSTM)

This model and the following ones have been defined on Keras, taking advantage of its Sequential model API which allows to build complex models by adding and connecting different layers with each other, obtaining an understandable and fast workflow ready to be trained and deployed.

The first layer for every model is the Embedding layer, which will be responsible for building the dense vectors that will be passed to the following layers. This layer is usually initialized by random weights, learning over the training process to identify the actual relationships between words. However, it is a common practice to used pretrained word embeddings to speed up the training process. As explained above, pretrained GloVe word embedding's weights have been loaded in order to start the training process with a complete understanding of the semantic relationships between words. This layer has the trainable flag switched on, meaning these weights will be modified to capture not only

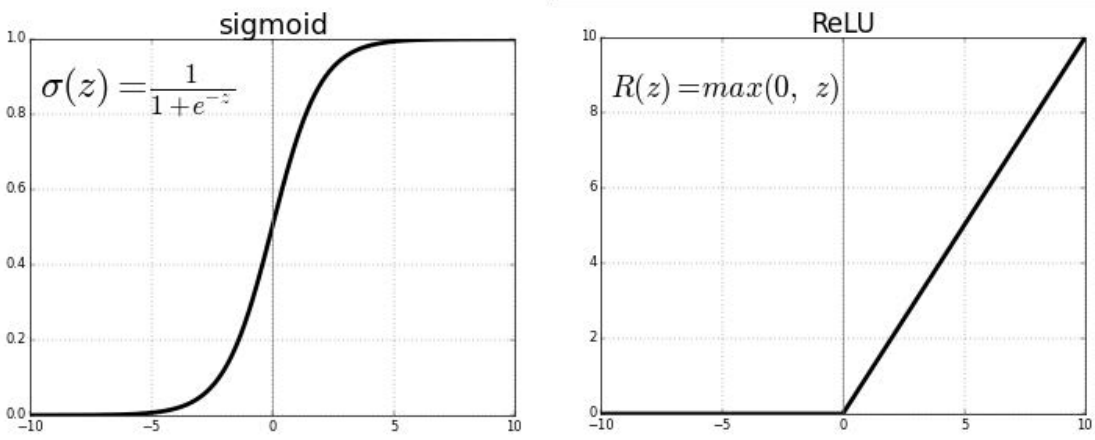
the semantic meaning but the sentimental relationships between words and semantic structures.

The output of the previous embedding layer, which has a dimensionality of 100 since GloVe's weights loaded had 100 dimensions, will be fitted into a LSTM network of 100 neurons. This network has been explained on a previous section, however, there are some parameters worth to mention.

- Dropout has been established as 0.1, which means each LSTM's unit (neuron) has a 10% chance to be dropped from the linear transformation process, increasing the capacity of the network to generalize and avoid overfitting.

- Best results have been obtained with Rectified Linear Units (ReLU) as activation function. The figure below shows the representation of ReLU vs Sigmoid (a widely used activation function). ReLU works pretty well on Recurrent Neural Networks as LSTM because it solves a crucial problem when dealing with recurrent weights that are retrofitted, the vanishing gradients. This problem usually arises when the gradients of the activation function (first derivative) become close to zero (both Sigmoid limits), therefore the weights' update amount will be close to zero and making the network hard to train. On contrast, Sigmoid avoids blowing up neuron's activation by limiting the output between [0, 1], avoiding great weights' updates that will make the model hard to converge.

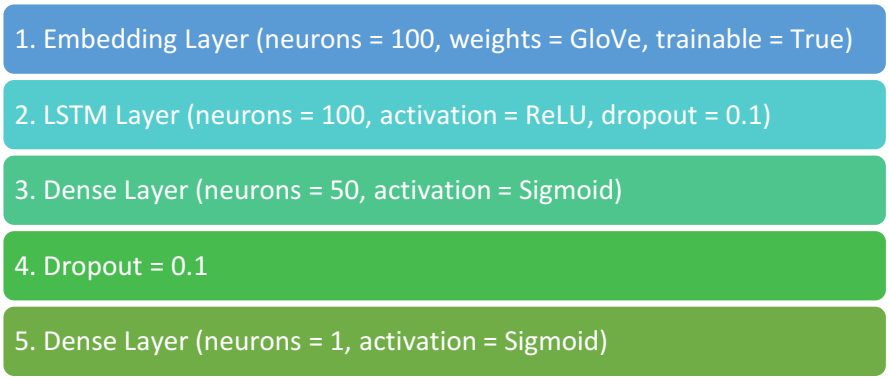
Figure 16: Activation Functions – Sigmoid vs ReLU



Source: S. Sharma (2017)

The output of the LSTM layer is not ready to be saved as the final model, as it has 100 dimensions and our model requires only 1 to determine if the sentiment of a review is positive or negative. As a dimensionality reduction, a Dense layer with 50 neurons has been added, followed with a dropout of 0.1 and a final Dense layer built with 1 neuron. A Dense layer performs linear operations on the input it receives, combined with a non-linear activation function (Sigmoid), will produce values lower than 0.5 to those reviews identified as sentimentally negative, and values greater than 0.5 to those sentimentally positives. The architecture of the model is illustrated on the figure below.

Figure 17: LSTM Model's Architecture



3.3.2 Bidirectional Long Short-Term Memory (BiLSTM)

This network has been built up on the previous one. A regular unidirectional LSTM preserves information that belongs to the past, as a human would do when reading a review. A bidirectional LSTM (BiLSTM), will receive two sources of information, one from the past and another one from the future (the words or phrases that are coming next to the word the network is processing). By doing this, a BiLSTM will combine the two hidden states and will be able to preserve information from past and future at any point in time. Figure 6 which was showed previously illustrates how this network works. Once input has been fed into both LSTM networks (original and inverse order), outputs are concatenated and passed to a Dense Layer of 50 neurons to reduce the dimensionality going through a dropout 0f 0.1 and ending up with a Dense Layer of 1 neuron activated

by a Sigmoid function which will produce outputs lower than 0.5 to reviews predicted as negative and outputs greater than 0.5 to reviews predicted as positive.

Here is an example to understand how LSTM and BiLSTM operate on a common problem, called text generation, where the network must predict the most suitable word or sentence given its context. Let's say the sentence starts with the following words:

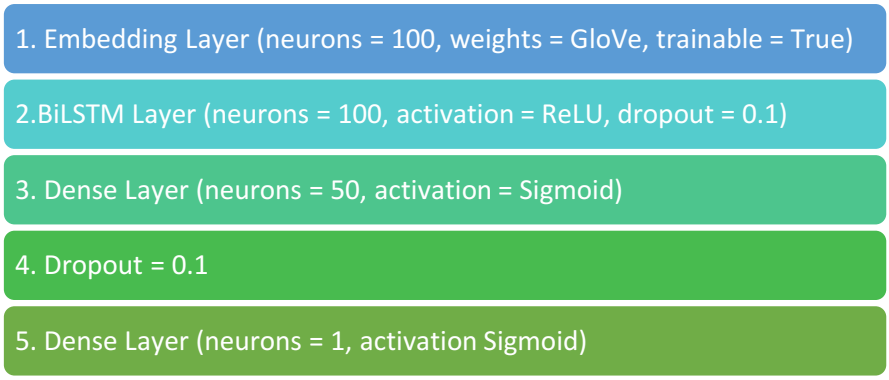
I will try to …

A LSTM network will try to predict the next word, which could be almost anything (*get, make, call, jump, play, etc.)*, being able to check previous sentences to get some context about what the text is talking about. However, a BiLSTM network will look to the future too, obtaining a completer and more reliable vision of the context since it's closer on space and time, therefore more a stronger relationship between the blank and the upcoming words exist.

I will try to …… your email tonight

Here a BiLSTM will produce more accurate predictions (*reply, forward, print, read*), as the network understands what the sentence is about. A similar approach is taken when predicting sentiments on reviews. The architecture of this model is showed on the figure below.

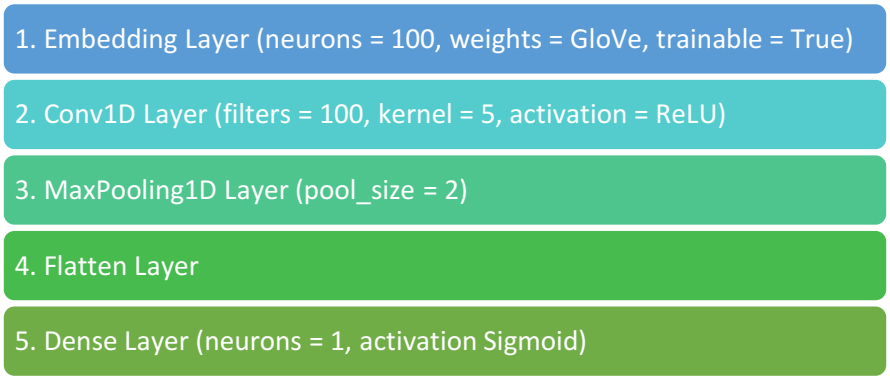
Figure 18: BiLSTM Model's Architecture



3.3.3 Simple Convolutional Neural Network (CNN)

This model starts as usual with an Embedding Layer to build the dense vectors that will be passed to a 1 Dimensional Convolution Layer made up by 100 filters (which will result in an output's dimensionality of that number), a kernel size of 5 (convolution window) and ReLU as activation function. The output will be fit into a Max Pooling Layer, which will down-scale the previous layer's output by a factor represented by *pool_size* parameter, therefore allowing an easier feature extraction. Then, a Flatten Layer will concatenate Max Pooling's output into a unique vector which will be fit into a Dense layer activated by a Sigmoid function which will perform the final predictions. The internal mechanism of a CNN is illustrated on Figure 7. The architecture of the model is shown on the following figure:

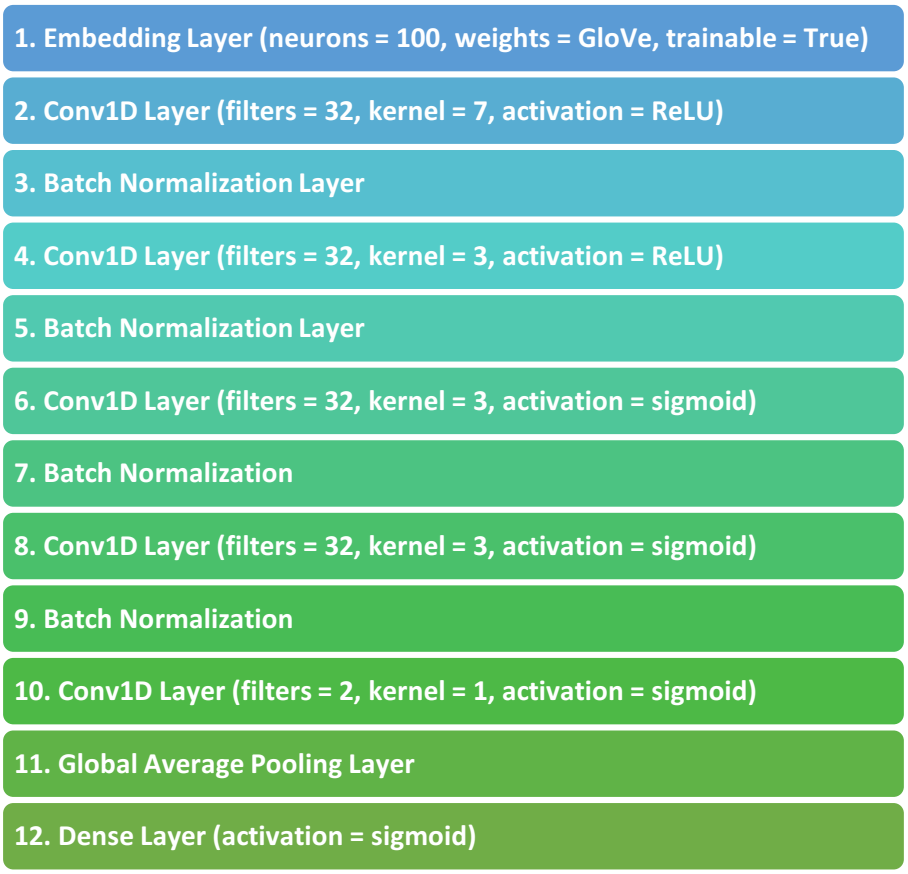
Figure 19: CNN's Model Architecture



3.3.4 Batch Normalized Convolutional Neural Network (CNN2)

This model uses the concepts explained in the previous network to perform more complex operations that led to accuracy improvements. Right after the Embedding Layer, it starts a block made of four 1D Convolutional Layers with a Batch Normalization layer between each of them. This layer will normalize the weights between [0, 1] reducing the covariance shift, which leads to reducing overfitting by a smooth regularization effect, adding noise to each hidden layer's activation. The output of this block will be used by the last 1D Convolutional Layer, after which a 1D Global Average Pooling Layer outputs an average of every incoming feature array extracted from convolutional layers previously obtained. The architecture of this model is as follows:

Figure 20: CNN2's Model Architecture



Section 4

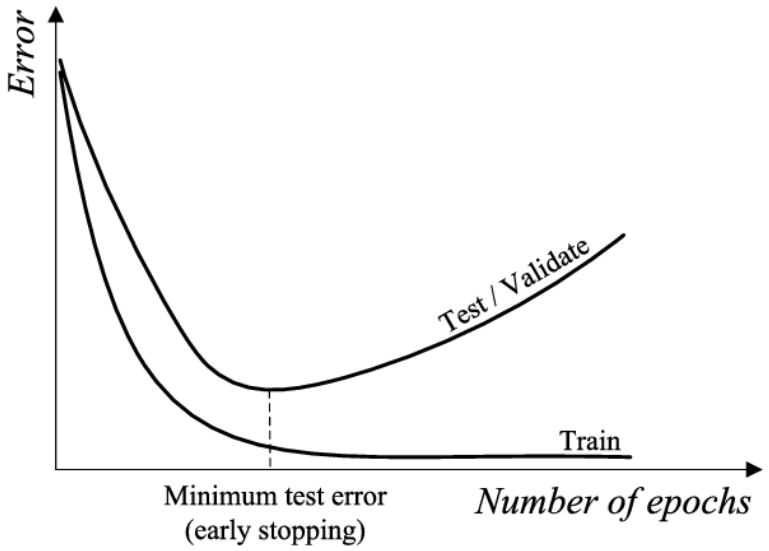
Results Analysis

4.1 Model Training

Every model has been trained under the same parameters and using the same dataset to be able to perform a reliable comparison between all of them. Every model has been compiled using the following parameters:

- Loss function: *Binary cross-entropy*, which measures the model's performance and increases as the predicted label diverges from the actual label.
- Optimizer: *Adam*, optimization algorithm which *"computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients."* (Kingma & Ba, 2014). This means that on each epoch the learning rate (learning capacity of the network for each step) is modified according to the gradients of the loss function (first and second derivatives). This optimizer result on a better performance than using static learning rates.
- Epochs: the model will receive the full training dataset 10 times during the training process.
- Validation split: a 20% of the full training dataset has been used as validation data to evaluate the model at the end of each batch of training.
- Batch size: the training dataset will be chunked into several batches made of 64 samples each.
- Early Stopping: the training process will be stopped if the loss (prediction error) on the validation set hasn't improved for the last 3 epochs. Therefore, the best model that will be saved will be the one obtained 3 epochs back.

Figure 21: Early stopping example



Source: Santos (2012)

4.2 Models Comparison

One of the most questioned assumptions made on this research has been the relevance on accuracy performance of the max length padding, performed during the data preprocessing. This parameter assumes that once the sentence is exclusively formed by Adjectives, Nouns, Adverbs and Verbs, only the first *"x"* words from the review will be used for training and testing the sentiment of the full review. If a sentence is shorter than that this value, zeros will be added at the end of it to match the padding length. As a result, we are assuming that first *"x"* words are more relevant than the last ones, and only short reviews (which have on average more sentimentally weight per word) are fully represented. To choose *max_length* value Equation 1 has been used, taking *z-values* equal to *0.3*, *0.5* and *0.8*. Each of the experiments below have been performed 5 times to ensure reliability on results.

Figure 22: Test Accuracy Comparison z-value=0.3

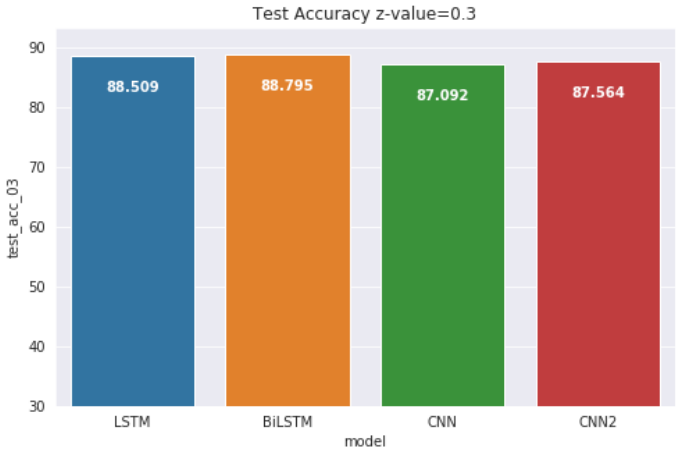


Figure 23: Test Accuracy Comparison z-value=0.5

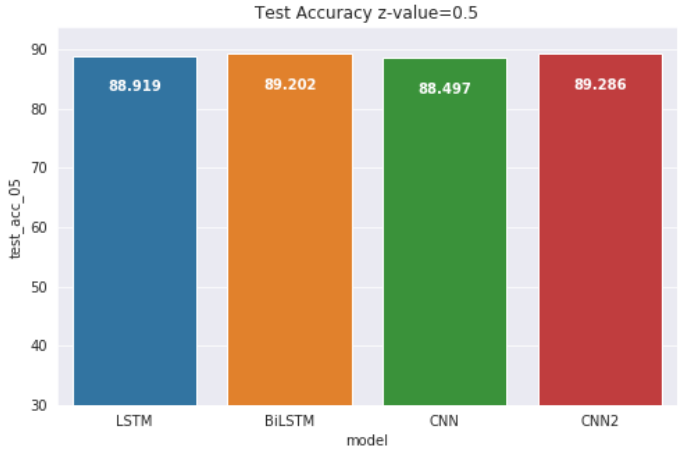
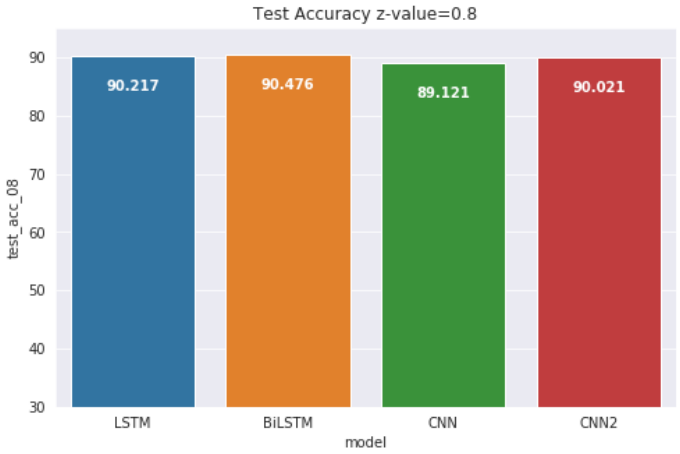


Figure 24: Test Accuracy Comparison z-value=0.8



As the accuracy values show, there is in fact an increase in accuracy with higher *z-values*, however it is remarkable how well networks have performed with the *z-value=0.3*, taking into consideration these values are pretty close to the State-of-the-Art on SST2 dataset. Higher *z-values* increase accuracy by taking into consideration more words per review to train and test the model, however, this leads to bigger model storage size and a greater time for performing predictions, both key factors for the second part of the project, a demo web application.

By analyzing the results above, it is worth to mention how the accuracy values on *z-value = 0.8* experiments are really close to the State-of-the-Art on SST2 dataset, especially BiLSTM *90.476%* accuracy value, which has improved Yu et al. (2017) approach of Tree LSTM. It would be of high interest to check the performance of the cleaning, preprocessing and BiLSTM model on the standard SST2 dataset, in order to evaluate how well the approach taken on this project is. However, this goes beyond the scope of this project and will be stated at the end of the document under the section *Future Work Guidelines*.

The following figures show the importance of Early Stopping parameter, which stops training when the loss on the validation dataset hasn't improved for the last 3 epochs. Every model has stopped it's training process before reaching the limit of 10 epochs, therefore the parameter improves training times and accuracy.

It is important to mention how LSTM, CNN and CNN2 tend to evolve in a similar way, improving with each epoch until they start overfitting and Early Stopping comes into play, which is usually between the second and the fourth epoch depending on the case and the network. On the other hand, BiLSTM finds always the best accuracy model on the first epoch, decreasing its performance until Early Stopping finishes the training process.

Figure 25: Validation Loss vs Accuracy Comparison z=0.3

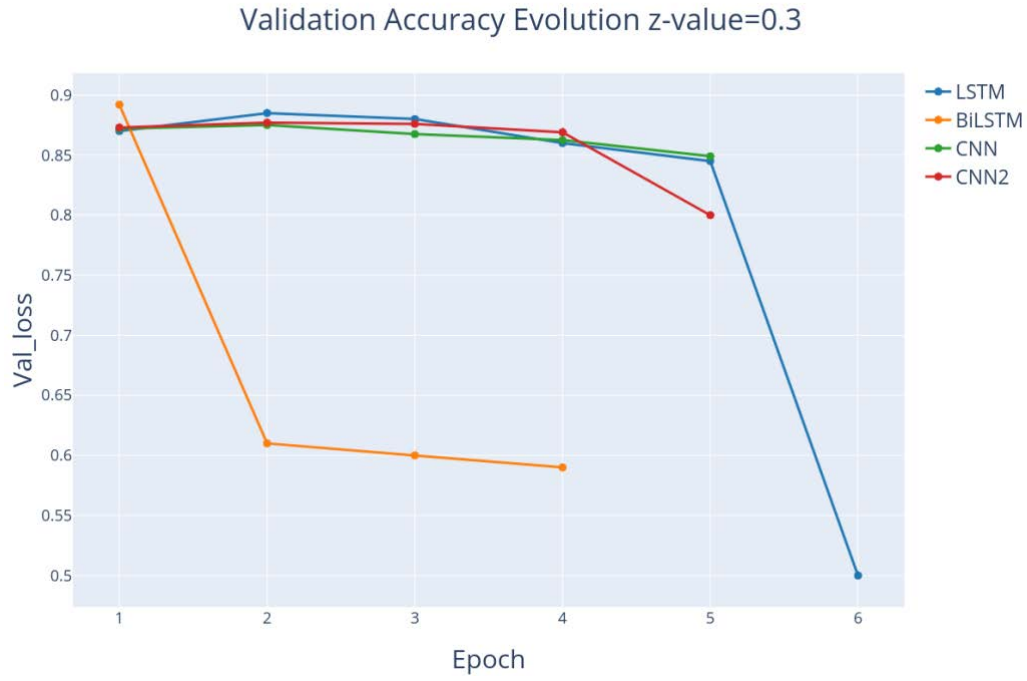
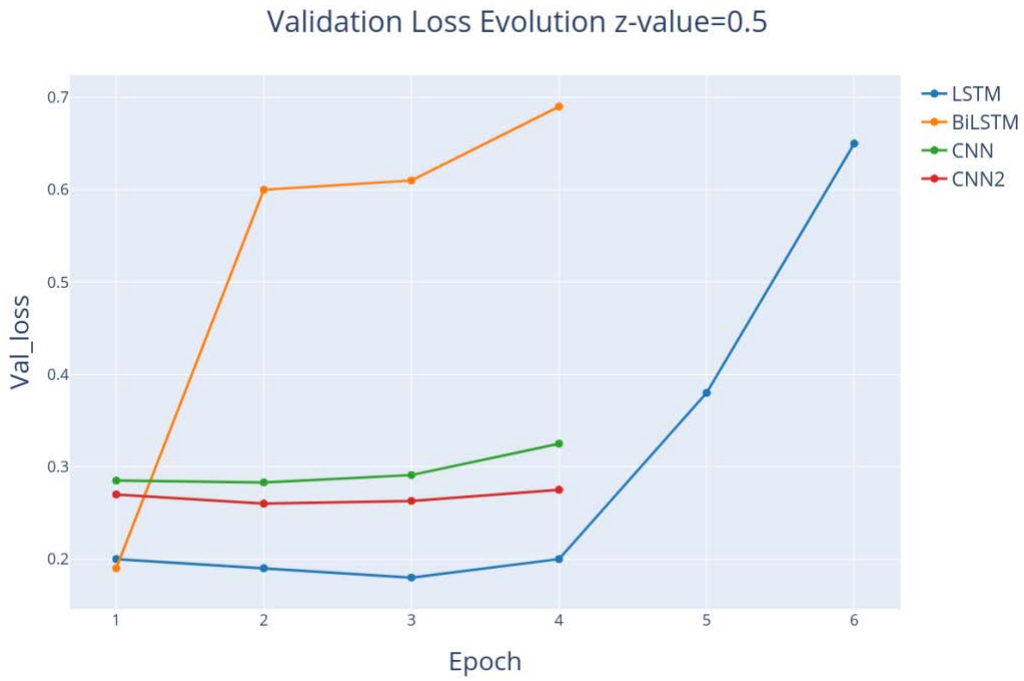


Figure 26: Validation Loss vs Accuracy Comparison z=0.5



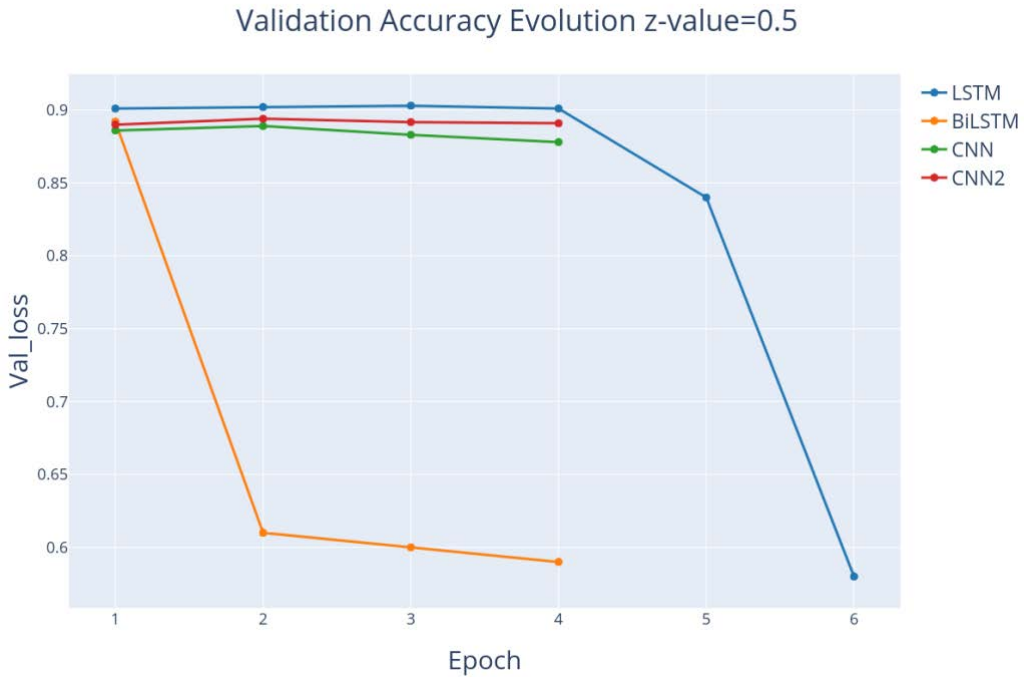
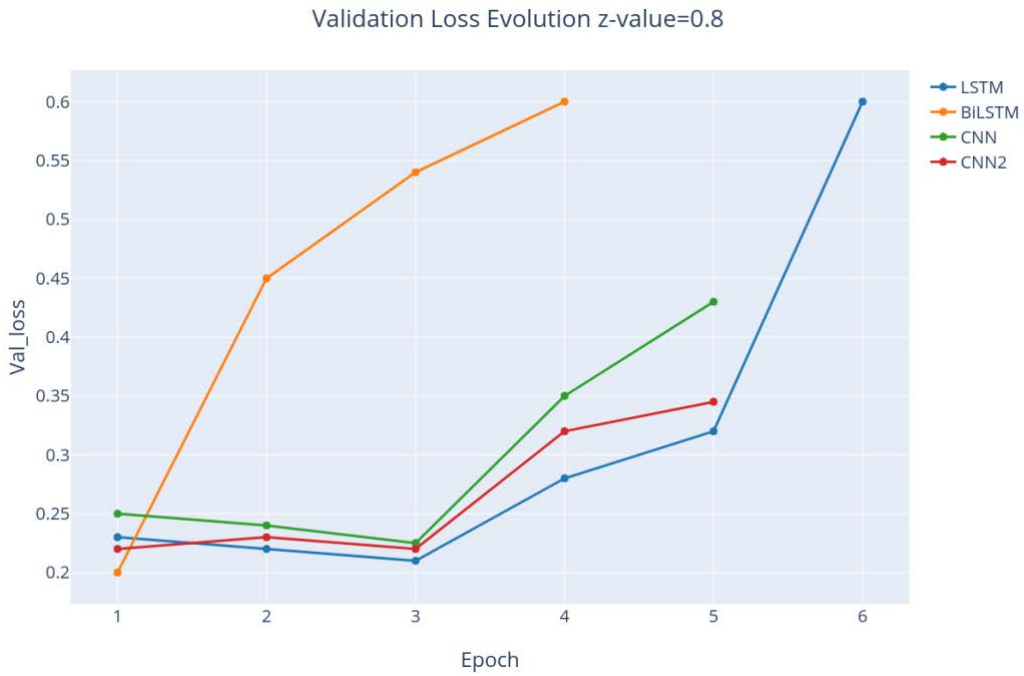


Figure 27: Validation Loss vs Accuracy Comparison z=0.8



After analyzing the four models not only by its final accuracy but by how step by stem the model has been trained during the training process, it is clear that BiLSTM is the best model on terms of accuracy for any *z-value* and on terms of training efficiency, since as the above figures show, on every experiment the model obtained on the first epoch is always the best one.

The following section will explain the development of the web application, which takes some files from the preprocessing and training steps. The first file is the vocabulary, the tokenizer's index table that will allow the app to make the preprocessing steps required to obtain the model's prediction, as every review has to be encoded following the same rules than the ones taken on the training dataset. The second file is the model trained, which will be the best model obtained, BiLSTM on *z-value = 0.8*, with a total storage size of 154 MB.

Section 5

Web Application Development

5.1 Framework

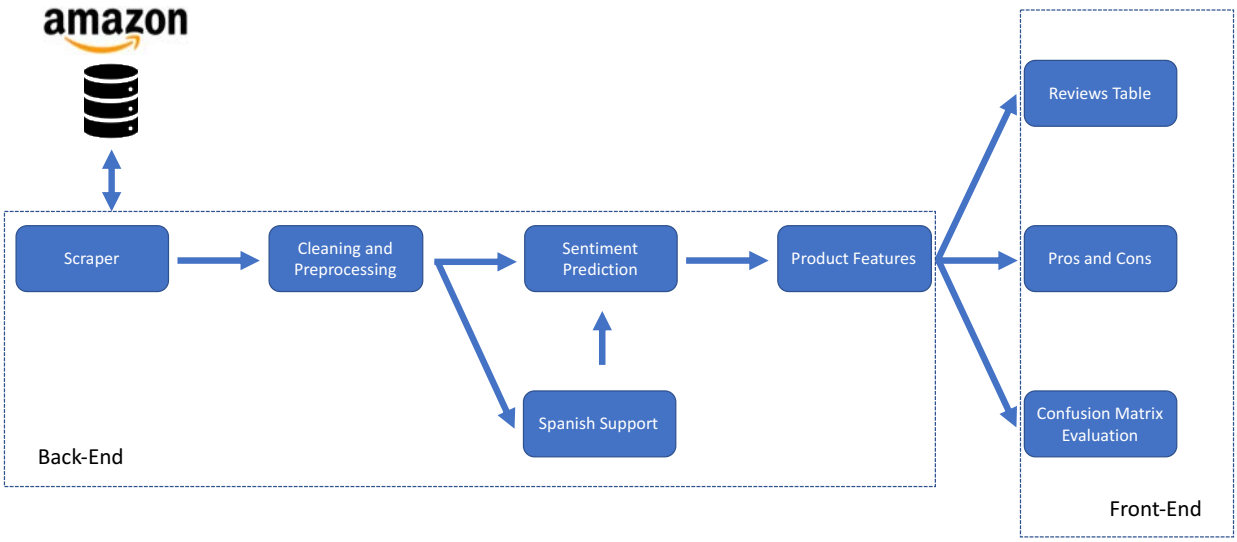
The main goal of this project wasn't just to train a sentiment analysis model with an accuracy close to the State-of-the-Art, but to show the potential of this model by developing a demo with which the user can interact, see the cleaning process and obtain sentiment predictions as well as useful insight reports for any product on Amazon.com or Amazon.es.

On web-based analytics apps, an R package called *Shiny* has been widely used for the last years to build interactive web apps. However, a new framework developed by Plotly's team (known for its interactive data visualization package) was released recently under the name of *Dash*, which could be easily integrated into the model and cleaning process developed on a previous stage of the project as it has Python support. The development of the web app has been made using PyCharm Free Community Edition and can be found on the following GitHub repository:

https://github.com/albergar2/SA_Project

The web application has been split into several functionalities that will allow the user to obtain a sentiment analysis report for any product on Amazon as well as to see the cleaning, preprocessing, and prediction steps taken for every review of the given product. The following figure illustrates how the different app's functionalities relate to each other.

Figure 28: Relationships between app's functionalities



5.2 Back-end

5.2.1 Scraper

The first step once the user has introduced an Amazon product's URL is to load the reviews written by customers who have purchased the product into memory. This process is constrained to the amount of pages Amazon lets a script to read before blocking its calls to the website. Experimental results show that this number is around 500 reviews, which translates into 50 pages from Amazon's website and is enough for our demo purposes.

The script for scraping each web page has been a personal development, generating from the product´s URL, every customer reviews URL, loading its HTML structure and extracting from the class names the following information for each review:

- Star Rating [1, 5]: rating given by the customer to the product
- Title: title of the customer review
- Body: customer review

The script will return a pandas dataframe with this information for each of the reviews loaded, which will be used as an input for the next functionality.

5.2.2 Cleaning, Preprocessing and Sentiment Prediction

A key point to remark here is that the expected input for the model has to follow the same structure and requirements as the data it was used to train it. This makes mandatory to clean and preprocess the customer reviews following the same steps explained earlier on this document, encoding every review with the tokenizer's index table (vocabulary). Thus, the pandas dataframe received will be transformed using a similar workflow as the one implemented for the training data, exposed on Figure 15.

Once each review has been encoded they are ready to be fed into the trained model to obtain a prediction. The model loaded at this point is the one which obtained best results, BiLSTM on *z-value=0.8*. Predictions made by this model will be a number between [0, 1], since the model has a Sigmoid function as the last layer, meaning any value lower than 0.5 will be categorized with a negative label and any value greater than 0.5 will have a positive label. However, the reviews had 5 points rating system, so to match these categories a discretization of the results has been done obtaining the following labels:

- 1 Star → $0 \leq \ Prediction\ Value < 0.2$
- 2 Stars → $0.2 \leq \ Prediction\ Value < 0.4$
- 3 Stars → $0.4 \leq \ Prediction\ Value < 0.6$
- 4 Stars → $0.6 \leq \ Prediction\ Value < 0.8$
- 5 Stars → $0.8 \leq \ Prediction\ Value \leq 1$

This discretization rule has been chosen to give the same amount of Prediction Value Points to every category, but a non-uniform distribution might be more accurate when evaluating the confusion-matrix.

5.2.3 Spanish Support

Natural Language Processing research has been done mostly in English for the last years It is difficult to find libraries and packages in other languages similar to the ones explained above, *Gensim* and *NLTK*, with similar accuracy values, since language semantics and syntax differ significantly between languages. Despite of the fact that a dataset from Amazon.es customer reviews could be build using a modified version of the script developed to scrape and load reviews, this would have been taken a considerably amount of time and the project would have gone off on a tangent.

In order to give support for any product from Amazon.es, an integration has been done with Google Translator Python Library, *googletrans*, to translate any review written in Spanish into English. The workflow works on a similar way, once the reviews have been translated into English, the model predicts the sentiment value for each of them and the following functionalities continue working the same way, generating the insights in English. At the end of all this process, the results and insights are translated again from English to Spanish, to show to the user the content in the original language. Some people may argue than translations made by Google are not accurate, however, for the purpose of this project, which is to identify if the sentence is positive or negative, there is no need for high accuracy translations, as a synonym sentimentally similar with the word will satisfy the requirements.

As expected, these translations increase dramatically the processing time the user has to wait since submitting the product until the results are showed on the app. Experiments show that reviews from Amazon.com have an average processing time of *0.15 sec/review*, while reviews from Amazon.es scale up to *0.8 sec/review*, which translates into a 433% increase.

5.2.4 Product features

Nowadays the practice of checking on internet for an expert analysis before purchasing a new smartphone, camera or any other product is widely expanded. These analyses are time consuming for experts and often biased by their own subjectivity.

Amazon gives access to reviews written by customers of all its products but doesn't offer a tool that aggregates this customer reviews into key positive and negative features. A naïve approach to this task has been included in this project, in the hope that some researcher will continue with the development of more sophisticated techniques.

On the cleaning and pre-processing process, a POS-Tagging filter to extract exclusively Adverbs, Adjectives, Verbs and Nouns was applied to every review, in order to keep sentimentally meaningful words. This step was crucial by removing from the training data words like Pronouns and Determinants that didn't have any sentimental meaning. Continuing with this approach, product's features are Nouns surrounded by Adjectives, Adverbs and Verbs that will determine if the sentiment about this features is positive or negative:

*"The new **smartphone** has a **camera** that works pretty well under **low-light circumstances**"*

The example above could be a positive review where the Nouns have been highlighted in bold font. Assuming the model predicts with good accuracy (which certainly it does), each of these nouns will be added to a python dictionary with a value of +1 in case of a positive review, and a value of -1 for negative reviews. Iterating this process over every review scraped, the resulting dictionary will contain every noun that appeared across the set of reviews, linked to a sentiment value. Sorting this dictionary, we can obtain the Top 5 Pros and Cons based exclusively on the most reliable and objective data, customer reviews.

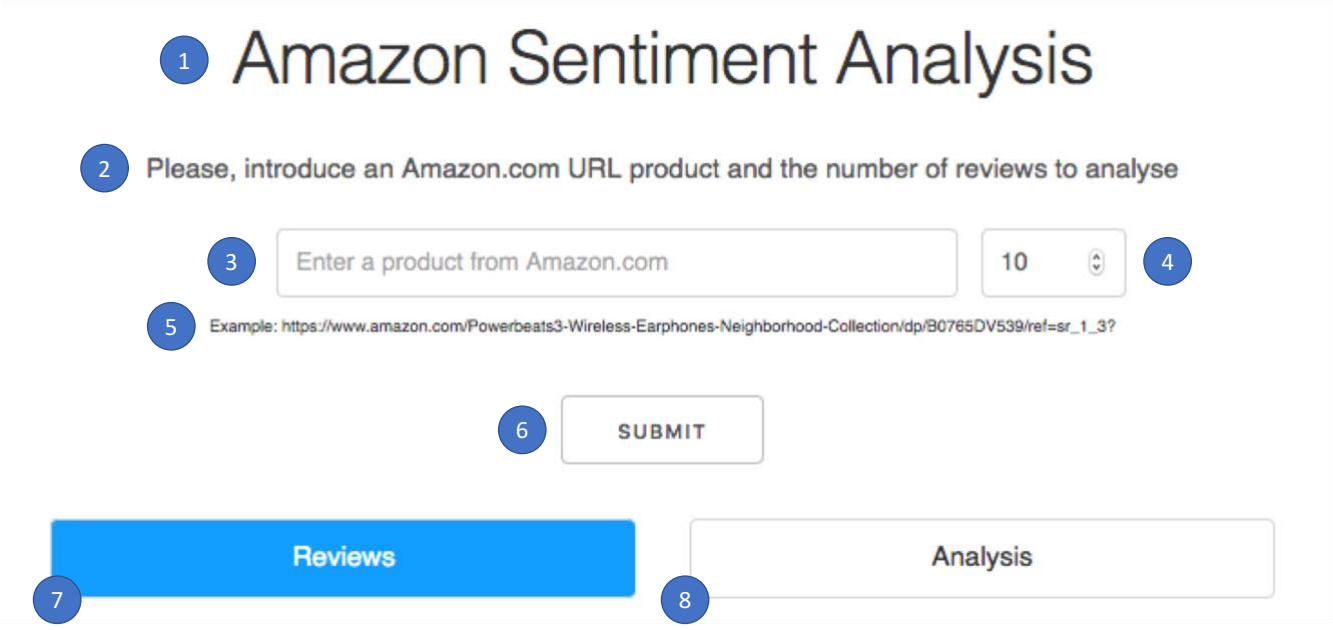
This approach has several problems, even after removing stop-words and aggregating words by its lemmas and synonyms, the appearance of wrong written words or wrong identified nouns by the Pos-Tagging algorithm can lead to inaccurate results. However, these problems tend to blur when the technique is applied over a great amount of reviews, since statistically the recurrent appearance of meaningful words is higher.

5.3 Front-end

The app has been developed using Dash by Plotly framework, which relies on Flask to run easy and deployable web analytics apps. This app has been developed exclusively as a demo of a real project that could be made integrating the model and preprocessing techniques explained on this document. Therefore, as a demonstration web application, this document won't go deep into functional analysis requirements as it goes well beyond the scope of this project.

5.3.1 Homepage

Figure 29: App's Homepage Structure



The figure above illustrates the homepage structure that will interact with every user interested in the project. The numbered elements are the following:

1. Demo App's Title.
2. Simple and direct instructions on how to interact with the app.
3. Search box where the user must introduce a URL of any product from Amazon.com or Amazon.es.
4. Total number of reviews to be loaded from that product.

5. Clarifying example of the input that the app is expecting
6. Submit button, which will result on the back-end into loading reviews from the product specified, preprocessing these reviews, loading the model, predicting their rating and showing the result.
7. Reviews Tab, each review loaded will be shown here, as well as its original rating, preprocessing output, normalized and discretized predicted rating and keywords (Nouns extracted from the review, useful for the Pros and Cons section).
8. Analysis Tab, where 4 plots will be shown:
    a. Top 5 Pros and Cons
    b. Real vs Predicted Star Distribution
    c. Fine-Grained Confusion Matrix
    d. Polar Confusion Matrix

The following subsections will explore the contents of both tabs, Reviews and Analysis.

5.3.2 Reviews Tab

In order to show a full example of the app's functionality a product from Amazon.com has been chosen. A remarkable point here is the difficulty to find Amazon products with a high number of negative reviews (best guess is that they must have been removed from the site), so the available products tend to have more positive reviews.

As a personal preference, a Camping Tent[1] has been chosen to be the product to test the app's functionalities and to evaluate model's performance. As the figure below shows, this product has over 2,000 reviews with unbalanced ratings towards positive ratings. This unbalanced set of reviews is usual for any product with more than 300 reviews, which is the minimum quantity to obtain useful insights.

[1] https://www.amazon.com/Coleman-6-Person-Instant-Cabin-2000018017/dp/B004E4ERHA/

Figure 30: Example Product

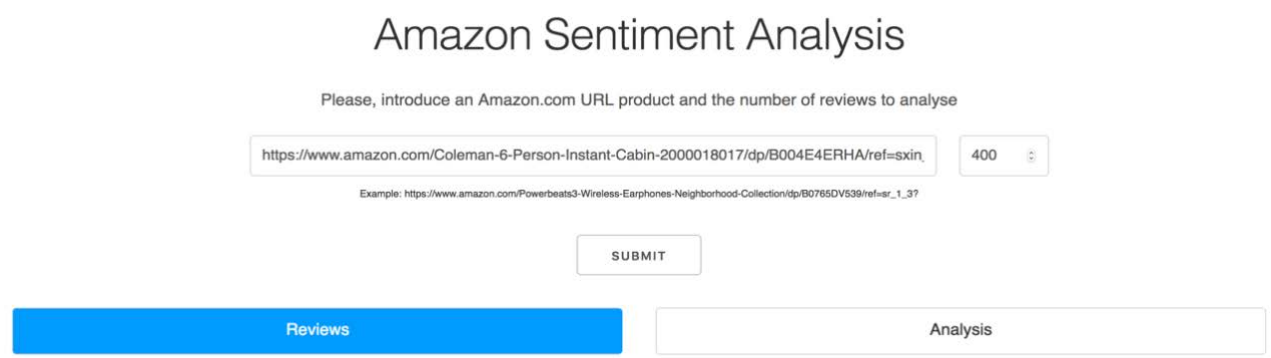
Source: Amazon.com

By introducing the product's link on the app and choosing 400 as the number of reviews to load, the app will start the backend workflow and finally showing the product's name and a table where each row represents a single review and the columns are defined by the following tags:

- Review: original review loaded from Amazon.com.
- Stars: rating given by the user who wrote the review.
- Clean: the original review is cleaned and preprocessed using the tools explained in Section 3.2.
- Prediction: model's output prediction for the review's rating. The prediction's interval is [0, 1].
- Predicted Star: star prediction from a discretization of the model's output, as explained in Section 5.2.2.
- Keywords: main features about the product to be considered for the Pros and Cons analysis.

The figure below illustrates the app's status at this point, showing just 3 relevant reviews, representing the steps taken by the app from loading the review to obtaining the number of predicted stars and keywords. The execution time for this example has been 43.742 seconds.

Figure 31: Example Reviews Tab Front-End



Coleman 6-Person Cabin Tent with Instant Setup | Cabin Tent for Camping Sets Up in 60 Seconds

Review	Stars	Clean	Prediction	Predicted Star	Keywords
Water-tight: Wrong Clip Sewn In After reading several reviews talking about leaks, I soaked mine with the hose for an hour. Luckily, it didn't leak! I let it dry and was about to pack it away when I noticed one of the hooks wasn't connected. This is when the one-star review begins. One of the hooks that fasten the tent to the poles was defective. Coleman's manufacturer sewed a smaller hook for the top part of the tent onto the bottom, where the poles are thicker. It snapped instantly when I tried to put it on the thicker pole.	1	water tight wrong clip sewn reading several reviews talking leaks soaked mine hose hour luckily leak let dry pack away noticed hooks connected star review begins hooks fasten tent poles defective coleman manufacturer sewed smaller hook top part tent bottom poles thicker snapped instantly tried put thicker pole	0.264	2	clip thicker manufacturer hour bottom mine hook leak leaks water part pack poles
Cheap fair weather tent, easy setup Great tent, simple setup and plenty of room for 4. I wouldn't sleep six unless we're all hauled away to a refugee camp. The seams do need to be attended too. Nothing seam seal won't take care of. Purchase tarp to put the tent on, as the bottom is very thin. I picked up a roll of indoor outdoor carpet to lay inside also (optional) to ward off sharp rocks	4	cheap fair weather tent easy setup great tent simple setup plenty room sleep hauled away refugee camp seams need attended nothing seam seal take care purchase tarp put tent bottom thin picked roll indoor outdoor carpet lay inside also optional ward sharp rocks	0.686	4	seams seal outdoor seam carpet care roll rocks purchase room tarp
Easy to set up! Probably not the best tent for four people. This tent was really great, and easy/quick to set up. I really love it, but it doesn't come with a rain fly, which was annoying. Also, this tent probably would only sleep four if you weren't using cots or air mattresses, and didn't have much of anything else inside the tent. I don't recommend trying this tent for four people.	4	easy set probably best tent people tent really great easy quick set really love come rain fly annoying also tent probably sleep using cots air mattresses much anything else tent recommend trying tent people	0.912	5	air rain anything cots mattresses

As it is possible to see, the model is not always accurate on predicting the star. Something important to note here is that the review written and the rating given to that review by an Amazon.com user are not related in some cases and could depend on the user's subjectivity. All of this three review's examples were correctly tagged on terms of positive/negative, however the discretization process didn't work with the precision expected.

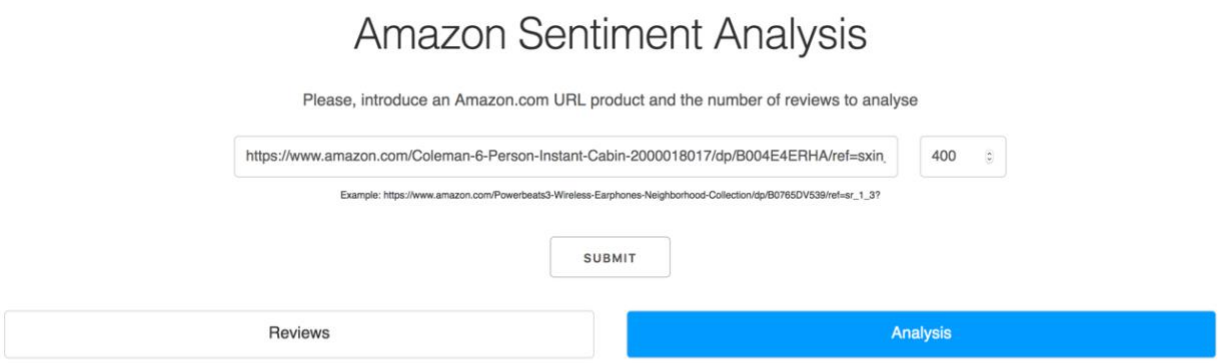
First review has been tagged as negative due to some words the user wrote on the review associated with that sentiment, such as *wrong, leaks, soaked, defective* and *snapped*. The second and third reviews have been tagged as positive due to the following words: *easy, best, quick, recommend, simple, cheap* and *fair*.

Keywords column show the outcome from the back-end's Product Features functionality. Basically, the app has taken the cleaned review, keeping only Nouns, which are the features of the product.

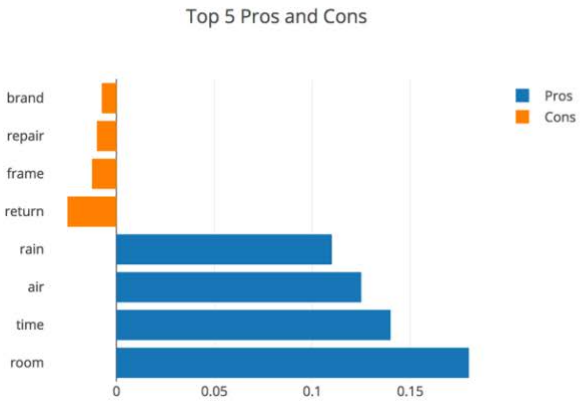
5.3.3 Analysis Tab

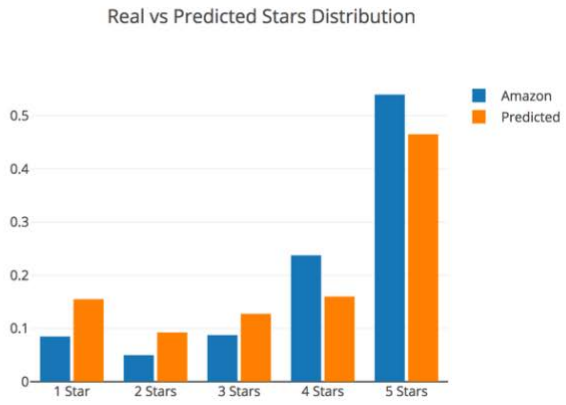
Continuing with the Camping Tent example, the Analysis Tab shows 4 different plots with insights and information about the performance of the model's output.

Figure 32: Example Reviews Tab Front-End



Coleman 6-Person Cabin Tent with Instant Setup | Cabin Tent for Camping Sets Up in 60 Seconds





Confusion Matrix Polarity

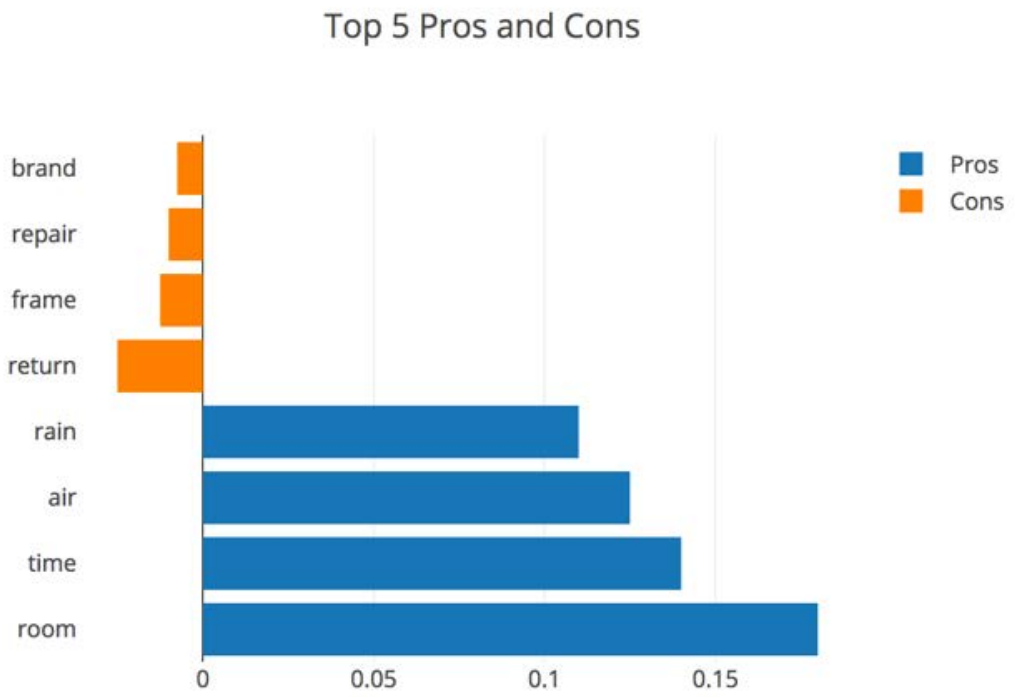
Pred \ Real	Negative	Neutral	Positve
Negative	46	16	37
Neutral	4	6	41
Positive	4	13	233

Confusion Matrix Stars

Pred \ Real	1 Star	2 Stars	3 Stars	4 Stars	5 Stars
1 Star	23	14	10	9	6
2 Stars	6	3	6	12	10
3 Stars	4	0	6	18	23
4 Stars	1	2	5	18	38
5 Stars	0	1	8	38	139

5.3.3.1 Top 5 Pros and Cons

Figure 33: Top 5 Pros and Cons Example



This plot shows the Top 5 features of the product for each sentiment (positive and negative). The example shows 4 negative features and 4 positive features because a threshold of appearance has been fixed to 0.005, meaning there are not more features with an appearance probability greater than this threshold.

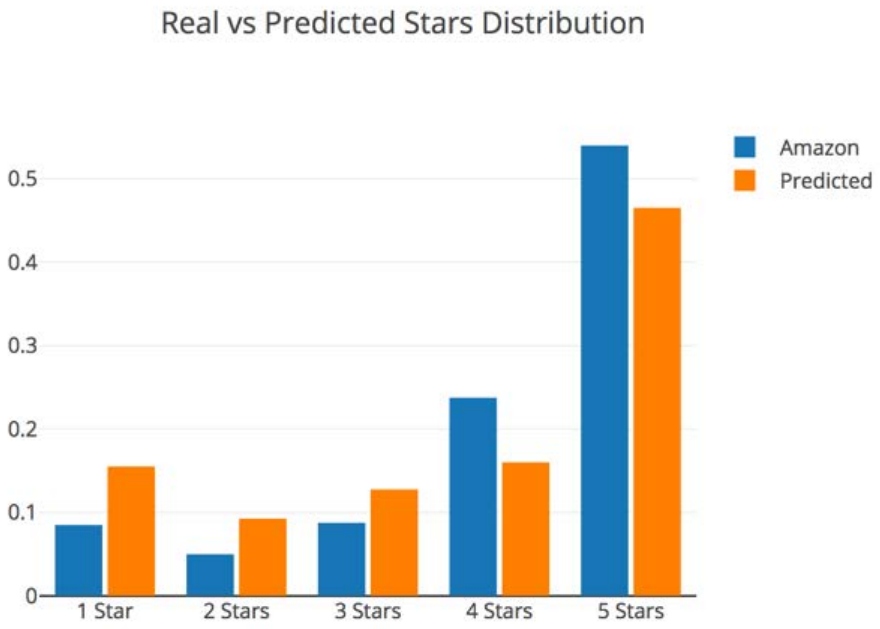
Due to the unbalanced ratings from this product, positive features appear more frequently than the negative ones. By looking at the plot we can get an idea of the key features the customers like or dislike about the product:

- Positive
  - Rain: the tent is waterproof; therefore, customers talk positively about it.
  - Air: good ventilation.
  - Time: the tent's main feature is its instantaneous setup.
  - Room: the tent is designed for up to 6 people, so the room space is also important.

- Negative:
  - Brand: some customers doesn't perceive the product as matching the brand's reputation.
  - Repair: few customers have had problems with tent and are complaining about the reparations needed.
  - Frame: the tent's frame seems not matching customers' expectations.
  - Return: this feature is very frequent on the negative side, as the unhappy customers return the product and write a negative review.

5.3.3.2 Real vs Predicted Star Distribution

Figure 34: Real vs Predicted Star Distribution Example



As expected, the unbalanced star distribution is represented on this plot. The original distribution from Amazon.com showed in Figure 30 may not match the real distribution represented at this plot, since the former has been obtained from the 400 reviews loaded instead of the over 2,000 written for the product on Amazon.

As the plot shows, the model predicts more negative reviews and less positive reviews than they actually are, trying to make a more balanced prediction, as it has been trained with a balanced dataset.

5.3.3.3 Polar Confusion Matrix

Figure 35: Polar Confusion Matrix Example

Confusion Matrix Polarity

Pred \ Real	Negative	Neutral	Positve
Negative	46	16	37
Neutral	4	6	41
Positive	4	13	233

This confusion matrix shows the accuracy on positive, neutral and negative predicted tags. There is a big amount of positive reviews that have been tagged as negative, as well as an arbitrary distribution of neutrals. The dataset used to train the model had examples of reviews with 1 and 2 stars as negative and reviews with 4 and 5 as positive, therefore the model didn't learn to identify neutral reviews (3 stars). To obtain neutral labels the discretization explained on a previous section has been used, labelling as neutral any prediction value greater or equal than 0.4 and lower than 0.6.

In fact, instead of giving the same amount of Prediction Value Points to every category (5 star ratings), a light improvement could be performed lowering the lower boundary beneath 0.4 and increasing the greater boundary over 0.6. By doing this the results will improve by labelling as true negatives some of those 16 false negatives labeled as neutrals and labelling as true positives some of those 13 false positives labeled as neutral. However, this adjustment will be made to artificially increase the performance on this specific example, therefore a deeper study of the discretization rules would be needed to obtain a general model that increases improves this confusion matrix for any product.

Taking the values from the diagonal line, the model has achieved a 71.25% accuracy on real data. The low performance could be explained by the fact the 90.476% accuracy was obtained on exclusively positive and negative reviews, a lack of correlation between the customer's rating and the customer's review and a discretization method that can be improved.

5.3.3.4 Confusion Matrix Stars

Figure 36: Confusion Matrix Stars Example

Confusion Matrix Stars

Pred \ Real	1 Star	2 Stars	3 Stars	4 Stars	5 Stars
1 Star	23	14	10	9	6
2 Stars	6	3	6	12	10
3 Stars	4	0	6	18	23
4 Stars	1	2	5	18	38
5 Stars	0	1	8	38	139

This last plot shows a detailed confusion matrix based on the real and predicted star rating. The model used to perform the predictions has been trained using positive and negative reviews and making a uniform discretization of the predicted value. This approach is manifested too on this plot as the model tends to give more weight to the extreme values (1 star and 5 stars) than to others. Taking the values represented on the diagonal line, the model has obtained a real accuracy of 47.25%, which for a 5 label prediction model, as illustrated on Figure 8, is close to the State-of-the-Art on SST-1. The discretization approach seems to be a feasible option, despite of some refinement would be needed to improve the results.

Section 6

Conclusions & future work guidelines

6.1 Conclusions

This document has described the development process of training a sentiment analysis model, a detailed comparison between different deep learning architectures, preprocessing steps on unstructured textual data, development of a web application for demonstration purposes of how to integrate the model into a Product Analytics Tool as well as the insights and reports produced by the web app.

The first part of the project was focused on obtaining the best results possible on the sentiment analysis dataset obtained from the researcher Xiang Zhang using different deep learning architectures. A four model comparison was done with a vanilla LSTM, a vanilla BiLSTM, a vanilla CNN and modified version of CNN. Best results were obtained with BiLSTM model when taking into consideration the padding length of the reviews as a value that makes that 80% (z-value=0.8) of them are fully represented. The accuracy obtained with this model was 90.476%, however, is notable how other models achieved similar but slightly lower performances. Taking into consideration the State-of-the-Art accuracy values, the models developed for this project have achieved pretty accurate predictions on polar sentiment analysis (positive and negative).

The second part of the project was to integrate the best model into a web application to show the potential of a Product Analytics Tool that can predict ratings and generate reports and insights for any product at Amazon. The app was developed using Dash, a web analytics app framework. It allows the user to introduce any product from

Amazon.com or Amazon.es, explore the cleaning and preprocessing steps for any review, obtain the Top 5 Pros and Cons features of the product and check the performance of the model with two confusion matrix. The model has been trained on negative and positive reviews, therefore to obtain a 5 stars prediction model to test the real product's reviews, a discretization method has been implemented. Finally, the naïve approach taken to extract the main product features by keeping only Nouns from the original reviews has been a good idea that gave interesting results.

The model's performance has been evaluated on two confusion matrix, the former evaluated positive, negative and neutral accuracy and achieved a 71.25%, while the last one evaluated the 5 stars rating system from amazon, obtaining a 47.25% accuracy. Fine-grained sentiment analysis State-of-the-Art on SST1 is at 54.00% (Yu et al., 2017), therefore there is room for improvement. Overall, this project has achieved its goal, by obtaining accurate predictions on a creative dataset and showing its potential by a demo web application.

6.2 Future Work Guidelines

This project can be expanded by future researchers in several ways:

- The discretization method used to transform predictions in range [0, 1] into three or five labels has been done giving the same amount of prediction value points to every category. However, has showed on both confusion matrix, a fine tuning can be made to adjust the boundaries for every category, therefore substantially increasing the accuracy. It would be of high interest a deeper development on this by analyzing whether it is more accurate to train a model in 5 different possible targets or a post discretization has the one exposed on this document is a better option.

- The neural network models trained on this project have been pretty basic and more effort has been put into cleaning, preprocessing and parameter tuning rather than on designing complex architectures. Results can be improved taking State-of-the-Art networks such as Tree-LSTM (Yu et al., 2017) which wasn't implemented on this project since there wasn't a stable build for Tensorflow or similar framework that gave support to tree structure networks.

- Focusing on the second part of the project, the web application has been developed as a demo to show the potential of a tool based on Sentiment Analysis and could be interesting to build it in a scalable way to which new features, reports and analysis can be added. The naïve approach taken to extract the most positive and negative features of the product from customer reviews has a good performance but could be improved a lot by introducing a neural feature extractors and making a better pre-processing of the reviews than the one explained in this document.

- The Spanish support approach using Google Translate taken on the project has been very simple and scalable since it can be extended to any language and still using English trained models to make the predictions. However, the processing times are a great deal nowadays and it could be a better option to train models on Spanish and other languages to get the most out of the Product Analytics Tool.

- This project has focused exclusively on Amazon reviews, but once trained the model can be useful for other purposes such as customer feedback, ticketing, social media monitoring, brand monitoring, market analysis, etc.

References

[1] N. Ackermann. "Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences." *Medium*, 2018. [Online].
Available:https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf
[Accessed March 2019]

[2] A. Bhande. "What is underfitting and overfitting in machine learning and how to deal with it." *Medium*, 2018. [Online].
Available:https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76
[Accessed May 2019]

[3] Agerri, Rodrigo & Cuadros, Montse & Gaines, Seán & Rigau, German. "OpeNER: Open polarity enhanced named entity recognition". *Procesamiento de Lenguaje Natural*. 51. 215-218. 2013

[4] J. Barnes, R. Klinger, and S. Schulte im Walde. "Assessing State-Of-The-Art Sentiment Models On State-Of-The-Art Sentiment Datasets*". University of Stuttgart.* 2016 [Online].
Available: https://arxiv.org/abs/1709.04219
[Accessed January 2019]

[5] J. Brownlee. "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning". *Machine Learning Mastery*. 2017. [Online].
Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
[Accessed March 2019]

[6] BOE. *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.* 2018. [Online].
Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673
[Accessed May 2019]

[7] Chen T, Xu R, He Y and Wang X. “Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert Systems with Applications.” 2017. [Online].
Available: https://dl.acm.org/citation.cfm?id=3138681
[Accessed January 2019]

[8] C. Olah. “Understanding LSTM Networks”. *Cristopher Olah Blog*. 2015. [Online].
Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[Accessed February 2019]

[9] D. Datta. “Understanding Convolutions in Text”. *Blick*. 2016. [Online].
Available:http://debajyotidatta.github.io/nlp/deep/learning/word-embeddings/2016/11/27/Understanding-Convolutions-In-Text/
[Accessed March 2019]

[10] A. Dertat. “Applied Deep Learning - Part 4: Convolutional Neural Networks”. *Towards Data Science*. 2017. [Online].
Available:https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
[Accessed March 2019]

[11] D. Karani. “Introduction to Word Embedding and Word2Vec”. *Towards Data Science*. 2018. [Online].
Available:https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
[Accessed March 2019]

[12] Domo. Data Never Sleeps Report 6.0. *Domo*. 2018. [Online]
Available: https://www.domo.com/solution/data-never-sleeps-6
[Accessed May 2019]

[13] European Union Law. “General Data Protection Regulation”. *European Union Law*. 2016. [Online].
Available:https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679&from=ES&lang3=choose&lang2=choose&lang1=EN
[Accessed May 2019]

[14] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. “Retrofitting Word Vectors to Semantic Lexicons”. [Online].
Available: https://arxiv.org/abs/1411.4166
[Accessed January 2019]

[15] Github. *Github: Pricing*. 2019. [Online].
Available: https://github.com/pricing
[Accessed May 2019]

[16] “Gensim: utils – Various utility functions”. *Gensim Topic Modelling for Humans*. 2019. [Online].
Available: https://radimrehurek.com/gensim/utils.html
[Accessed Febreuary 2019]

[17] “Word2Vec”. Google Code Archive. 2019. [Online].
Available: https://code.google.com/archive/p/word2vec/
[Accessed March 2019]

[18] S. Hochreiter and J. Schmidhuber. “Long short-term memory. Neural computation” 9(8):1735–1780. 1997. [Online].
Available: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
[Accesed January 2019]

[19] “Pycharm: Toolbox Subscription.” *JetBrains PyCharm*. 2019. [Online].
Available: https://www.jetbrains.com/pycharm/buy/#edition=personal
[Accessed May 2019]

[20] A. Jhingrans. “Dynamic Warehousing Around Unstructured Data”. 2018. [Online].
Available: https://jhingran.typepad.com/anant_jhingrans_musings/2007/04/dynamic_warehou.html
[Accessed December 2018]

[21] N. Jumhare, R. Rajeswari, B. Jayakrishnan. “Sentiment analysis based on Twitter data on violence”. *Asian Journal of Pharmaceutical and Clinical Research*. 2017. [Online].
Available: https://innovareacademics.in/journals/index.php/ajpcr/article/view/20521
[Accessed May 2019]

[22] "Kaggle: Your Home for Data Science". *Kaggle*. 2019. [Online].
Available: https://www.kaggle.com/
[Accessed February 2019]

[23] A. Bittlingmayer. "Amazon Reviews for Sentiment Analysis". *Kaggle Dataset*. 2017. [Online].
Available:https://www.kaggle.com/bittlingmayer/amazonreviews/kernels?sortBy=hotness&group=everyone&pageSize=20&datasetId=1305&turbolinks%5BrestorationIdentifier%5D=87982a73-73c6-4182-90ab-94e2e7550c4b
[Accessed February 2019]

[24] "RNNNLP - @madhurimaganguly" *Kaggle Kernel*. 2018. [Online].
Available: https://www.kaggle.com/madhurimaganguly/rnnnlp
[Accessed February 2019]

[25] "Sentiment Analysis with Bidirectional LSTM - @liliasimeonova" *Kaggle Kernel*. 2017. [Online].
Available: https://www.kaggle.com/liliasimeonova/sentiment-analysis-with-bidirectional-lstm
[Accessed February 2019]

[26] "LSTM_With_1M_reviews - @arnabd2002". *Kaggle Kernel*. 2018. [Online].
Available: https://www.kaggle.com/arnabd2002/lstm-with-1m-reviews
[Accessed February 2019]

[27] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A convolutional neural network for modelling sentences*". In Proc. 52nd Annu. Meeting Association Computational Linguistics*, 2014, vol. 1, pp. 655–665. [Online].
Available: https://www.aclweb.org/anthology/P14-1062
[Accessed January 2019]

[28] "Keras: The Python Deep Learning Library". *Keras*. 2019. [Online].
Available: https://keras.io/
[Accessed February 2019]

[29] *"*The Sequential model API*". Keras Documentation*. 2019. [Online].
Available: https://keras.io/models/sequential/
[Accessed February 2019]

[30] Kingma, Diederik & Ba, Jimmy. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations*. 2014. [Online].
Available: https://arxiv.org/abs/1412.6980
[Accessed March 2019]

[31] Y. Kim. "Convolutional neural networks for sentence classification". *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014. [Online].
Available: https://www.aclweb.org/anthology/D14-1181
[Accessed January 2019]

[32] P. Lambert. "Aspect-level cross-lingual sentiment classification with constrained SMT". *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015. [Online].
Available: https://www.aclweb.org/anthology/P15-2128
[Accessed January 2019]

[33] "Lenovo ThinkPad T480". *Lenovo*. 2019. [Online].
Available: https://www.lenovo.com/es/es/laptops/thinkpad/t-series/ThinkPad-T480/p/22TP2TT4800
[Accessed May 2019]

[34] T. Mikolov, G. Corrado, K. Chen, and J. Dean. "Efficient estimation of word representations in vector space". *In Proceedings of the International Conference on Learning Representations (ICLR 2013)*. 2013. [Online].
Available: https://arxiv.org/abs/1301.3781
[Accessed January 2019]

[35] V. Nair and G. Hinton. "Rectified linear units improve restricted boltzmann machines". *In Proceedings of the 2010 International Conference on Machine Learning (ICML)*. 2010.
Available:https://www.researchgate.net/publication/221345737_Rectified_Linear_Units_Improve_Restricted_Boltzmann_Machines_Vinod_Nair
[Accessed January 2019]

[36] "NLTK 3.4.1 documentation". Natural Language Toolkit. 2019. [Online].
Available: https://www.nltk.org/
[Accessed February 2019]

[37] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson. "Semeval-2013 task 2: Sentiment analysis in twitter". *In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. 2013. [Online].
Available: https://www.aclweb.org/anthology/S13-2
[Accessed January 2019]

[38] M. Nguyen. "Illustrated Guide to LSTM's and GRU's: A step by step explanation". *Towards Data Science*. 2018. [Online].
Available:https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
[Accessed Febreruary 2019]

[39] "Constituency Parsing". *NLP Progress*. 2019. [Online].
Available: http://nlpprogress.com/english/constituency_parsing.html
[Accessed May 2019]

[40] "Pandas: Python Data Analysis Library". *Pandas*. 2019. [Online].
Available: https://pandas.pydata.org/
[Accessed February 2019]

[41] "Lexpanded-PPDB: Lexically-Expanded Paraphrase Database". Paraphrasing. 2019.
Available: http://paraphrasing.org/~fujita/resources/lexpanded-PPDB.html
[Accessed March 2019]

[42] "Embedding Projector – Visualization of high dimensional data". Projector Tensorflow. 2019. [Online].
Available: http://projector.tensorflow.org/
[Accessed June 2019]

[43] J. Pennington, R. Socher, C. Manning. "Empirical Methods in Natural Language Processing. GloVe: Global Vectors for Word Representation". 2014. [Online].
Available: http://www.aclweb.org/anthology/D14-1162
[Accessed February 2019]

[44] B. Plank, A. Søgaard, and Y. Goldberg. "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss". *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016. [Online].
Available: https://arxiv.org/abs/1604.05529
[Accessed January 2019]

[45] "Dash: Python framework for building analytical web applications". *Plotly Dash*. 2019.
Available: https://plot.ly/products/dash/
[Accessed April 2019].

[46] "Googletans 2.4.0". *Python Software Foundation*. 2019. [Online].
Available: https://pypi.org/project/googletrans/
[Accessed May 2019]

[47] G. Rachiele. "Tokenization and Parts of Speech(POS) Tagging in Python's NLTK library". *Medium*. 2018. [Online].
Available:https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b
[Accessed February 2019]

[48] J. Santos. "Data classification with neural networks and entropic criteria". 2012
Available:https://www.researchgate.net/publication/37655851_Data_classification_with_neural_networks_and_entropic_criteria
[Accessed May 2019]

[49] S. Sharma. "Activation Functions in Neural Networks". *Towards Data Science*. 2017. [Online].
Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
[Accessed May 2019]

[50] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. "*Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank"*. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013. [Online]
Available: https://www.aclweb.org/anthology/D13-1170
[Accessed January 2019]

[51] "Shiny from R Studio". *Shiny* 2019. [Online].
Available: https://shiny.rstudio.com/
[Accessed May 2019]

[52] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. “Learning sentiment-specific word embedding for twitter sentiment classification”. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014. [Online].
Available: https://www.aclweb.org/anthology/P14-1146
[Accessed January 2019]

[53] S. Tai, R. Socher, and C. Manning. “Improved semantic representations from tree-structured long short-term memory networks”. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2015. [Online].
Available: https://www.aclweb.org/anthology/P15-1150
[Accessed January 2019]

[54] “Tensorflow: An end-to-end open source machine learning platform”. *Tensorflow*. 2019. [Online].
Available: https://www.tensorflow.org/
[Accessed March 2019]

[55] O. Uryupina, B. Plank, A. Severyn, A. Rotondi, and A. Moschitti. “Sentube: A corpus for sentiment analysis on youtube social media”. *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*. 2014. [Online].
Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/180_Paper.pdf
[Accessed January 2019]

[56] Xiang Zhang Google Drive. Amazon Reviews dataset. [Online]
Available:https://drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M
[Accessed January 2019]

[57] X. Zhang, J. Zhao, Y. LeCun. *Character-level convolutional networks for text classification*. In Advances in neural information processing systems , pages 649–657. 2015. [Online].
Available: https://arxiv.org/abs/1509.01626
[Accessed April 2019]

[58] Xiang Zhang Webpage. [Online].
Available: http://xzh.me/
[Accessed January 2019]

[59] T. Young, D. Hazarika, S. Poria, E. Cambria. “Recent Trends in Deep Learning Based Natural Language Processing”. IEEE Computational Intelligence Magazine. 2018. [Online].
Available: https://arxiv.org/abs/1708.02709
[Accessed January 2019]

[60] L. Yu, J. Wang, K. R. Lai, and X. Zhang. “Refining word embeddings for sentiment analysis”. *In Proc. Conf. Empirical Methods Natural Language Processing*, 2017, pp. 545–550. 2017. [Online].
Available: https://dl.acm.org/citation.cfm?id=3186455
[Accessed January 2019]

[61] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and Bo Xu. “Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling”. *In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016. [Online]
Available: https://arxiv.org/abs/1611.06639
[Accessed January 2019]

APPENDIX

APPENDIX

Appendix 1: Planning

This section explores how a company could develop a similar product as the one detailed in this document. To start with, a team with experts on different fields is needed:

- Software Engineer: requirements engineering, project leader.
- Data Scientist: data extraction, model's architecture and configuration, and training.
- Back-end Developer: application's back-end functionalities, integration with model's output and front end.
- Front-end Developer: visualization tools, tables and figures, application user interface.

Tasks have been grouped into 5 groups: Initiation, Planning, Design, Execution and Deployment. By using Microsoft Excel, a Gantt model has been developed, focusing on each task definition, an estimate of the duration to complete it, dependencies between tasks and team members time constrains. As a result, the project is expected to be developed by a team of four members in 24 days.

Figure 37: Project Tasks

	Task ID	Task Name	Duration (days)	Previous Task ID	Assigned to
INITIATION	1	Requirement definition	2	-	Software Engineer
	2	Use case modelling	2	-	Software Engineer, Data Scientist
	3	System requirements specification	1	1, 2	Software Engineer, Back-End Developer
PLANNING	4	Problem definition	1	-	Software Engineer
	5	Project organization and management	1	4	Software Engineer
	6	Alternatives and viability study	2	5	Data Scientist, Back-End Developer, Front-End Developer
	7	Risk analysis	1	5	Software Engineer
	8	System definition	1	4	Back-End Developer
	9	Estimation and prioritization	1	8	Software Engineer
	10	Testing plan	2	8	Back-End Developer, Front-End Developer
	11	Deployment plan	1	9	Software Engineer, Front-End Developer
DESIGN	12	System architecture	2	8	Back-End Developer
	13	UI design	2	8	Front-End Developer
	14	Data Extraction	1	8	Data Scientist
	15	Data Exploration	3	14	Data Scientist
EXECUTION	16	App functionlities development	5	12	Software Engineer, Back-End Developer
	17	App UI implementation	5	13	Front-End Developer
	18	Data preprocessing	1	15	Data Scientist
	19	Model architecture and training	3	18	Data Scientist
	20	Integration and Test	3	16, 17, 19	Software Engineer, Back-End Developer, Front-End Developer
DEPLOYMENT	21	Deployment	2	20	Back-End Developer, Front-End Developer

Figure 38: Gannt Diagram

Task ID	Task Name	Planned Start	Planned Duration	Real Start	Real Duration	PCT Completed	Period 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	Requirement definition	1	2	0	0	0%																								
2	Use case modelling	1	2	0	0	0%																								
3	System requirements specification	3	1	0	0	0%																								
4	Problem definition	4	1	0	0	0%																								
5	Project organization and management	5	1	0	0	0%																								
6	Alternatives and viability study	6	2	0	0	0%																								
7	Risk analysis	6	1	0	0	0%																								
8	System definition	8	1	0	0	0%																								
9	Estimation and prioritization	9	1	0	0	0%																								
10	Testing plan	10	2	0	0	0%																								
11	Deployment plan	10	1	0	0	0%																								
12	System architecture	12	2	0	0	0%																								
13	UI design	12	2	0	0	0%																								
14	Data Extraction	12	1	0	0	0%																								
15	Data Exploration	13	3	0	0	0%																								
16	App functionlities development	14	5	0	0	0%																								
17	App UI implementation	14	5	0	0	0%																								
18	Data preprocessing	16	1	0	0	0%																								
19	Model architecture and training	17	3	0	0	0%																								
20	Integration and Test	20	3	0	0	0%																								
21	Deployment	23	2	0	0	0%																								

Appendix 2: Budgeting

Once an estimation plan has been developed, this section will analyze the costs of the project, taking into consideration the time required per team member as well as software and hardware costs.

Figure 39: Team Cost

Member	Hourly wage	Days	Total Hours	Cost
Software Engineer	20.00 €	18	144	2,880.00 €
Data Scientist	18.00 €	12	96	1,728.00 €
Back-End Developer	18.00 €	18	144	2,592.00 €
Front-End Developer	18.00 €	17	136	2,448.00 €
TOTAL				9,648.00 €

In order to estimate the material costs, I've considered the ones needed when developed this project, taking into consideration that some software licenses are not valid for professional development and is required to purchase the appropriate license.

Figure 40: Software and Hardware Cost

Description	Unit Price	Units	Months	Depreciation	Cost
Laptop ThinkPad T480	1,239.00 €	4	1	25.81 €	103.25 €
GitHub Team License	25.00 €	1	1	- €	25.00 €
Pycharm Proffesional	8.90 €	4	1	- €	35.60 €
Kaggle	- €	1	1	- €	- €
TOTAL					163.85 €

Figure 41: Total Project Cost

Description	Cost
Team Cost	9,648.00 €
Software and Hardware Cost	163.85 €
TOTAL	9,811.85 €

As a result, if a company would like to replicate this project with a similar outcome, the estimated costs would be 9,811.85 €.