Luís
Santos

Recolha, extração e classificação de opiniões sobre aplicações lúdicas para saúde e bem-estar

Collection, extraction and classification of opinions on playful applications for health and well-being

Luís
Santos

Recolha, extração e classificação de opiniões sobre aplicações lúdicas para saúde e bem-estar

Collection, extraction and classification of opinions on playful applications for health and well-being

"*The greatest challenge to any thinker is stating the problem in a way that will allow a solution*"

— Bertrand Russell

Universidade de Aveiro
2022

Luís
Santos

Recolha, extração e classificação de opiniões sobre aplicações lúdicas para saúde e bem-estar

Collection, extraction and classification of opinions on playful applications for health and well-being

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Sérgio Guilherme Aleixo de Matos, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutora Pétia Georgieva Georgieva
Professora Associada, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Rui Pedro Sanches de Castro Lopes
Professor Coordenador, Instituto Politécnico de Bragança

Prof. Doutor Sérgio Guilherme Aleixo de Matos
Professor Auxiliar em Regime Laboral, Universidade de Aveiro

agradecimentos / acknowledgements

Aproveito esta secção para começar por agradecer às pessoas que sem elas, não teria a possibilidade de chegar aqui, os meus Pais. A eles e às minhas irmãs agradeço todo o sacrificio e trabalho que tiveram de se sujeitar para eu poder estar a realizar este documento e por todo o apoio. Agradecer ao meu Tio e aos meus Avós por todo o bem que fizeram por mim. Um especial agradecimento à minha namorada por estar sempre ao meu lado para me aturar e pelo seu apoio incansável. Deixar também uma palavra de agradecimento à restante família que esteve presente e que me apoiou.

Aos meus amigos, que estiveram sempre do meu lado, por tudo o que me ensinaram, que me fizeram crescer e por todas as dificuldades que passaram comigo provinientes de consequências que uma vida académica acarreta. Agradecer aos meus colegas, docentes e orientador que cruzaram comigo durante este percurso académico, por toda a partilha de conhecimento e experiências.

Por fim, deixar uma palavra de apreciação à Universidade de Aveiro que permite reunir todas as condições para uma excelente formação.

Palavras Chave

Análise de Sentimento Baseada em Aspetos, Classificação, Extração, Reviews, Deep Learning, Transformers, BERT, LCF-ATEPC.

Resumo

Atualmente, as aplicações móveis fazem parte da vida de qualquer pessoa que possua um smartphone. Com a evolução tecnológica, novas aplicações surgem com novas funcionalidades, o que traz uma maior exigência por parte dos utilizadores quando usam uma aplicação. Numa altura em que a saúde e bem-estar são uma prioridade, existem cada vez mais aplicações com o intuito de providenciar uma melhor experiência ao utilizador, não só a nível de monitorização de saúde, mas também de uma experiência agradável em termos de entertenimento e bem estar. Contudo, existem ainda algumas limitações no que toca à experiência e usabilidade do utilizador. O que melhor pode traduzir a satisfação e experiência do utilizador são as reviews das aplicações. Assim sendo, para ter uma perceção dos aspetos mais relevantes das atuais aplicações, foi feita uma recolha das reviews e respetivas classificações.

O objetivo desta tese consiste no desenvolvimento de um sistema que permita apresentar os aspetos mais relevantes de uma determinada aplicação de saúde e bem estar, após a recolha das *reviews* e posterior extração dos aspetos e classificação dos mesmos. No processo de recolha de reviews, foram usadas duas bibliotecas em Python, uma relativa à Google Play Store e outra à App Store, que providenciam métodos para extrair dados relativamente a uma aplicação. Para a extração e classificação dos aspetos, o modelo LCF-ATEPC foi o escolhido dada a sua performance em estudos de análise de sentimento baseada em aspectos.

Keywords

Aspect-Based Sentiment Analysis, Classification, Extraction, Reviews, Deep Learning, Transformers, BERT, LCF-ATEPC.

Abstract

Nowadays, mobile apps are part of the life of anyone who owns a smartphone. With technological evolution, new apps come with new features, which brings a greater demand from users when using an application. Moreover, at a time when health and well-being are a priority, more and more apps provide a better user experience, not only in terms of health monitoring but also a pleasant experience in terms of entertainment and well-being. However, there are still some limitations regarding user experience and usability. What can best translate user satisfaction and experience are application reviews. Therefore, to have a perception of the most relevant aspects of the current applications, a collection of reviews and respective classifications was performed.

This thesis aims to develop a system that allows the presentation of the most relevant aspects of a given health and wellness application after collecting the reviews and later extracting the aspects and classifying them. In the reviews collection task, two Python libraries, one for the Google Play Store and one for the App Store, provide methods for extracting data about an application. For the extraction and classification of aspects, the LCF-ATEPC model was chosen given its performance in aspects-based sentiment analysis studies.

Contents

List of Figures

List of Tables

Acronyms

ABSA	Aspect-Based Sentiment Analysis
API	Application Programming Interface
APC	Aspect Polarity Classification
ATE	Aspect Term Extraction
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CBoW	Continuous Bag of Words
CDM	Context-Feature Dynamic Mask
CDW	Context-Feature Dynamic Weighting
CNNs	Convolutional Neural Networks
CNNs	Convolutional Neural Network
DAs	Denoising Autoencoders
DBNs	Deep Belief Networks
DRL	Deep Reinforcement Learning
FIL	Feature Interactive Learning
GAN	Generative Adverisal Networks
GCFG	Global Context Feature Generator
GRNN	Gated Recurrent Neural Networks
GRU	Gated recurrent units
JSON	JavaScript Object Notation
LCF	Local Context Focus
LCFG	Local Context Feature Generator
LSTM	Long Short-Term Memory
LSTMs	Long Short-Term Memory Networks
MHSA	Multi-Head Self-Attention
MLM	Masked Language Model
NLP	Natural Language Processing
NSP	Next Sequence Prediction
RAE	Recursive Autoencoders
RBMs	Restricted Boltzmann Machines
RNTN	Recursive Neural Tensor Networks
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RNNs	Recurrent Neural Networks
RvNN	Recursive Neural Network
RvNNs	Recursive Neural Networks
SRD	Semantic-Relative Distance
SVM	Support Vector Machine
UX	User Experience
UI	User Interface

CHAPTER 1

Introduction

Nowadays, we live in the "online age" in which we are constantly connected to technology and the internet from the moment we get up until we go to bed. It is possible to say that we are highly dependent on technology and internet connection to "survive" these days. The evolution was towards automating and reducing the effort necessary to have or do something as much as possible. Day-to-day activities that used to require a certain amount of logistics on the part of people are now just a few clicks away. One of the driving tools of this evolution was undoubtedly smartphones. Nowadays, through a smartphone, we have access to a world full of services and applications where it is possible to do almost everything without having to travel and communicate with anyone. In addition, today, there is also a world of exposure. Social networks, forums, and other platforms enable a person to express their opinion of their own free will. Nowadays, online reviews can play a crucial role in the growth and recognition of a product. Often, a user's first perception of service is through the reviews made of the product, which can influence whether or not to acquire the same. Sentiment analysis has been applied to several domains, not only in social networks but also in product reviews, to understand online users' opinions influence [1].

One of the markets that emerged from this dependence on applications was the market for so-called mHealth apps, which represent medical practices supported by mobile devices that positively impact patients' lives. In addition, fitness and health monitoring apps are increasing on app stores and in users' daily lives. However, although several solutions explore playful aspects in an attempt to obtain positive effects on health and well-being, several factors limit the adoption and continuation of these solutions, namely in terms of usability and user experience [2][3].

This project aims to collect and analyze comments in public forums related to games and other recreational applications focused on health, well-being, and quality of life. Having this in mind, creating tools for extracting and classifying relevant information, including identifying

usability and User Experience (UX) aspects mentioned in the comment and determining the polarity (positive/negative) of the comment, is required.
The collection of information regarding the aspects through the reviews and their classification will be presented in a web app that will allow to add an application and observe the most relevant positive and negative aspects.

Scrapers from Google Play Store and App Store are responsible for extracting reviews from app stores. The classification and extraction of review aspects are done through a deep learning model based on the transformers architecture, LCF-ATEPC, which provides a method for inference. Furthermore, an architecture based on micro-services ensures that the extracted information is stored in a MongoDB database and presented through an Angular-based User Interface (UI).

Chapter 2 presents background on NLP, sentiment analysis, its classic approaches, deep learning, and respective approaches in sentiment analysis, and finally, a survey of current approaches in aspect-based sentiment analysis studies. Chapter 3 details the chosen model for classifying and extracting aspects from reviews. In chapter 4, the developed architecture of the system is presented. Chapter 5 illustrates the classification model training results, as well as the views of the developed web app. Finally, in chapter 6, the conclusions are presented, taking into account the objective of the project and the final result, and future improvements.

CHAPTER 2

Background

The amount of natural language text that is available in electronic form is truly staggering, and is increasing every day.

Online reviews are an important factor when it comes to apps. Through them it is possible to understand the user feedback, the benefits or the missing features that users want to be implemented. This way is possible to have a direct contact between application developers and it is possible to have a bigger number of satisfied users.

2.1 Natural Language Processing

It is possible to define Natural Language as the communication used by humans daily, whether in person or online. In order to compute the natural language, understand the context of a conversation, syntactic and semantic meanings, and sentiment analysis, Natural Language Processing (NLP) is primordial.

"NLP is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications." [4].

Nadkarni P et al. [5] in their study about NLP, define NLP tasks in two levels: low-level and high-level tasks.

Low-level tasks are:

- Sentence Boundary Detection
- Tokenization - stands for identifying individual tokens, a word or punctuation, within a sentence.
- POS tagging - Part-Of-Speech assignment to individual words to perform morphological decomposition of compound words.
- Chunking - Related to shallow parsing, stands by identifying phrases from constituent part-of-speech tagged tokens.

On the other hand, High-level tasks include:

- Spelling/grammatical error identification and recovery.

- NER - Named Entity Recognition, identifies and classifies named entities in text. For example, “Jonh was born in 1990”, “Jonh” is detected and classified as a person name and “1990” as a temporal expression.
- WSD - Word Sense Disambiguation, determines a homograph’s correct meaning.
- Negation and uncertainty identification - infers whether an entity is implicitly or explicitly denied or not, and quantifies the uncertainty of the inference.
- Relationship Extraction - responsible for determining relationships between entities or events and and temporal inferences/relationship extraction.
- Information Extraction - identification of problem-specific information and its transformation into problem-specific structured form.

Artificial Intelligence uses machine learning or deep learning approaches to deal with all this complexity of interpretation.
Nowadays, NLP can be essential in business, analyzing customers’ reviews, and detecting fake news. It can even have a political impact by analyzing posts on social networks and understanding customers’ opinions through sentiment analysis.

2.2 Sentiment Analysis

Every day, millions of online users express opinions and emotions about various subjects such as politics, products, brands, mobile apps, hotels, and restaurants. These expressions are made through social networks and service platforms (Google Play Store, Apple Store, TripAdvisor) or forums. Therefore, it is interesting to understand the essential role and influence that reviews can have on the online community, product vendors, customers, and politics. One of the NLP tasks is to study the sentiments expressed by users’ reviews/posts, known as sentiment analysis.

Andrés Montoyo et al. define sentiment analysis as “the task of detecting, extracting and classifying opinions and sentiments concerning different topics, as expressed in textual input.” [6]. This way, it is possible to admit that every textual input has a sentiment behind it. Therefore, depending on the context, a set of reviews/posts allows drawing several conclusions, not only concerning one person but a group of people, which has become increasingly important in the online world.
The power of social networks allows the spreading of information that reaches many people more effectively. For example, in the 2008 US Presidential elections, Obama gained 118,000 new followers on Twitter in 17 months due to his high number of tweets during the campaign period [7][8]. One of the purposes of sentiment analysis is that it can greatly impact society in terms of political and governance questions because it facilitates politicians to analyze public sentiments [9]. It has been used as an election predictor, where posts on social networks are analyzed, and it is possible to understand which candidate has more positive or negative comments concerning the posts’ polarity, as Ussama Yaqub and Vijayalakshmi Atluri et al. performed in the 2016 US presidential elections with an exploratory sentiment base analysis

of Twitter data that was gathered both before and after the election day [7]. One interesting fact was that most of the pre-election polls, during ten days, predicted a Hillary Clinton victory as the winner, but in the subsequent sentiment analysis, during these ten days, Donald Trump was leading Hillary Clinton with a lower average negative sentiment of tweets[7].

Another purpose of sentiment analysis is to automatically determine the expressive direction of user reviews[10].
Nowadays, more and more online reviews influence business. That is why many of the current services provide users platforms to leave their opinion and evaluate the provision of the service. A decisive factor when someone wants to benefit from a service/product is the online reviews to understand other users' experiences and have a better perception of the quality of the service/product. So, in terms of business, sentiment analysis allows companies to better perceive how much customers are satisfied or not with their services and how they can improve them [9]. It has been applied to various kinds of business, such as restaurants [11], hotels [12] and airline companies. For example, Ayat Zaki Ahmed and Manuel Rodríguez-Díaz did their study in which the purpose was to understand the critical aspects related to passenger satisfaction through TripAdvisor reviews. They concluded that the labels extracted from the titles of comments help to determine customers' sentiments in their full comments [13] and it is possible through labels to understand what satisfies customers the most.

Besides online website reviews, sentiment analysis is applied to mobile app reviews. App stores allow users to express their opinions and classify their experiences when using the app. Through this feedback, it is possible to provide important information to developers about the most varied issues regarding app usability and functionality or even ask for new features [14]. Mobile applications are increasingly a constant in the daily lives of people who, through their smartphones, can have applications that provide entertainment, information, education, service provision, and health care.

Mobile healthcare apps are increasingly common on smartphones [15].As noticeable in the figure below, the number of downloads of mobile health care apps in 2020 increased significantly compared with 2019. Covid-19 global pandemic and the consequent lockdowns made consumers rethink new ways of exercise and have a healthy lifestyle [16].

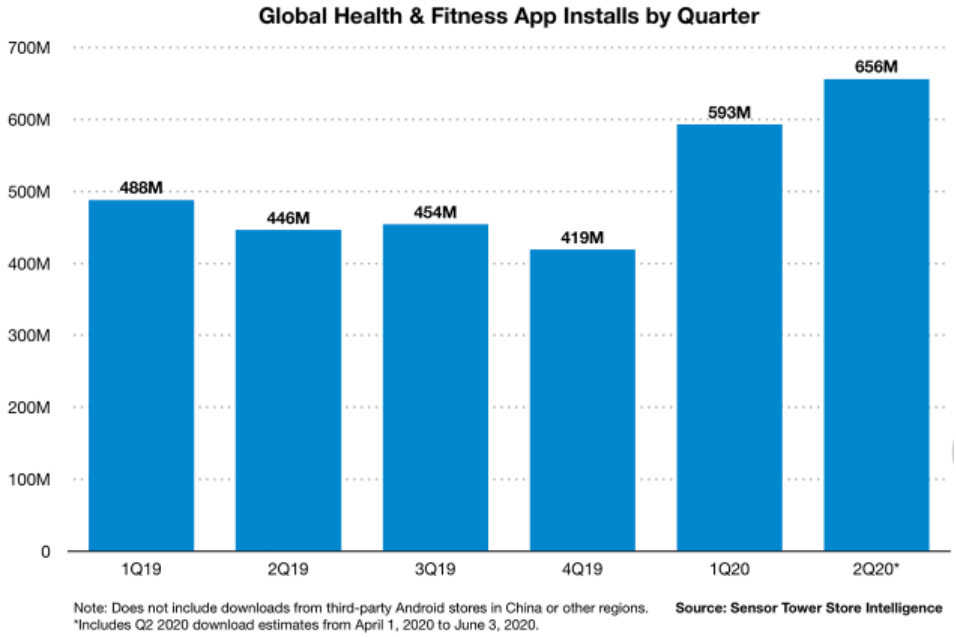


Figure 2.1: Global Health & Fitness App Installs by Quarter during 2019 and 2020. Available at `https://sensortower.com/blog/health-and-fitness-app-record-download-growth`. Accessed on May 4, 2021

Paschou et al. to understand the effectiveness of mhealth care apps, analyzed and classified several health apps. The results show that applications can facilitate the communication process between the doctor and the patient and, consequently, provide the patient greater security knowing that the doctor is aware of their health condition [17].

Due to their relevance nowadays, mhealth care apps are also an important subject to sentiment analysis. Marlene Camacho-Rivera, et al., from June 23, 2020, until July 21, 2020, collected reviews of asthma mobile apps that are publicly available to understand their usability, features, and content through positive and negative user ratings. Three hundred seventy-three reviews were examined across ten apps. From the sentiment analysis, it was possible to conclude that the key features of the highly-rated apps in asthma management were tracking peak flow readings, asthma symptom monitoring, and action plans. In terms of functionality, the key feature found was the app's ease of use, and the most common reported issues were the loss of data and app crashing [18]. Through this example, it is possible to realize how sentiment analysis is and how important and accurate it can be.

However, despite its usefulness, sentiment analysis faces some challenges: people express opinions differently; the same word can represent a positive or negative opinion depending on the situation and context; a single word can make the difference and change the polarity of an opinion; the fact that there is the possibility of having different opinions in the same sentence and the presence of irony and sarcasm can be a difficult task for the computer to understand. [19][20].

Sentiment analysis can be applied at different levels, with different approaches, to classify sentiments.

2.2.1 Sentiment Analysis Approaches

Sentiment Analysis aims to identify sentiments and classify their polarity [1]. In [1], three main classification levels in sentiment analysis are defined: document-level, sentence-level, and aspect-level. Document-level, as the name implies, classifies an opinion document (e.g., online review) to understand if it expresses positive or negative sentiment. Sentence-level classifies sentiments in each sentence. Lastly, aspect-level classifies sentiments relative to specific aspects of entities. For example, in a restaurant review like *"In this restaurant food is good but the waiter wasn't that kind"*, aspect-based sentiment analysis should identify the restaurant as the entity, the food and the waiter as the aspects, and classify the sentiment expressed on the food as positive and on the waiter as negative.

Regarding sentiment classification, there are two main approaches, lexicon-based and machine learning. Besides that, there is the hybrid approach that combines both machine learning and lexicon-based approaches [19][21][22].

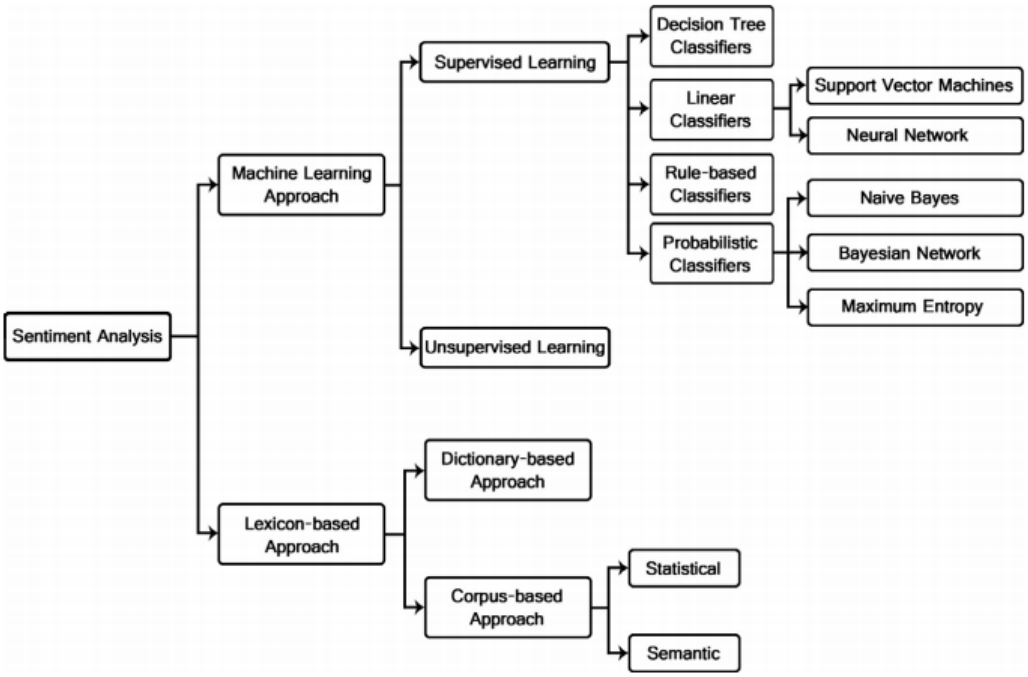


Figure 2.2: Sentiment classification techniques [1].

Figure 2.2 shows the methods used for sentiment classification inside lexicon-based and machine learning approaches. The lexicon-based uses a collection of words used to express positive or negative sentiments named *opinion lexicon*, which already has an attributed score (positive, negative, or neutral). The sentiment polarity of the text depends on the ratio between positive and negative terms [19][23][24]. The machine learning approach automatically classifies sentiments by training an algorithm with known data sets before classifying the intended text [25].

Inside lexicon-based, there is the dictionary-based and corpus-based approach. The

dictionary-based approach consists of manually collecting a set of words, seed words, which already have a known orientation, and then searching in dictionaries and thesaurus for synonyms and antonyms. The new ones are added to the seed list until no new words are found [21][24]. The disadvantage of this approach is that it has difficulty finding opinions in specific domains and contexts [21]. The corpus-based approach solved the dictionary-based domain and context problem by finding opinion words in a large corpus[1] [22]. It can be done by applying statistical or semantic methods. Statistical methods consist of the percentage of the word's polarity in the texts. If it mostly appears in positive text, it will have a positive polarity and vice-versa [24]. Semantic methods, as the name implies, calculate the polarity of a word based on semantically close words. A lexical database like *WordNet* which contains semantic relations between words, can be used to calculate sentiment polarities [21].

As mentioned before, the machine learning approach consists of automatic classification and relies on text features [24]. This approach is divided into supervised and unsupervised learning.
Supervised learning methods rely on many training documents [21]. Regarding classification, the word features are compared and labeled with the class that matches the most [24]. The limitation of supervised learning is that it is dependent on the quality and amount of data. If the data is biased or insufficient, it can produce unexpected results [22]. According to Figure 2.2, the supervised learning classifiers for sentiment classification are decision tree, linear, and rule-based classifiers.

Decision Tree classifiers divide the training data hierarchically and recursively depending on a condition. It happens until the leaf nodes contain certain minimum numbers of records. The condition can be one or more words presence or absence [21][24]. For example, the presence or absence of the positive word "great" in text, could be the condition to perform the classification.

Linear classifiers are based on the value of a linear combination of the characteristics:

$$LP = W.C + S \tag{2.1}$$

Transposing to sentiment analysis, the linear product (LP) is the scalar product between word frequency (W) and the linear coefficient vector (C) plus the scalar value (S). This predictor is defined as a hyper-plane that divides the search space [24]. The two main linear classifiers used in sentiment analysis are SVM and Neural Networks.

- Support Vector Machines
  The main goal of SVMs is to find the best hyper-plane which completely separates different classes. During the execution of the SVM training algorithm, the data is assigned to one of the classes [24]. The Figure 2.3 shows an example of how SVM works.

[1] A collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject.

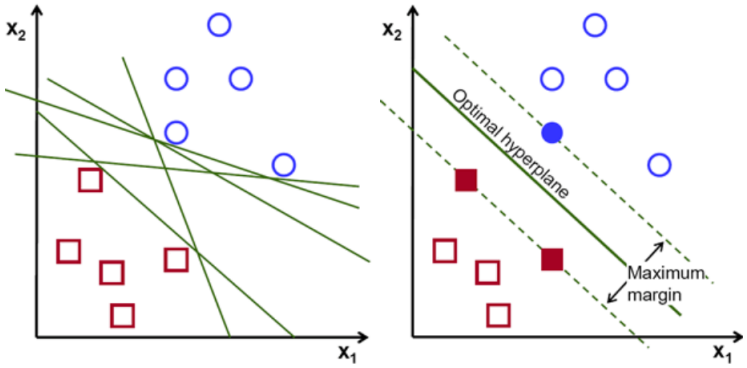


Figure 2.3: SVM linear classifier.
`https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c`.
Accessed on June 5, 2021

The example of Figure 2.3 allows an understanding of how SVMs work. The best hyperplane is the one that provides the maximum margin between the two classes. This way, in terms of classification, it is easier to identify to which class the new data belongs.

- Neural Networks
  Neural networks consist of a group of interconnected neurons where each neuron computes the function of a set of inputs and associated weights [24].Given the previous equation, $LP = W.C + S$, in terms of sentiment analysis, it is possible to match W as the word frequency in the document, C as the weights associated with each neuron, and S as the bias. Neural networks will be described in detail in the next section, 2.3

As the name suggests, rule-based classifiers use rules to classify data. These can be expressed as "IF condition THEN decision." [24].
Three of the most famous probabilistic classifiers are:

- Naive Bayes
  It is one of the most commonly used classifiers. When applied to sentiment analysis, it uses the Bag of Words (BoW) feature extraction, which ignores the position of the words in the documents and is based on word frequencies. The model uses the Bayes theorem to compute the probability that a given set of features is a part of a particular label.
  $$P(label|features) = \frac{P(label) * P(features|label)}{P(features)} \tag{2.2}$$
  where P(label) is the prior probability of label, P(features|label) is the prior probability that a given feature set is being classified as a label, and P(features) is the prior probability that a feature set will occur.

- Bayesian Network
  In contrast to the Naive Bayes classifier, the Bayesian Network classifier assumes that all the features are strongly dependent on each other. It is a directed acyclic graph where

the nodes represent random variables and edges represent conditional dependencies.

- Maximum Entropy
  Through encoding, the Maximum Entropy classifier converts labeled feature sets into vectors. Those are used to calculate weights for each feature to determine the most likely label for a feature set [21]. The probability of label can be calculated using the following equation:

$$P(fs|label) = \frac{dotprod(weights, encode(fs, label)}{sum(dotprod(weights, encode(fs, l))forlinlabels)} \tag{2.3}$$

Unsupervised Learning overcomes the difficulty of creating labeled training documents. It consists of training data without supervision, meaning there is no input labeled data to compare and depends on previous experience to improve its accuracy [26]. The use of topic models to extract topics from unlabeled text and turned into features [27], splitting documents into sentences and categorize each one using a keywords list and sentence similar measure [28] are some examples of sentiment analysis approaches [22] [21]. The need for a large amount of data is the limitation of this learning method [22].

2.3 Deep Learning

Machine Learning enables the computer to learn automatically by itself, extracting patterns from raw data or training labeled data. It brings advantages that allow new functionalities and applications in the IT World. For example, it is possible to automatically identify an email as spam or not spam using the Naive Bayes algorithm [29]. Machine Learning algorithms have proved to have highly effective results compared to traditional methods, and they can solve intellectually difficult problems for human beings. As mentioned before, machine learning can be applied to many tasks and solve many problems. However, the effectiveness of the algorithms highly depends on the representation of the data they are given [29]. Many of these tasks can have the expected performance if the right features are extracted and provided to the algorithms. However, in some tasks, it is difficult to identify what features to extract. For example, detecting objects in images can be tricky due to image details like shadows, shapes, and angles.
When it comes to extracting raw data from images, such as numbers in an image, many details like shadows and shapes can lead to wrong feature extraction. Deep Learning solves this problem by reaching high complexity representations and introducing simpler ones. It is a machine learning method based on artificial neural networks. It makes use of multiple layers of "neurons" where there is a nonlinear processing unit for feature extraction and transformation [30][31]. The lower layers start by learning the simple features, whereas higher layers learn more complex features derived from lower layers, increasing the abstraction level [30][31]. Figure 2.4 shows an example of a simple deep neural network.

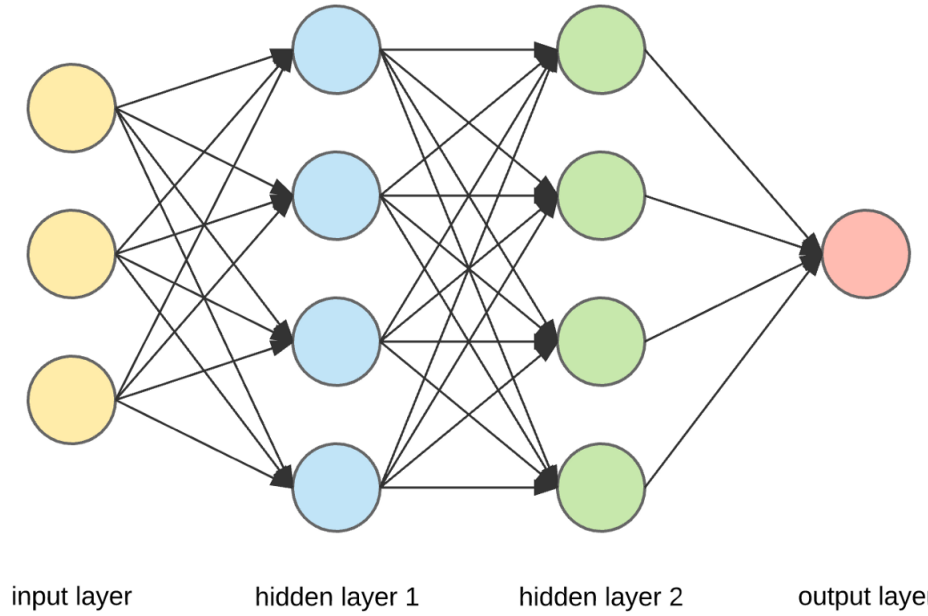


Figure 2.4: Simple Deep Neural Network Example.
`https://towardsdatascience.com/converting-a-simple-\`
`deep-learning-model-from-pytorch-to-tensorflow-b6b353351f5d`.
Accessed on September 17, 2021

The inputs are converted into numerical vectors and fed into the input layer. Then, the computation starts with neurons. Each one is characterized by its weight, bias, and activation function. Through biases and weights, neurons perform a linear transformation on the inputs as described in equation 2.4

$$a = \sum_{i=1} w_i x_i + b \tag{2.4}$$

The output of each neuron is then returned by an activation function that is a non-linear one, and it is applied to the previous result:

$$s(a) = s\left(\sum_{i=1} w_i x_i + b\right) \tag{2.5}$$

Several activation functions can be applied. Some of the most common are the sigmoid function, hyperbolic tangent function, or rectified linear function.

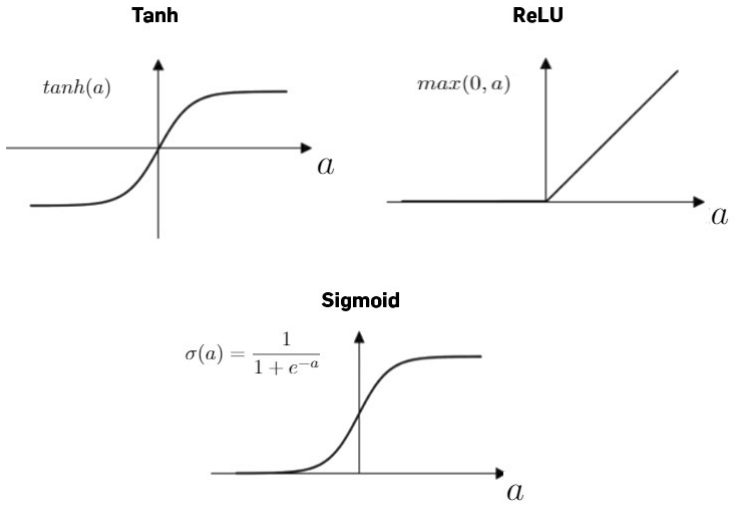


Figure 2.5: Main activation functions representation.
Adapted `https://www.programmersought.com/article/25143220932/`
Accessed on September 27, 2021

The Sigmoid function sets the input values in a range between 0 and 1. Tanh's function is very similar to Sigmoid's. The difference is in the range in which the inputs are set, between -1 and 1. The Rectified Linear Unit (ReLU) function has some differences. It is less susceptible to vanishing gradients[2] and outputs 0 if the value is negative. Otherwise, it returns the input value [31].

The computation of layer l is composed of a matrix W, with all weights where each row corresponds to the connection between one layer and a particular neuron in the next layer, and the vectors a and b with layer activations and biases, respectively [30]. An activation function s is then applied like it is described in the following equation:

$$l = s(Wa + b) \tag{2.6}$$

The softmax function generally calculates the output layer, which assigns probabilities to all labels/classes by normalizing all the input values, returned by the last hidden layer, to the value between 0 and 1. The formula is as follows:

$$y_i = softmax(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.7}$$

A loss function for the softmax output (equation 2.7) is employed as the output comes out. It calculates the loss between the actual output and the expected one [30][32]. In other words, it tells how far the network is from predicting the expected output. Considering $x = x_1, ..., x_n$ as the input, $y = y_1, ..., y_n$ as the output from the deep neural network algorithm, $\hat{y} = \hat{y}_1, ..., \hat{y}_n$ as the actual labels, and θ as the average loss, the goal is to estimate a function $y = f(x)$ to match the inputs with the correct label. The $\mathcal{L}$ value represents the loss when predicting y concerning $\hat{y}$.

[2]It describes the situation where a deep multilayer feed-forward network or a recurrent neural network is unable to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x_i;\theta), y) \tag{2.8}$$

The training process is about minimizing the loss function to improve the network accuracy [32]. Optimization algorithms can update the parameters based on the expected value of the loss function. The gradient is the most commonly used property to optimize the training process [29].

There are optimization algorithms based on the gradient that simultaneously process the entire training set (deterministic or batch gradient methods) and the ones that only process one training example at a time (stochastic or online methods). However, most deep learning algorithms use minibatch or minibatch stochastic methods, which process a few training examples [29]. Gradient descent is a method to converge towards some local minimum of a loss function. Stochastic gradient descent is a minibatch stochastic method and one of the most common approaches to optimizing the training algorithms [32]. It allows obtaining the expected gradient, taking the average gradient of the randomly selected training subsets (minibatches).

Backpropagation is an algorithm for computing and applying the gradient concerning the parameters [32]. After each subset, each layer updates and adjusts the weights and biases accordingly [31]. Those changes lead the network to a step that decreases the loss function most quickly. That is how a deep neural network learns.

Deep Learning has produced state-of-the-art results in many application domains, including NLP, allowing it to solve some of its challenges.

The following subsection will describe in detail deep learning approaches for sentiment analysis.

2.3.1 Deep Learning Approaches for Sentiment Analysis

As mentioned before, deep learning results outperform the traditional methods described in 2.2.1 and the NLP domain was not an exception. The following figure (2.6) shows the difference between classic NLP processes and the ones based on deep learning. Deep learning allows for automatic extraction of features, better performance, and a greater abstraction in terms of context, which can make a difference when it comes to the sentiment polarity [33][34][35].

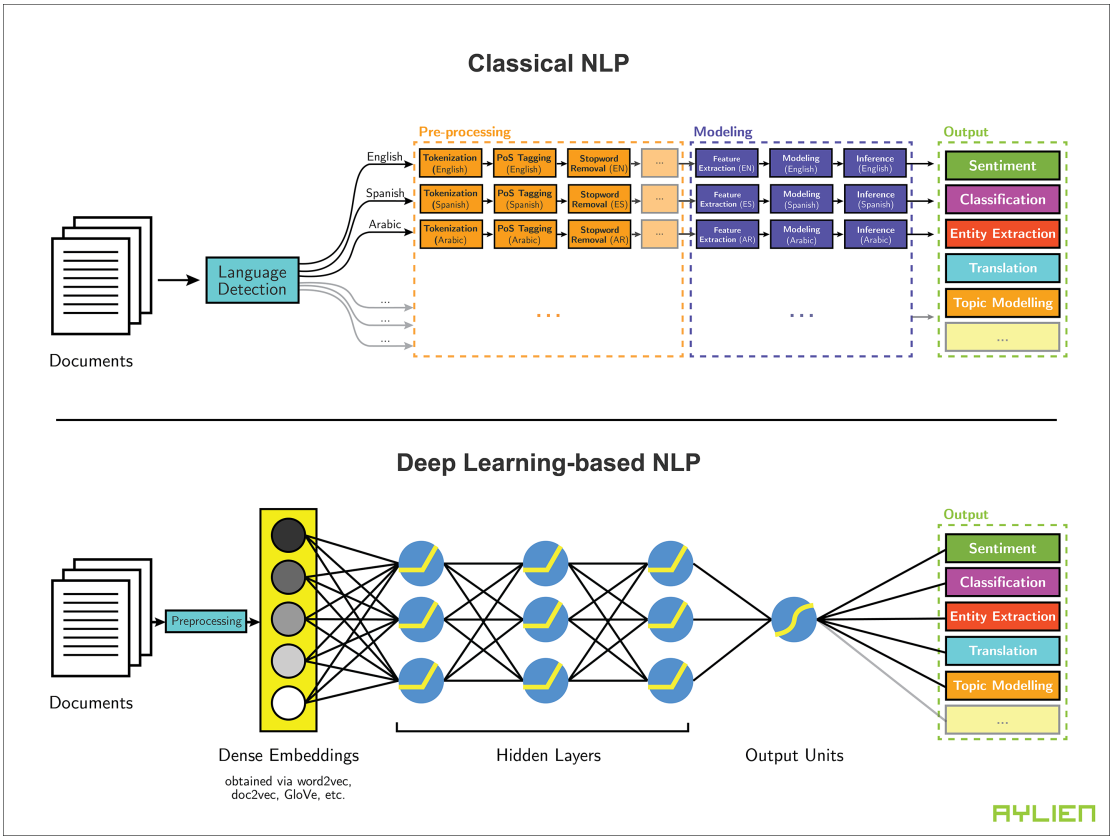


Figure 2.6: Classic vs Deep Learning NLP processes
`https://s3.amazonaws.com/aylien-main/misc/blog/images/nlp-language-dependence-small.png`.

Most deep learning-based NLP processes contain a dense/word embeddings layer. It consists in representing words in the form of vectors where words with similar meanings have similar vectors [31][33]. Two of the most common word embeddings models are word2vec [3] and GloVe[4]. Word2Vec uses a neural network to learn word associations from a corpus and create word embeddings. It contains the Continuous Bag of Words (CBoW) model that predicts the target word from surrounding words and the Skip-gram that predicts the surrounding words through the target word. GloVe, on the other hand, uses a words co-occurrence frequency matrix in a corpus to train and obtain vector representations. This word embedding layer successfully improved deep learning approaches' performance in sentiment analysis [33].

Deep Learning approaches for sentiment analysis can be mainly classified into seven types: Unsupervised pre-trained networks, Convolutional Neural Network (CNNs), Recurrent Neural Network (RNN), Recursive Neural Network (RvNN), Deep reinforcement learning, and Hybrid deep neural networks, as demonstrated in the following Figure 2.7.

[3]https://code.google.com/archive/p/word2vec/
[4]https://nlp.stanford.edu/projects/glove/

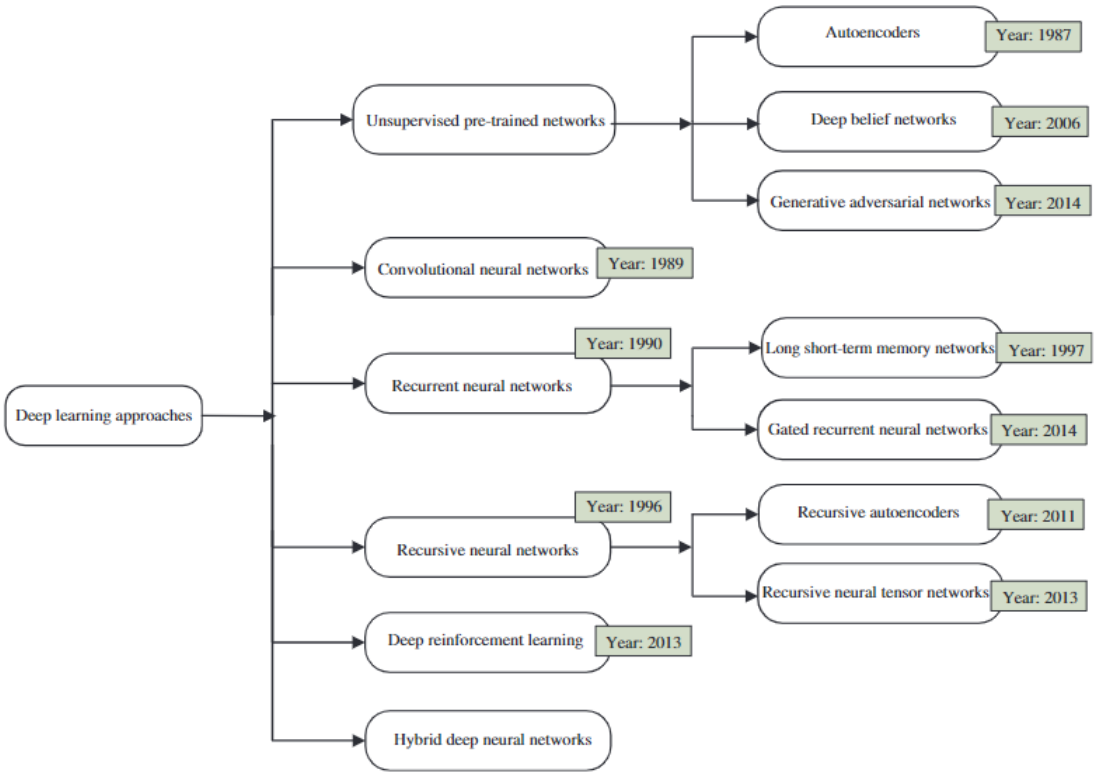


Figure 2.7: Deep learning approaches for sentiment analysis [33].

Unsupervised Pre-Trained Networks

Unsupervised pre-trained networks perform the pre-training of the layers of a neural network with unlabeled data [33]. It allows the network to start the training with non-random values and converge quickly [33].

Autoencoders map the inputs into the outputs. The goal is to obtain a clean representation of the input. The training process allows for minimizing the error in that representation. In terms of sentiment analysis, a variant of autoencoders, Denoising Autoencoders (DAs), are used and applied in domain adaptation [31][33].

Deep Belief Networks (DBNs) are composed of a set of Restricted Boltzmann Machines (RBMs) [33]. It consists of a two-layer network, a visible layer, and a hidden layer [33]. The output from the hidden layer is used as the input of another RBM. That is how a deep belief network is built. Regarding classification, DBNs can be used as feature extractors within a supervised network to classify based on the extracted features [36]. The advantage of DBNs is the potential to exploit a large amount of unlabeled data, but on the other hand, the training process is difficult [33].

Generative Adverisal Networks (GAN) consist of two models competing against each other, one named discriminator (D) and the other generator (G). The goal is for G to generate examples to fool D, and D has to discriminate whether it comes from G (fake) or the training set (real) [33]. The limitation of GAN is the training process [33].

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the main approaches to computer vision [31]. Based on the animal visual cortex[5], it consists of network with an input layer, convolutional layers, pooling layers, and a classification layer [30][33]. The convolutional layer is responsible for detecting features. The pooling layer will extract the relevant features detected by the convolutional layer, and the classification layer is a fully connected network with a classifier [33].

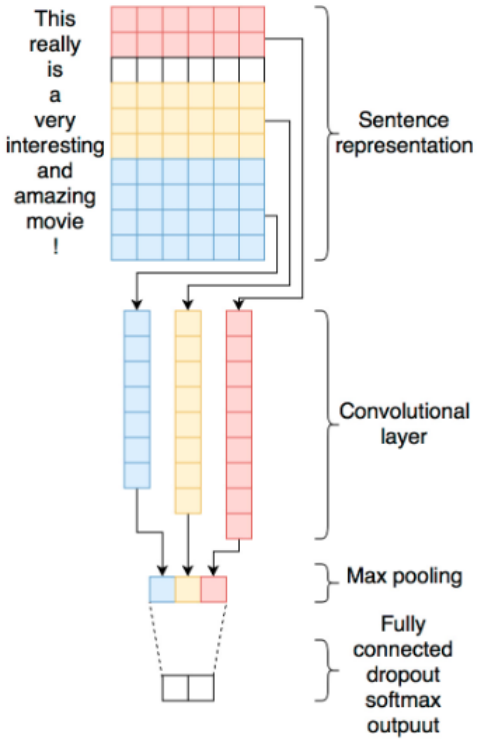


Figure 2.8: Example of a Convolutional Neural Network [37].

As mentioned in [30], CNNs achieved some interesting performances in aspect-based sentiment analysis in the consumer review domain. The main motivation is the assumption that there are keywords that can identify the aspect and determine its polarity independently of its position. Thus, one of the advantages of CNNs is the local feature extraction. The main disadvantage is that without expensive computing, it does not perform wisely on long-term dependencies [30][33].

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) arise in order to model the long-term dependencies in sequential inputs (documents, sentences) [33]. What distinguishes RNNs from previous approaches is their internal memory which relies on the fact that each output of the sequence depends on the previous computations [30][33].

[5]primary cortical region of the brain that processes visual information relayed from the retinas.

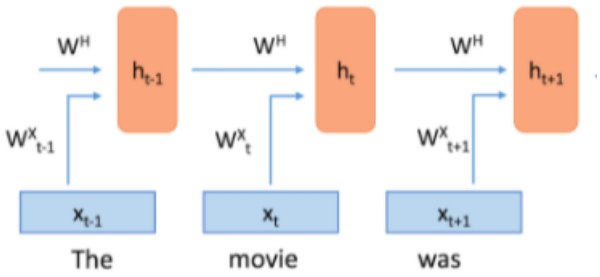


Figure 2.9: Example of a Recurrent Neural Network.
Adapted from `https://www.oreilly.com/content/perform-sentiment-analysis-with-lstms-using-tensorflow/`
Accessed on December 14, 2021

Figure 2.9 represents a simple example of an RNN.The x_t is the input at the time step t, h that, at a high level, summarizes all of the information seen in the previous time steps and W^H the weight matrix of the hidden state h which is the same for each step. The fact that the weight matrix does not change along the steps allows one to apply the same task in each and, consequently, have a better understanding of the input context. But, on the other hand, it can lead to one of the disadvantages of RNNs, the potential occurrence of vanishing gradient or exploding gradient, thus, linked to a small or big long-term dependencies range, respectively [30][31] [33]. To overcome this limitation, long short-term memory networks came up.

Long Short-Term Memory Networks (LSTMs), introduce a mechanism that allows to retain information for a long period of time and discard the worthless one. It is composed by four main components: a cell state, forget gate, an input gate, and an output gate, as shown in Figure 2.10.

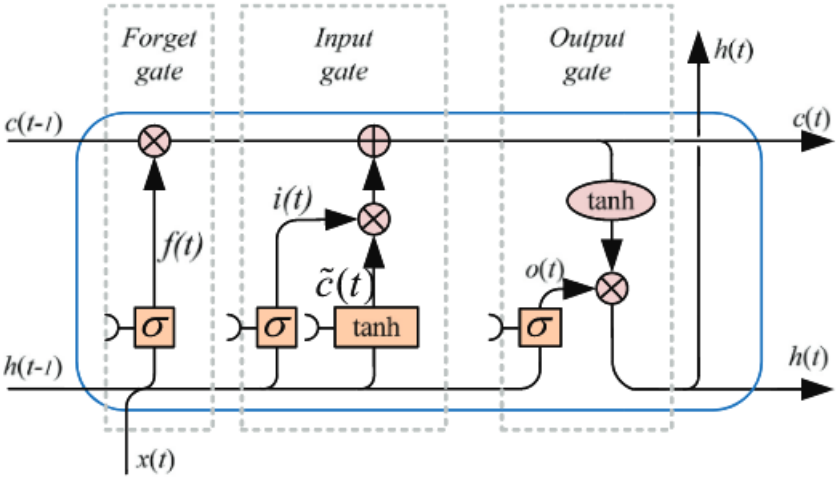


Figure 2.10: Basic Long Short-Term Memory Network Structure [38].

The cell state $c(t)$ transports data throughout the steps. Then, LSTM can change it through internal gates. Forget gate depending on $h(t-1)$ and $x(t)$, will keep or remove data

from the cell state, which helps control gradient. Input gate calculates new possible values to update the cell state, $i(t)$ gives the values that will be updated, and $\tilde{c}(t)$ represents the new candidate values for the cell state. The computation done by these two gates will update the previous cell state $c(t-1)$ into the new cell state $c(t)$. Finally, the output gate processes the output to the next state by filtering data from the cell state, where $o(t)$ represents the filter applied to the cell state values [31]. The negative aspect of LSTMs is their complex structure [33].

Gated Recurrent Neural Networks (GRNN), in contrast to Long Short-Term Memory (LSTM), present a gate mechanism with only two components called Gated recurrent units (GRU). GRU consists of two gates, a reset gate and an update gate. The reset gate is responsible for the computation of the previous inputs and the new ones [33]. Dubbed update gate, in turn, decides what information to keep [33].

RNNs have been applied in several studies, which even led to the emergence of several models derived from the previous ones [33]. However, there are some memory training difficulties, and they can not deal with structured tree inputs [33].

Recursive Neural Networks

The goal of Recursive Neural Networks (RvNNs) is to learn semantic composition [30]. That is why, differently from RNNs, they can process input as a tree structure, such as syntactic structures [30]. For example, given a sentence, RvNN recursively generates the parent tokens in a bottom-up way, as shown in Figure 2.11 [31].

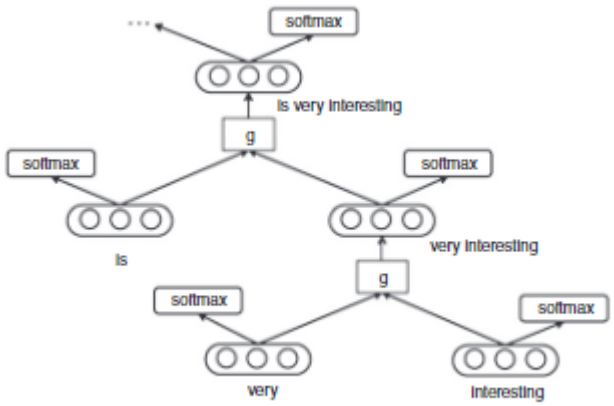


Figure 2.11: Recursive Neural Network Example [31].

Recursive Autoencoders (RAE) and Recursive Neural Tensor Networks (RNTN) are both variants of RvNNs. RAE involves applying the same autoencoder to each node to minimize the loss of sentence reconstruction [33]. RNTN, designed for sentiment analysis, resorts to using a tensor-based composition function that is applied to all nodes and captures the sentiment [33].

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines deep learning and reinforcement learning. The learning process is about an interaction between one agent and the environment, as shown in Figure 2.12.

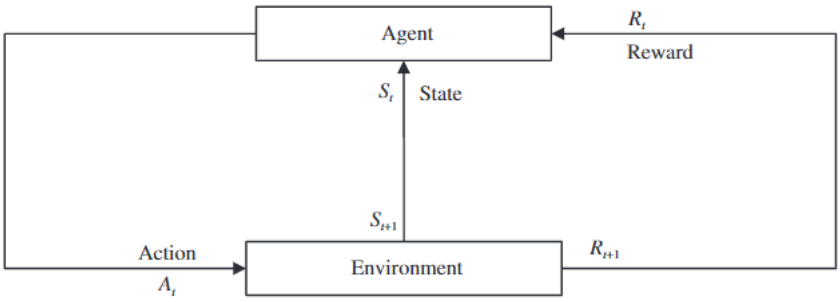


Figure 2.12: Interaction between Agent and Environment [33].

The agent receives a state from the environment and can extract a feature from the state through deep learning [33]. Depending on agent judgment, action will be mapped to the current state. After that, the agent receives a reward for a new observation from the environment [33]. The goal is to maximize the reward [33]. In terms of sentiment analysis, DRL can be used to obtain a structured word and sentence representation together with other deep learning approaches [33]. Furthermore, the reward can represent the correct label prediction probability [33][39].

Hybrid Deep Neural Networks

Hybrid Deep Neural Networks combine some of the previous deep learning approaches. In addition, the approaches can be applied to specific tasks like CNNs as local feature extractors and LSTMs to process long-term dependencies as proposed in [40].

The table down below details CNNs, RNN and RvNN models, their advantages, disadvantages, and what implications the models have for Aspect-Based Sentiment Analysis (ABSA) extracted from [30].

	CNN	RNN	RvNN
Advantages	- Ability to extract meaningful local patterns(n-grams) - Non-linear dynamics - Fast Computation	- Distributed hidden state that can store past computations - Ability to produce a fixed size vector that takes into account the weighted combination of all words and summarizes the sequence - Do not require large dataset - Require fewer parameters	- Simple architecture - Ability to learn tree-like structures - Ability to construct representations for any new word
Disadvantages	- High demand for data	- Chooses the last hidden state to represent the sentence which may lead to incorrect prediction	- Requires parsers/parameters which can be slow and lead to inaccuracies
Implication for ABSA	- Fixed size of hidden layer - Useful if the sentence is supposed to have one opinion/target of fixed length - Not effective for parsing longer sentences	- Failure to capture long-term dependencies - Not useful if the sentence is represented bya key phrase	- Models are still in their infancy - The application and training regime still require further research

Table 2.1: Models Comparsion Summary from [30]

Transformers

To overcome the disadvantages of the previous approaches, some mechanisms have been applied to improve their performance [41].

The attention mechanism, one of the most well-known, prioritizes parts of the input sentence given its weight representation [33]. Its first use in NLP [42] consists of using an encoder and a decoder. The encoder processes the input sentence, and the word representation from the decoder will depend on the weight combination of the input state and not only on the previous state [31]. The higher the weight, the higher the attention to that input segment. It has mostly been applied to RNNs to pick contextual information at every step [33].
Besides that, some studies were made to allow word embeddings to be mapped according to context without one-to-one mapping, leading to a better context-based mutability [41]. As mentioned in [41], there is the possibility of having the same word representing two different things in two sentences. The following examples were given: "Apple Inc performed well this year." and "Apple fruits are exported to various countries.". The word Apple will have the same encoded value in both examples, although it represents different entities. That is why contextual word embeddings can make a significant difference and provide a more reliable

word representation.

Vaswani A, Shazeer N, Parmar N et al [43] present a solution that covers both improvements. It contains a self-attention mechanism without RNNs or convolution and allows for the building of contextualized embeddings, and it is named the Transformer model. Its architecture is shown in Figure 2.13.

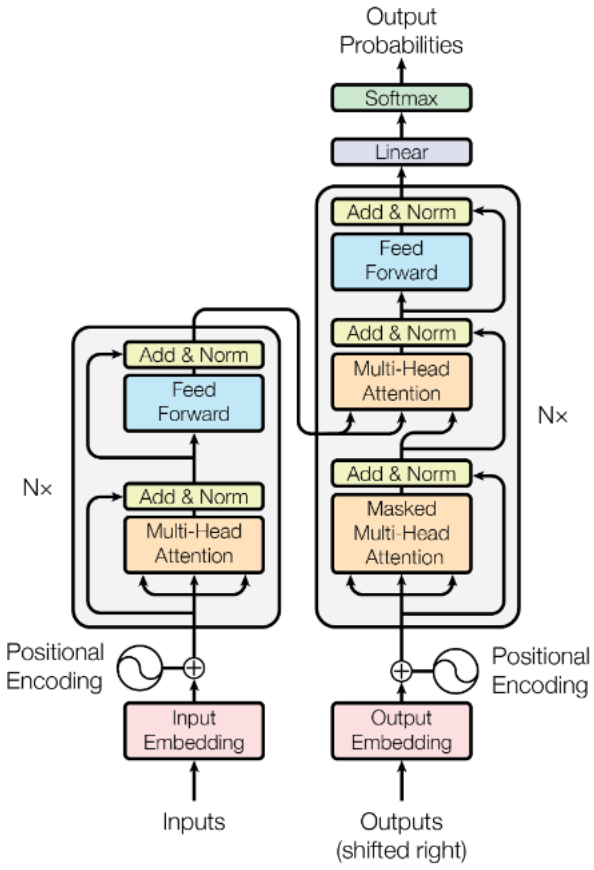


Figure 2.13: Transformer architecture [43].

The Transformer architecture is mainly composed of one encoder and a decoder. Its purpose is to transform one sequence into another where the encoder is responsible for generating contiguous representations for the input sequence, and the decoder will generate the output sequence, one element at a time, depending on previous outputs [43][41].
The fact that there are no recurrent networks to remember sequences reduces computation complexity and allows parallel computation and a better understanding of long-term dependencies [41]. Positional embedding comes up to replace the recurrent network role on long term-dependencies [41]. It relies on having an embedded representation of each word according to its position in the sequence [43].

Scaled dot-product attention, as the name suggests, is an attention mechanism that computes the attention to the inputs. It is computed by giving a score to a word against other words of the input sentence. The following equation describes its computation:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.9}$$

Given a matrix representation of a word in a sequence Q and all vector representations of all words in the sequence K, it is possible to compute the influence of a word in the input sequence. The multiplication of both matrices returns the attention score divided by the dimension $\sqrt{d_k}$ to stabilize the gradient [43][41]. The softmax function ensures that the calculated weights are distributed between 0 and 1. These weights are then applied to the vector Vwhich contains the representation of all words in the input sequence [43][41]. This way is possible to increase the representation values of the words that need more attention and decrease the other ones.

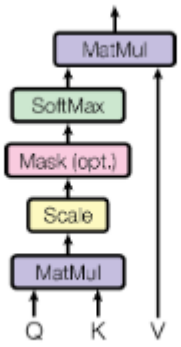


Figure 2.14: Scaled Dot-Product Attention [43].

Multi-Head Attention Mechanism allows for computing multiple scaled dot-product attention in parallel. It computes the attention weights for each word and returns a vector with encoded information on how each word is associated with others. It is a crucial model to obtain context embeddings [41].

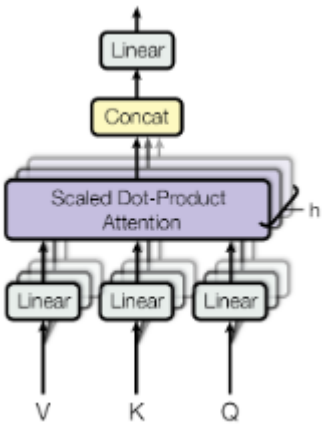


Figure 2.15: Multi-Head Attention [43].

Add & Norm layer normalizes the sum between the input and output from the previous layer. It helps to stabilize the network reducing the training time [44]. The Feed Forward

network layer consists of linear layers with a ReLU activation function that allows having a rich representation of the attention output [44].

On the decoder side, its architecture is very similar to the encoder one. At each step, it will generate output and use it as an additional input in the following steps until it reaches the end token of the sequence[43].
To ensure that when predicting an output for a specific position in the output sequence, the decoder will not look up to future positions, outputs embeddings are shifted right. In addition, masked multi-head attention will mask the tokens that follow the current position with $-\infty$. This way, the scores output from this layer will only consider the positions of the previous output [43].
After being normalized, the following multi-head attention layer will have in the count all the keys and values from the encoder and will apply the score matrix to the values output from the masked multi-head attention [43]. This way, it is possible to map both encoder and decoder inputs which allows the decoder the inputs that should focus on predicting the output for that specific position. The Feed Forward and Add & Norm layers' purposes are the same as the decoder. Furthermore, to turn the output decoder into a word, the Linear layer will project the output vector from the decoder in a vector with the size of the total number of words processed by the model, and the Softmax layer will return a distributed probability over these words [43].

BERT

Back in 2018, Jacob Devlin et al. [45], introduced a new language representation model based on transformers that revolutionized NLP tasks and produced stat-of-the-art results, the Bidirectional Encoder Representations from Transformers (BERT)[6] model. It aims to consider the context of a word in which representation was learned from unlabeled data simultaneously based on each layer's left and right side context.
In order to obtain a language representation model, BERT has two main steps: pre-training and fine-tuning. Pre-training is done by two tasks: Masked Language Model (MLM) and Next Sequence Prediction (NSP). The MLM task consists in randomly masking input tokens which will be predicted then. NSP allows understanding the relationship between two sentences, for example, if sentence B is the sentence that follows sentence A.
The inputs of the BERT model can represent one or more sentences in a token sequence. Each token starts with a special classification token [CLS]. Then, inside the sequence, in order to split the sentences, a special token [SEP] is used. Finally, each token is associated with a learning embedding indicating which sentence it belongs to, as shown in Figure 2.16.

[6] https://github.com/google-research/bert

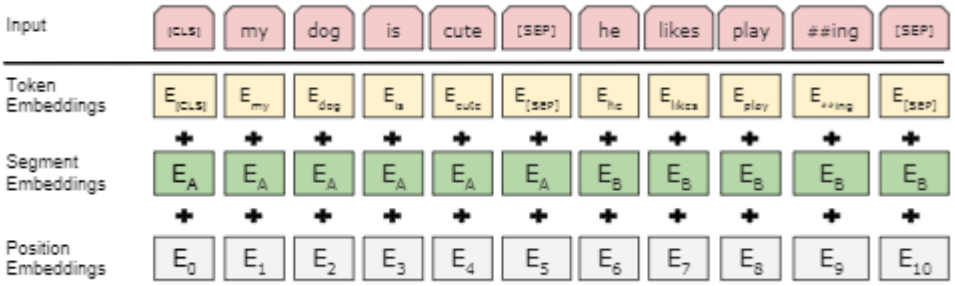


Figure 2.16: BERT Input Representation [45]

The fine-tuning step is initialized with BERT pre-trained data, and all parameters are fine-tuned depending on the task they want to perform. Each task has different fine-tune models, but all of them are initialized with the same pre-trained parameters, as it is shown in Figure 2.17.

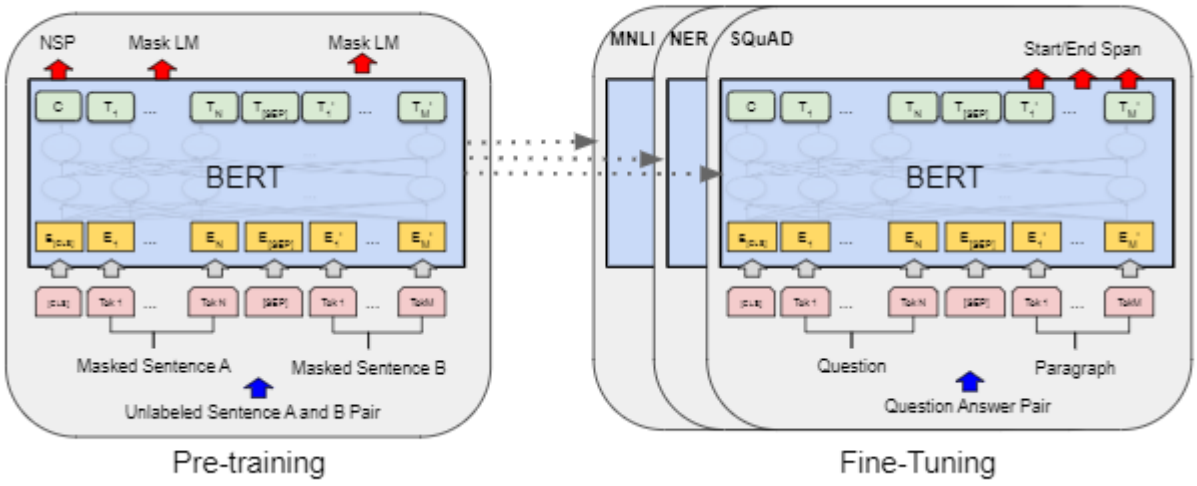


Figure 2.17: BERT Pre-Training and Fine-Tuning Overview [45]

Shivaji Alaparthi and Manit Mishra [46] compared the BERT model to other classic approaches for the sentiment analysis of the corpora based on *SentiWordNet*[7], and the results are presented in the following table:

[7] `https://github.com/aesuli/SentiWordNet`

Classification Technique	Model	Accuracy	Precision	Recall	F1 Score
Unsupervised Lexicon Based Model	Sent WordNet	0.6308	0.6747	0.6308	0.6064
Supervised Machine Learning Model	Logistic Regression	0.8941	0.8975	0.8941	0.8941
Supervised Deep Learning Model	LSTM	0.8675	0.8680	0.8675	0.8675
Advanced Supervised Deep Learning Model	BERT	0.9231	0.9235	0.9231	0.9231

Table 2.2: Classification Results on the Different Models [46].

Although BERT clearly overcomes other models, as it is mentioned, one of the points against BERT is its high demand for computation power.

2.4 Aspect Based Sentiment Analysis Research

Aspect Based Sentiment Analysis (ABSA) applies sentiment analysis to specific entities and aspects. It has three main tasks: Aspect category classification, Opinion target expression, and Sentiment polarity classification.

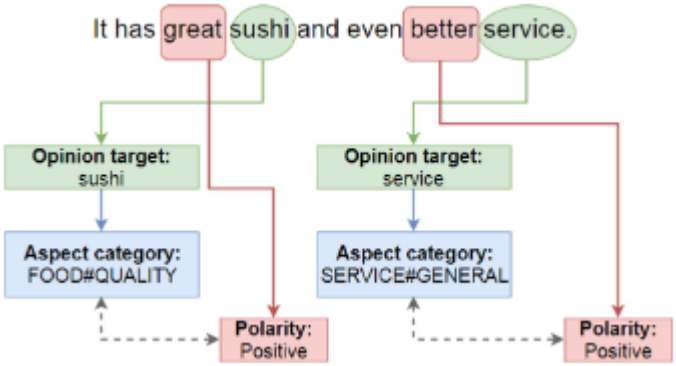


Figure 2.18: ABSA tasks example [30]

ABSA was first introduced in 2014 by Pontiki et al. [47], where an ABSA dataset with annotated reviews from restaurants and laptops domain was released. The main purpose was to identify the aspect and its sentiment polarities. Furthermore, SemEval 2015 and SemEval2016 datasets defined the aspect as a combination of an entity type and an attribute [48]. Figure 2.18 shows how an entity and an attribute define an aspect.

Figure 2.19 presents the models with the highest mean accuracy using the SemEval 2014 laptop and restaurant datasets.

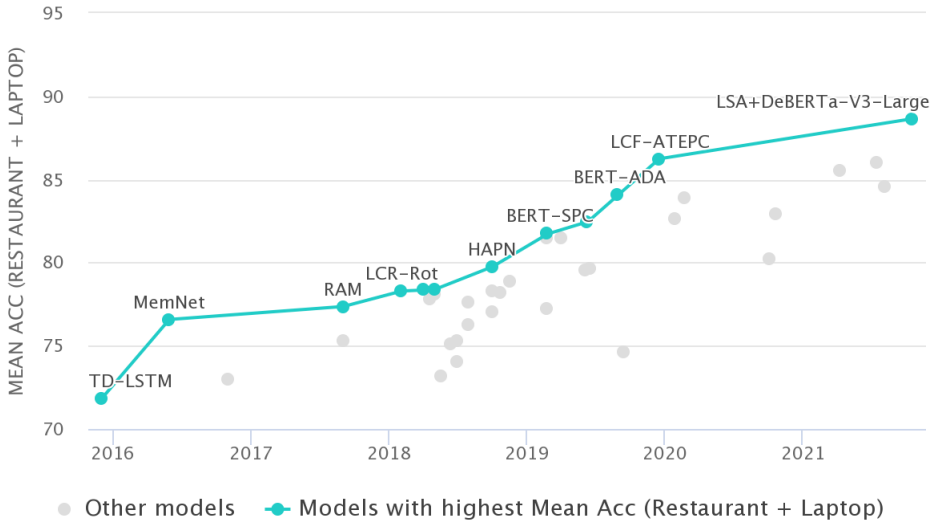


Figure 2.19: Aspect-Based Sentiment Analysis on SemEval 2014 Task 4 Sub Task 2
`https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval`
Accessed on February 19, 2022

The following tables show the models' results in the SemEval tasks involving Aspect-Based Sentiment Analysis.

Rank	Model	Mean Acc	Restaurant Acc	Laptop Acc	Paper	Year
1	LSA+DeBERTa-V3-Large	88.64	91.07	86.21	[49]	2021
2	LCF-ATEPC	86.24	90.18	82.29	[50]	2019
3	ABSA-DeBERTa	86.11	89.46	82.76	[51]	2021
4	RoBERTa+MLP	85.58	87.37	83.78	[52]	2021
5	KaGRMN-DSG	84.61	87.35	81.87	[53]	2021

Table 2.3: Top 5 Aspect-Based Sentiment Analysis on SemEval 2014 Task 4 Sub Task 2.
`https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval`
Accessed on February 19, 2022.

Rank	Model	Restaurant Acc	Paper	Year
1	HAABSA	80.6	[54]	2019

Table 2.4: Aspect-Based Sentiment Analysis on SemEval 2015 Task 12.
`https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval-1`
Accessed on February 19, 2022.

Rank	Model	Restaurant Acc	Paper	Year
1	BERT-IL Finetuned	88.70	[55]	2020
2	HAABSA	88.0	[54]	2019
3	HAABSA++	87.0	[56]	2020

Table 2.5: Aspect-Based Sentiment Analysis on SemEval-2016 Task 5 Subtask 1. `https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval-2` Accessed on February 19, 2022.

CHAPTER 3

Aspects Extraction and Classification

Taking into account not only the results of section 2.4 but also the flexibility provided, the model chosen for the extraction and classification of the aspects was LCF-ATEPC.

3.1 LCF-ATEPC

Yang H, Zeng B, Yang J et al. [50] developed a novel multilingual and multi-task model using SemEval-2014 task4 datasets that produced state-of-the-art results, LCF-ATEPC. Through multi-head self-attention Multi-Head Self-Attention (MHSA), within the BERT and the Local Context Focus (LCF) mechanisms, it performs the Aspect Term Extraction (ATE) and Aspect Polarity Classification (APC) subtasks of ABSA. Self-attention and local context focus techniques were applied to improve the collaborative training of ATE and APC. In addition, domain-adapted BERT was applied to improve word embeddings and poor performance problems on small datasets, enhancing both the ATE and APC, especially the F1 score of ATE task [50].

In terms of the ATE task, IOB labels were used to identify the aspects present in a sentence to set up the data to be classified. Regarding the APC task, one set of words contains the sequence tokens and another with the words that compose the aspect is prepared [50].

3.1.1 Architecture

LCF-ATEPC architecture is based on two distinct context feature generators, Local Context Feature Generator (LCFG) and Global Context Feature Generator (GCFG). In addition, it contains an aspect extractor and a feature interactive learning layer that combines both local and global context features and holds the polarity extractor, as shown in the figure below.

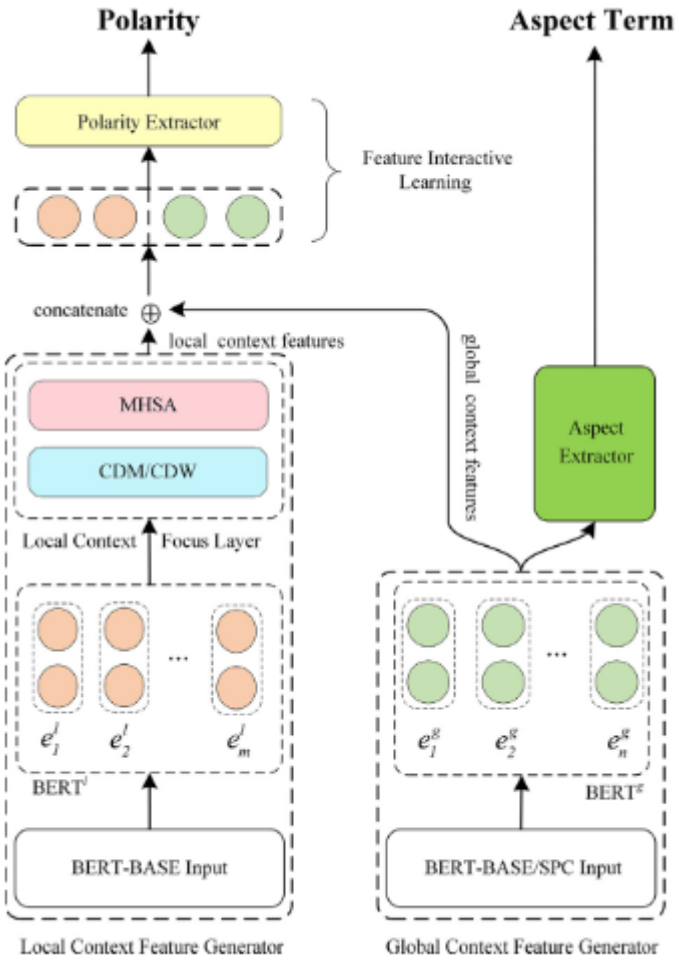


Figure 3.1: LCF-ATEPC Architecture [50]

Both local and global context feature generators contain a BERT-shared layer,$BERT^l$, and $BERT^g$, respectively. The output for both layers is the preliminary extracted features of the inputs, X^l and X^g, the corresponding tokenized inputs of LCFG and GCFG. It is possible to represent the output of both BERT layers as follows:

$$O^l_{BERT} = BERT^l\left(X^l\right) \tag{3.1}$$

$$O^g_{BERT} = BERT^g\left(X^g\right) \tag{3.2}$$

The LCFG layer, besides the BERT layer that will extract local context features, it contains a local context focus layer that relies on Semantic-Relative Distance (SRD) to identify how far a token is from a targeted aspect. SRD is applied to each considered token by applying the following equation:

$$SRD_i = |i - p_a| - \lfloor \frac{m}{2} \rfloor \tag{3.3}$$

where $i(1 < i < n)$ stands for the token position, p_a is the central position of each aspect term, and m the aspect length. It will calculate the distance of the tokens around the aspect term. The CDM/CDW layer is where the context features can be dynamically masked or weighted. Context-Feature Dynamic Mask (CDM) consists of masking the non-local context's features learned from $BERT^l$ to discard non-local context words' contribution to the learning process [50]. The masking process is done by creating masking vectors that will have the value' 1' if the SRD from the input token belongs to the threshold α and the value '0' if the SRD is bigger than the threshold, as defined in the following equation:

$$V_i = \begin{cases} E & SRD_i \leq \alpha \\ O & SRD_i > \alpha \end{cases} \tag{3.4}$$

where E represents the one's vector and O the zeros vector. Having the masking vector for each input token, [50] defined a masking matrix M with each masking vector V_i^m of the n input sequence tokens:

$$M = [V_1^m, V_2^m, ..., V_n^m] \tag{3.5}$$

Finally, the matrix M is applied to the output of the previous BERT layer O_{BERT^l}:

$$O_{CDM}^l = O_{BERT^l} \cdot M \tag{3.6}$$

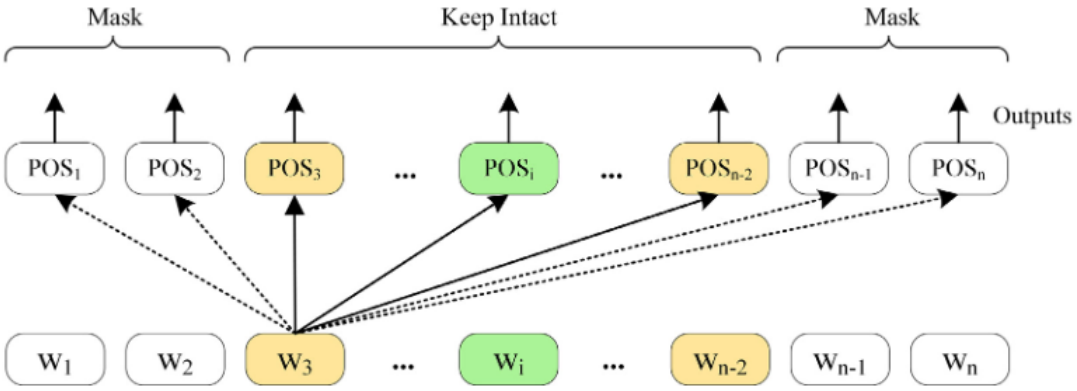


Figure 3.2: CDM Mechanism Representation [50].

Context-Feature Dynamic Weighting (CDW) consists of completely discarding the non-local context features [50]. Instead of masking the non-local context features, thus will be weighted decay depending on their SRD. Consequently, instead of masked vectors of the input tokens, weighted vectors of each input token V_i^m are defined by the following equation:

$$V_i = \begin{cases} E & SRD_i \leq \alpha \\ \frac{n-(SRD_i-\alpha)}{n} \cdot E & SRD_i > \alpha \end{cases} \tag{3.7}$$

The weight matrix W, will contain the weighted vectors for the n input sequence tokens:

$$W = [V_1^w, V_2^w, ..., V_n^w] \tag{3.8}$$

and similarly to the CDM mechanism, the W matrix will be applied to the previous layer output O_{BERT^l} with the following dot-product operation:

$$O^l_{CDW} = O_{BERT^l} \cdot W \tag{3.9}$$

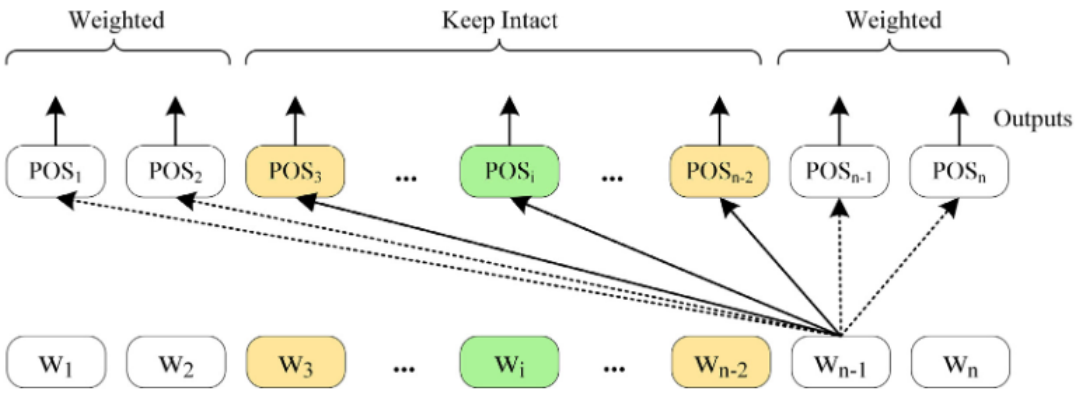


Figure 3.3: CDW Mechanism Representation [50].

MHSA layer, in its turn, as explained in section 2.3.1, is based on scaled-dot attention and, in LCF-ATEPC, performs an important role in terms of extracting deep semantic features of the context represented in the self-attention score [50]. Considering I as the input features, the parameters to calculate scaled-dot attention are defined as follows:

$$f(I) = \begin{cases} Q = I \cdot W^q \\ K = I \cdot W^k \\ V = I \cdot W^v \end{cases} \tag{3.10}$$

The input features are obtained from the previous CDM/CDW layers for the first layer of the scaled-dot attention. Then, each weight matrix is randomly initialized and updated during training [50]. MHSA allows extracting the local context features learned in CDM/CDW layers in three alternative ways: LCF-CDM, LCF-CDW, and LCF-Fusion. It is then applied to the output of each mechanism and can be represented as:

$$O^l = MHSA\left(O^l_{CDM}\right) \tag{3.11}$$

$$O^l = MHSA\left(O^l_{CDW}\right) \tag{3.12}$$

Equations 3.11 and 3.12 represent the local focus mechanism based on CDM and CDW, respectively. Both outputs are denoted as O^l because as mentioned in [50], CDM and CDW layers are independent, alternative, and fusible. The local context feature fusion stands for the concatenation of both learning features from CDM and CDW:

$$O^l_{fusion} = \left[O^l_{CDM}; O^l_{CDW}\right] \tag{3.13}$$

followed by a linear transformation with W^f, O^f as weight matrix, and b^f as the bias vector so that MHSA can be applied to O^l_{fusion}:

$$O^l_{fusion} = W^f \cdot O^f + b^f \tag{3.14}$$

$$O^l = MHSA\left(O^l_{fusion}\right) \tag{3.15}$$

Furthermore, the Feature Interactive Learning (FIL) will concatenate local and global context features. This way, it can preserve some information loss that might happen when discarding local context features. To learn the concatenated features, a multi-head self-attention process is applied after being transformed:

$$O^{lg} = \left[O^l; O^g\right] \tag{3.16}$$

$$O^{lg}_{dense} = W^f \cdot O^f + b^f \tag{3.17}$$

$$O^{lg}_{FIL} = MHSA\left(O^{lg}_{dense}\right) \tag{3.18}$$

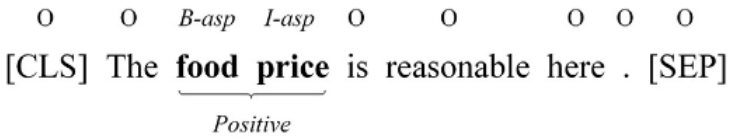
The layer that allows to get the aspect polarity classification, the polarity extractor, which starts by extracting the first output from the previous layer (O^{lg}_{FIL}) through head-pooling:

$$X^{lg}_{POOL} = POOL\left(O^{lg}_{FIL}\right) \tag{3.19}$$

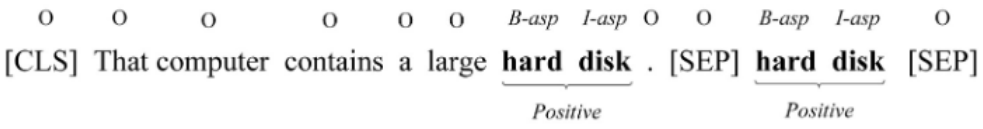
and then applies a softmax function to predict the polarity inside the C available sentiment categories:

$$Y_{Polarity} = \frac{exp\left(X^{lg}_{POOL}\right)}{\sum_{k=1}^{C} exp\left(X^{lg}_{POOL}\right)} \tag{3.20}$$

On the other side, the GCFG allows to have BERT-BASE or BERT-SPC as BERT input layer. The difference between both models is that BERT-SPC not only reforms the input sequence tokens but the aspect as well (Figure 3.4). As mentioned in [50], it improves the aspect polarity classification sub-task of LCF-ATEPC model.



(a) BERT-BASE input format example [50].



(b) BERT-SPC input format example [50].

Figure 3.4: BERT-BASE and BERT-SPC input format examples [50].

The global context features learned from the BERT layer are the aspect extractor inputs that aim to identify the aspect term. As mentioned before, aspects are labeled with IOB2 labels: B_{asp}, identifies the beginning of the aspect, I_{asp}, whether the token is inside the aspect or not, and O for the tokens outside the aspect. In addition, the aspect term extractor is responsible for classifying to which one of the N categories the token T_i belongs using the following softmax operation:

$$Y_{Term} = \frac{exp\,(T_i)}{\sum_{k=1}^{N} exp\,(T_i)} \tag{3.21}$$

In terms of loss, the model uses the cross-entropy loss within L_2 regularization for both ATE and APC tasks[50].
L_2 regularization shrinks all the weights to small values to prevent the model from overfitting [57].
Considering λ as the L_2 regularization parameter, Θ_1 and Θ_2 as the parameter set of LCF-ATEPC, the loss function for the aspect polarity classification task is the following one:

$$\mathcal{L}_{apc} = -\sum_{1}^{C} \hat{y}_i \log y_i + \lambda_1 \sum_{\theta_1 \in \Theta_1} \theta_1^2 \tag{3.22}$$

where C is the number of categories.
The loss function for the aspect extraction task is:

$$\mathcal{L}_{ate} = -\sum_{1}^{N}\sum_{1}^{k} \hat{t}_i \log t_i + \lambda_2 \sum_{\theta_2 \in \Theta_2} \theta_2^2 \tag{3.23}$$

where N is the number of token classes and k is the sum of the tokens in each sequence. This way, the loss function for the model is defined as follows:

$$\mathcal{L}_{atepc} = \mathcal{L}_{apc} + \mathcal{L}_{ate} \tag{3.24}$$

3.1.2 Datasets Results Overview

The optimal performance obtained by the model was reached when using domain-adapted BERT-BASE [50]. It was applied to Laptop and Restaurant datasets, and the results are represented in the following tables:

- Laptop Dataset:

Model	$F1_{ate}$	ACC_{apc}	$F1_{apc}$
LFC-ATEPC-CDM	85.29	83.07	79.84
LFC-ATEPC-CDW	84.24	81.76	78.06
LFC-ATEPC-Fusion	85.06	81.76	78.65

Table 3.1: Optimal Performance on Laptop Dataset [50].

- Restaurant Dataset:

Model	$F1_{ate}$	ACC_{apc}	$F1_{apc}$
LFC-ATEPC-CDM	89.78	90.18	85.88
LFC-ATEPC-CDW	89.99	88.65	83.7
LFC-ATEPC-Fusion	89.94	89.45	84.76

Table 3.2: Optimal Performance on Restaurant Dataset [50].

CHAPTER 4

System Overview

This chapter is related to the system developed to create a web app. The idea was to create an app that allows to add healthcare and well-being apps and display the respective top positive and negative aspects of the extracted reviews.

4.1 Methodology

Nowadays, there is a massive amount of data all over the internet. It is rare to find a search that does not present a website or platform with the desired information. Regarding product reviews, there are numerous platforms where users can expose their opinion. Therefore, extracting data in the intended domain is a must in sentiment analysis. The increased need for data led to the development of several data extraction methods through crawlers, scrapers, and APIs, allowing efficient review extraction.
To obtain reviews from official sources directly associated with the product, smartphone application stores are sources that, in addition to being the sites where most users gain access to the application, also allow for an up-to-date perception of the users' opinion. Considering this, finding a data source that would allow obtaining information about the applications and their respective reviews from the application stores, both the Google Play Store and the Apple Store, was the goal. The search resulted in two open-source scrapers developed in Python, one for Google Play Store[1] and the other for Apple Store[2]. These scrapers provide APIs to obtain the app's name, id, language, reviews, and in the case of Google Play, the permissions of a specific app and reviews classifications. To perform the aspects extraction and classification task, the aspects are extracted through the *ATEPCCheckpointManager* module from the pyabsa[3] library. It provides a method to inference the reviews aspects and polarity.

A system was developed to extract and classify aspects for the reviews of any app through a web application. The web app allows users to add an app to the system from the Google

[1] `https://pypi.org/project/google-play-scraper/`
[2] `https://pypi.org/project/app-store-scraper/`
[3] `https://github.com/yangheng95/PyABSA`

Play Store or Apple Store.
When users introduce a new app into the system, the last 100 reviews are collected, and their aspects are extracted and classified. Each time users access the respective app, the information regarding the aspects is updated, checking if in the last 100 reviews there is any with a more recent date concerning the last saved review. If so, the information regarding the aspects is updated.

Next, the system architecture and application sequence diagrams will be described.

4.2 Architecture

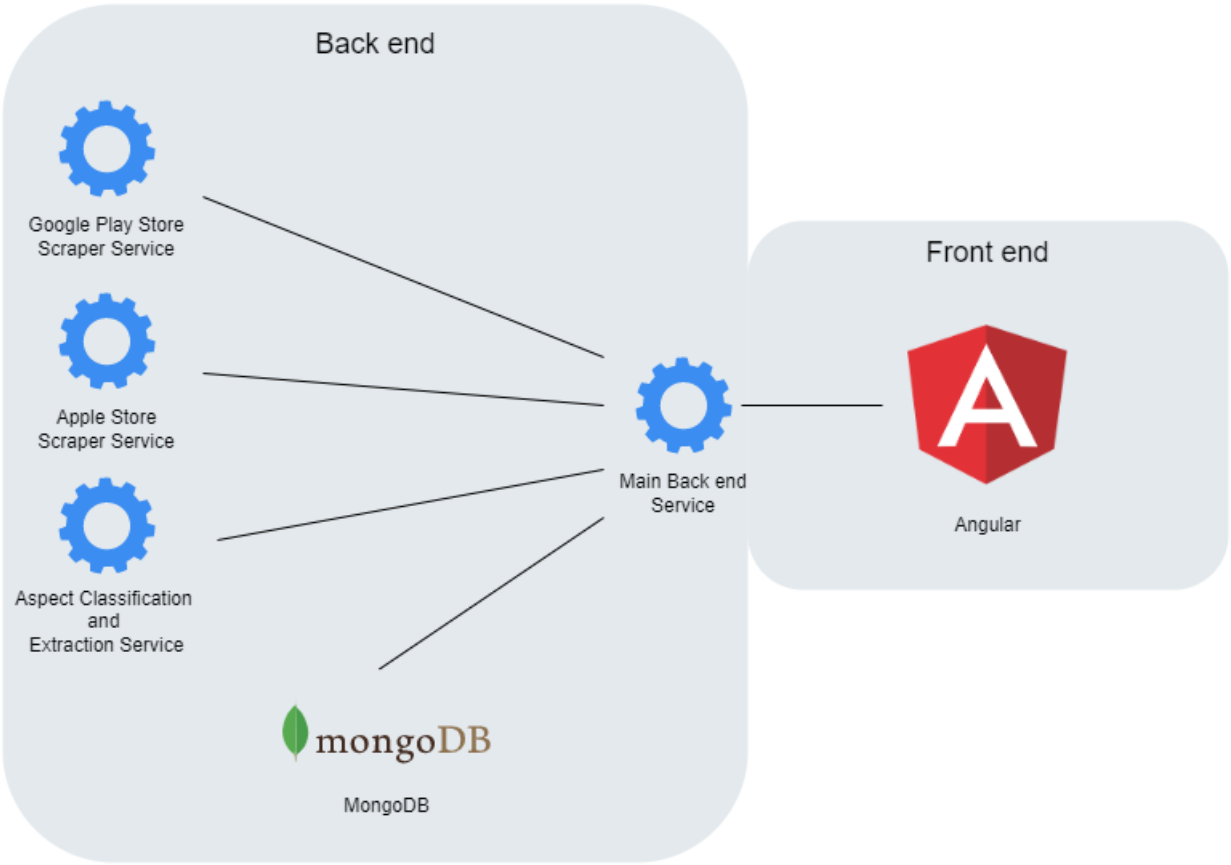


Figure 4.1: System Architecture.

The back end is based on three micro-services, the main back end service, and a mongo database. Micro-services and the main back end service are flask-based apps, a web application framework written in Python that provides much flexibility. Alongside the flask app API, each service has an additional layer to process data.
Both Google Play Store and App Store services are responsible for fetching reviews and app info.
In turn, the Aspect Extraction and Classification Service demand the aspects extraction and classification of the reviews.
The Main Back end Service handles the interconnection between the micro-services and the

database and is the one responsible for handling the front end requests.
The Front end is the user interface responsible for creating a point of communication between the user and the system.

4.2.1 Review collection services

Google Play Store Service

Google Play Store Scraper provides a method that, through the app id, the desired language, and country, it is possible to get the general information about the app. The id can be found in the URL of the app page. For example, MyFitnessPal: Calorie Counter app, the URL is `https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=en`, app id is *com.myfitnessppal.android*. Lang and country parameters are described depending on the ISO 3166[4] and ISO 639-1[5] standards, respectively. The result provides app information such as score, description, developers info, price, screenshots, and other properties available on the documentation page. To filter the extracted reviews, parameters such as the number of reviews to extract, the sort type, score, language, and country, giving the app id can be set.

```
result, continuation_token = reviews(
    'app_id',
    lang = 'language', # defaults to 'en'
    country ='country', # defaults to 'us'
    sort =Sort.NEWEST, # defaults to Sort.NEWEST
    count = number_of_reviews_to_extract, # defaults to 100
    filter_score_with = filter # defaults to None (means all score)
)
```

The *continuation_token* contains the number of items and the arguments used in the current result. If the method is called again with the same *continuation_token*, it will retrieve the data from the previous result or later ones. Its default value is *None* which is represented by '_'.

Google Play Store Service uses the described library to retrieve the info about a google play store app, it receives the needed arguments from the main back end service, and then the service layer fetches the data. Beyond app info and reviews, it has a method that retrieves all reviews from a given date.
Although the scraper provides much information such as score, description, developers info, and more, the current information being extracted is the title, icon, id, and the reviews.

The setup for this project is to find 100 reviews at a time, in English, from Great Britain and sorted by date. The scrape parameters are described in the following code snippet:

[4] `https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes`
[5] `https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes`

```
app =  app('app_id', lang="en", country="gb")

reviews, _ = reviews(
    'app_id',
    lang = 'en', # defaults to 'en'
    country ='gb', # defaults to 'us'
    sort = Sort.NEWEST | Sort.MOST_RELEVANT, # defaults to Sort.NEWEST
    count = 100, # defaults to 100
    filter_score_with = None # defaults to None (means all score)
)
```

The *app* schema, will be stored as follows:

```
{
    "title": str,
    "id": str,
    "icon": str
    "reviews" :  {
        "userName": str,
        "userImage": str,
        "content": str
        "score": int,
        "thumbsUpCount": int,
        "reviewCreatedVersion": str,
        "at": str,
        "replyContent": str,
        "repliedAt": str,
        "reviewId": str
    }
}
```

The extracted data is sent to main back end service through the flask Application Programming Interface (API) and one will store it in the database.

Apple Store Service

Apple Store Scraper, similarly to Google Play Scraper, allows extracting app information and respective reviews. However, beyond the app id, it requires the app name to search for the pretended app.

```
app = AppStore(country='country', app_name='app_name', app_id=app_id)
```

Both country and name are required arguments to search for an app. The country parameter is a two-letter country code of ISO 3166-1 alpha-2 standard[6]. The id is an optional parameter

[6] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

that, when not in the input parameters, the scraper will execute a method *search_id()* based on the app name [58]. Otherwise, it is present on the app URL page. For example, give an URL: `https://apps.apple.com/us/app/myfitnesspal-calorie-counter/id341232718`; the id is *341232718*.

To extract the app reviews, the scraper provides the method *review()*:

```
app.review(how_many=100)
```

The *how_many* parameter sets the number of reviews to be fetched. If not provided, it will fetch all reviews. Accordingly to [58], the maximum number of reviews fetched per request is twenty. Beyond the *how_many* parameter, there are two additional optional parameters, *after* and *sleep*: *after* is a *DateTime* object to filter older reviews from the given date; *sleep* is an int that allows setting a several seconds to sleep between each call.

Regarding the arguments received from main back end service, the app name and app id, the app info retrieved from scraper is country, name, id, URL, and reviews count.
Like Google Play Store service, the setup stands for find 100 reviews from Great Britain:

```
app = AppStore(country='gb', app_name='app_name', app_id=app_id)

app.review(how_many=100)
```

To get the icon URL, an additional method was created. Using urllib.request module from urllib[7] python package, given the app id, the URL `https://itunes.apple.com/lookup?id={app_id}` is read, and the icon URL is extracted from the "artworkUrl60" property.
In the end, the schema retrieved from the service is as follows:

```
{
    "name": str,
    "id": str,
    "icon": str
    "reviews":{
            "date": str,
            "isEdited": bool,
            "rating": int,
            "review": str,
            "title": str,
            "userName": str
    }
}
```

[7] `https://docs.python.org/3/library/urllib.html`

4.2.2 Aspect Extraction and Classification Service

The Aspect Extraction and Classification Service is a service that, as the name implies, extracts and classifies the aspects of reviews extracted from applications added by users.
It receives the app's id and respective reviews for which the aspects will be extracted and classified from the Main Back End Service and returns the aspects info.

```
aspect_extractor = ATEPCCheckpointManager.get_aspect_extractor(
                                                checkpoint=checkpoint
                                                auto_device=False)
                                                # False means
                                                #load model on CPU


atepc_result = aspect_extractor.extract_aspect(
                        inference_source=reviews,
                        save_result=True,
                        print_result=False,
                        pred_sentiment=True)
```

The checkpoint parameter dictates the settings and performance results of the performed training, so that the inference performance will depend on the checkpoint used.
The inference result is saved in a JavaScript Object Notation (JSON) file. The text, the IOB sequence, the review tokens, the aspects, the position of the aspects, and their respective polarity are extracted for each review.
From the JSON file, the information extracted relative to each review is the following one:

```
{
    "sentence": str,
    "aspects": str [],
    "sentiment": str []
}
```

From there on, the goal was to obtain the most relevant aspects and understand how often they are classified as positive or negative.
For each review, aspects are extracted. Therefore, stop words and punctuation are removed from the aspects token list. Then, and the nltk frequency distribution class[8] is applied to record the frequency of each aspect.
The next step was sorting each aspect by the number of occurrences through the

[8] https://tedboy.github.io/nlps/generated/generated/nltk.FreqDist.html

most_common()[9] method applied to the previous frequency distribution.
To obtain the number of times it was classified with positive or negative polarity, the first step was to group, for each review, the aspect and respective classification. Then, the frequency distribution class was applied after retrieving all the data. This way, it is possible to know how many times an aspect was classified as positive or negative. Finally, a split was done between aspects with positive and negative classification, and the *most_common()* method was once applied to sort the aspects by the number of classifications.
The data retrieved from this service is stored in the database, has the following schema:

```
{
    "app_id" : str,
    "top_positive": [
        {
        "$aspect": int
        }
    ],
    "top_negative": [
        {
        "$aspect": int
        }
    ]
}
```

4.2.3 MongoDB

MongoDB is a document-oriented database. It was chosen due to its flexibility, and since it stores data in JSON format, it perfectly matches python.
MongoDB Atlas cloud was used to host the database in the cloud. The database has two collections: apps and aspects.
The app collection stores the data returned from Google Play Store Scraper Service and Apple Store Scraper Service described in 4.2.1 and 4.2.1, respectively.
The aspects collection stores the schema retrieved from Aspect Extraction and Classification Service mentioned 4.2.2.

4.2.4 Main Back end Service

Main Back end Service is the one that controls the system data flows. Generically, the flow starts by receiving a request from the front end. Then, to comply with the request, it communicates with the micro-services or with the database to get the data and retrieve it to the front end.

[9] `https://tedboy.github.io/nlps/generated/generated/nltk.FreqDist.most_common.html#nltk.FreqDist.most_common`

4.2.5 Front end

The Front end is based on an Angular[10] platform built on TypeScript[11]. It includes a component-based framework and many integrated libraries that provide many features that ease its learning process and use. Beyond that, it contains structured and complete documentation that helps implement most of the standard components in a web application. Nebular[12] is a customizable Angular UI library that provides four visual themes. The theme chosen was the nebular default due to previous familiarization and complete documentation, simplifying the implementation of the available UI components.

4.3 Sequence Diagrams

The following sequence diagrams represent the data flows and the role of each service in the system when exploring the web app.

4.3.1 List Apps on Homepage

To present the stored apps on the system, a simple query to find all the apps into the app collection is performed by the main back end service and returned to the front end.

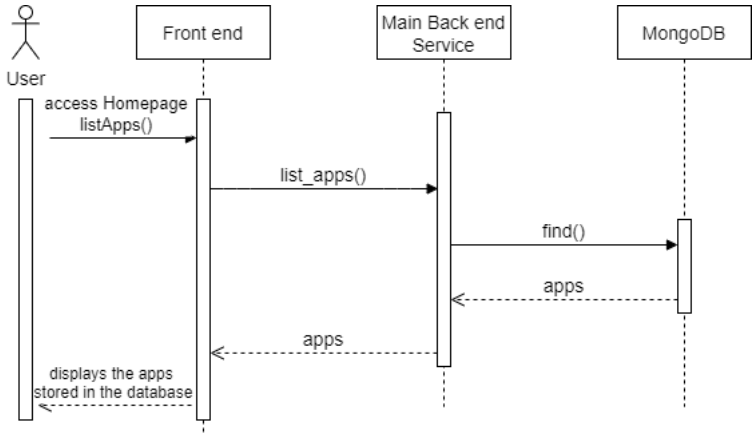


Figure 4.2: List Apps Sequence Diagram.

4.3.2 Save Apps

The following two diagrams represent the sequence diagram to store an app and aspects data on the database. When a user submits the app form on the front end page, the main back end service first checks if the app is already in the database. If so, the user returns to the homepage. Otherwise, the main back end service sends the app parameters to the apple store or google play store to fetch the app data. Next, the reviews retrieved from one of the review collection services are sent to the aspect extraction classification service to get the most relevant positive and negative aspects. When finished, app and aspects data are saved in the database, and the user is redirected to the homepage.

[10] `https://angular.io/`
[11] `https://www.typescriptlang.org/`
[12] `https://akveo.github.io/nebular/`

Save App Store App

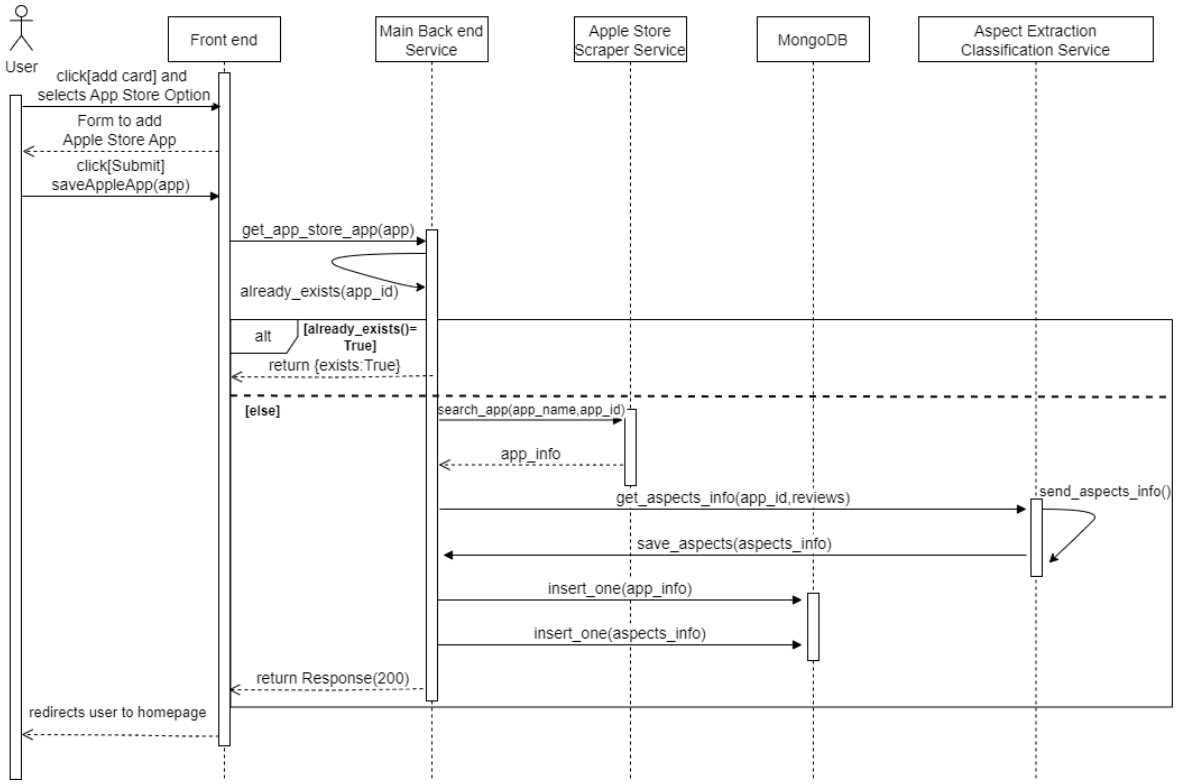


Figure 4.3: Save App Store App System Sequence Diagram.

Save Google Play Store App

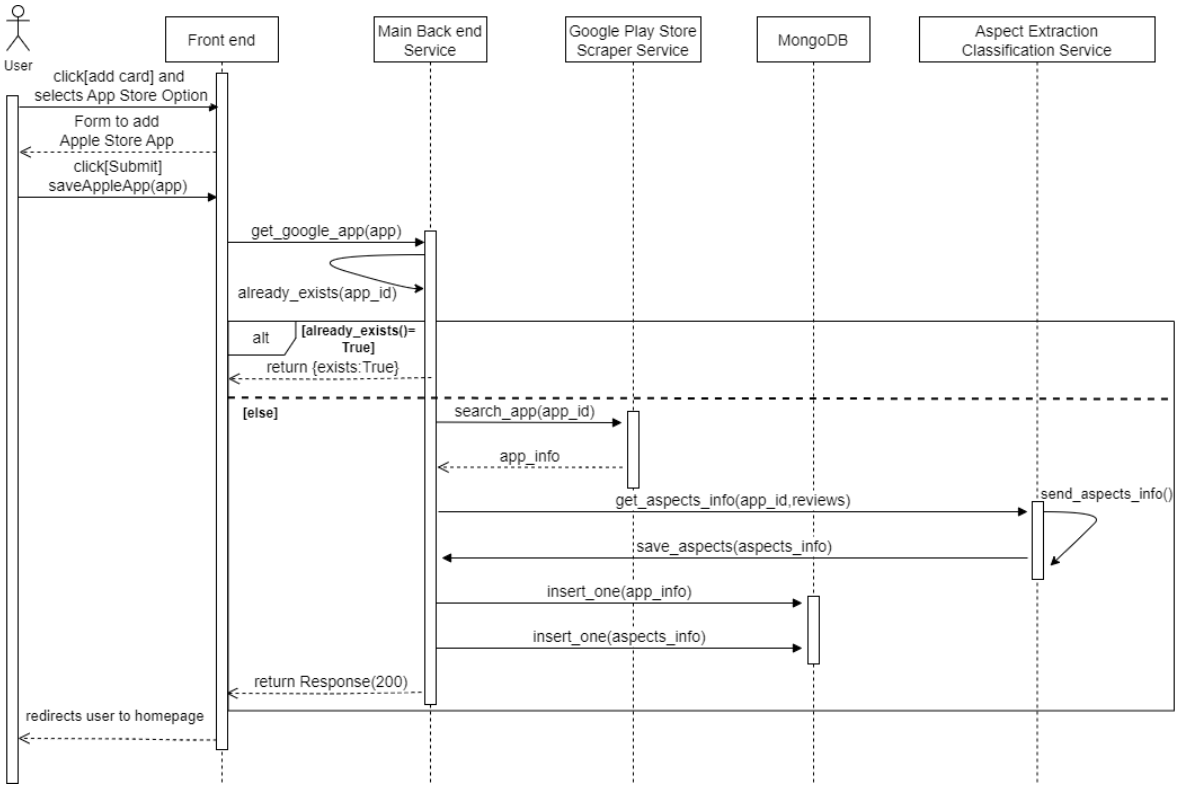


Figure 4.4: Save Google Play Store App Sequence Diagram.

4.3.3 Update Apps Reviews and Aspects

The app reviews and aspects data update is performed when the user accesses a specific app page. The first request from the front end consists of listing the top 10 positive and negative of the current aspects stored in the database. Then, another request is made to update that list.

First, the date of the last stored review is queried to the database. Then, the main back end service redirects it to the app collection review service to fetch new reviews since the given date. If no reviews are found, the user is notified.

Otherwise, the main back end service will send new reviews to the aspect extraction and classification to retrieve their aspects info. The current aspects collection needs to be updated when receiving the aspects info. Therefore, the current aspect collection is compared with new aspects info, and the number of occurrences of the aspects that match on both lists is updated. Finally, the new aspects are added to the current aspects collection.

Update App Store App Reviews and Aspects

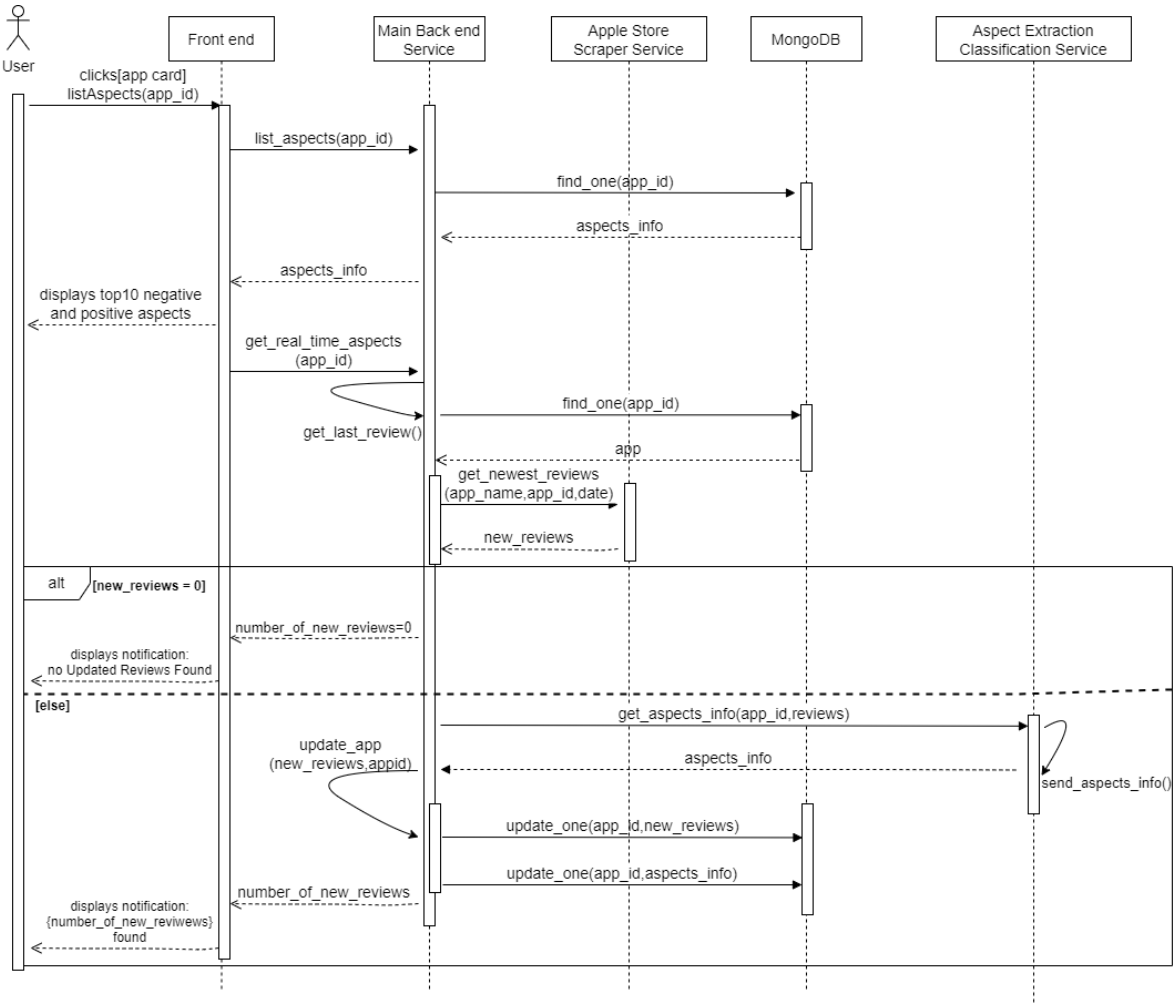


Figure 4.5: Update App Store App Reviews and Aspects Sequence Diagram.

Update Google Play Store App Reviews and Aspects

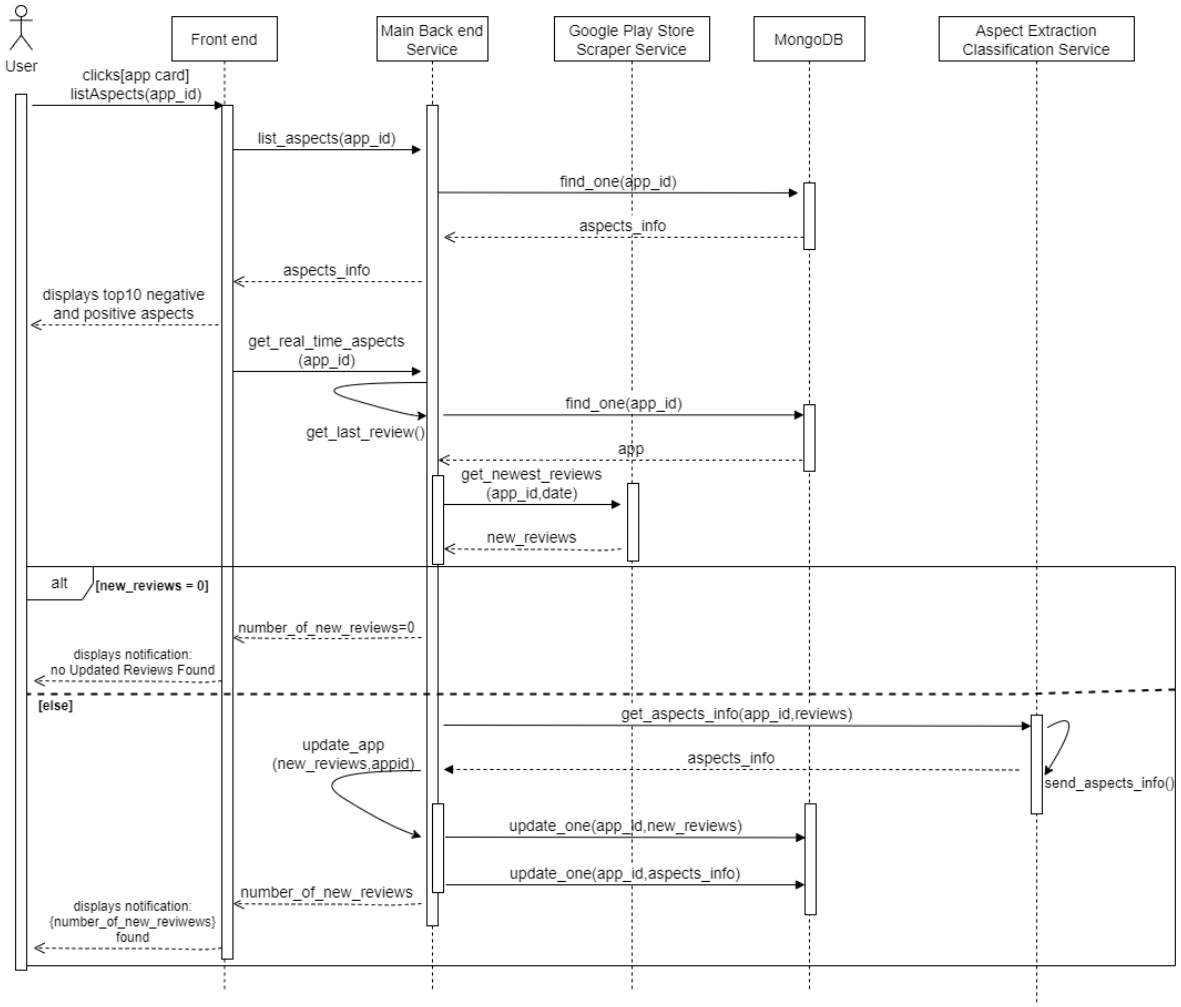


Figure 4.6: Update Google Play Store App Reviews and Aspects Sequence Diagram.

CHAPTER 5

Results

In this chapter, the settings and individual results will be described regarding the training performed with the model described in chapter 3. Next, some views of the web app that represents the system described in chapter 4 will be presented.

5.1 Aspect Extraction and Classification Task

The goal was to train the model with the three different local focus context mechanisms described in chapter 3. Although in [50], the default SRD threshold for all experiments was 5, on the GitHub repository documents page[1], it is mentioned that the value 3 is the best option and it was the one used in all the training sets.

All training sessions carried the following parameters:

```
atepc_config_english = ATEPCConfigManager.get_atepc_config_english()
atepc_config_english.num_epoch = 3
atepc_config_english.evaluate_begin = 1
atepc_config_english.model = ATEPCModelList.LCF_ATEPC
atepc_config_english.use_bert_spc = True
atepc_config_english.lcf = ``cdm'' | ``cdw'' | ``fusion''
```

The training was done in Google Colaboratory[2] platform that allows to execute python in the browser. The code to execute the training was the following one:

```
aspect_extractor = ATEPCTrainer(config=atepc_config_english,
                                dataset=dataset,
                                checkpoint_save_mode=1,
                                auto_device=True
                                )
```

[1] https://github.com/yangheng95/PyABSA/tree/release/demos/documents
[2] https://colab.research.google.com/?utm_source=scs-index

Laptop Training Dataset Results

```
dataset = ABSADatasetList.Laptop14
```

Local Context Focus Mechanism	ACC_{apc}	$F1_{apc}$	$F1_{ate}$
CDM	79.62	74.63	84.18
CDW	79.62	75.28	85.13
Fusion	79.15	75.02	86.08

Table 5.1: Results for Laptop Dataset Training.

Restaurant Dataset Training Results

```
dataset = ABSADatasetList.Restaurant14
```

Local Context Focus Mechanism	ACC_{apc}	$F1_{apc}$	$F1_{ate}$
CDM	85.51	78.26	90.09
CDW	85.24	76.73	89.31
Fusion	86.4	79.16	89.3

Table 5.2: Results for Restaurant Dataset Training.

To understand the model's performance in both datasets, some reviews from My-FitnessPal - Calorie Counter[3] were manually extracted and provided to the aspect extractor to overview the model behavior when extracting aspects and classifying the same reviews for different trained checkpoints. The CDM checkpoint was chosen for the restaurant dataset and the fusion one for the laptop. The results are presented in the tables down below.

[3] `https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=en_US&gl=GB`

Review	Aspects	Sentiment
“I think this is a great app and easily the best free option around. However, I’m not a fan of the new look. I’m happy to spend some time getting used to it but two things in particular bother me. 1) The removal of the + which allowed you to quickly add weight and food etc from the home screen. 2) I may be missing it , but I can’t find a simple weight progress update . And I’m sure this used to be easily visible. Something that says current weight - loss so far etc .”	[’weight’, ’food’, ’, but’]	[’Negative’, ’Negative’, ’Negative’]
“The app is great and has been a big help with me starting out in the gym a few months ago. The dashboard has recently changed and though it does look better , it is not very smooth to scroll through. The other pages run as smooth as butter when scrolling but the dashboard needs improving quite bad.”	[’app’, ’dashboard’, ’pages’, ’dashboard’]	[’Positive’, ’Negative’, ’Negative’, ’Negative’]
“I always liked this app, even though there were times its filtering and history worked poorly. But the recent update has been so badly done that I’ve given up. The dashboard is unhelpful, and many other aspects are nearly unusable. I’ve gone to another app (Loseit). I might come back if the new app gets as foolish as MFP, but MFP will need to return to its former ease of use first”	[’filtering’, ’history’, ’dashboard’, ’to return’, ”]	[’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’]
“No not so good since update. I’m a type 1 Diabetic insulin pump user. I have to carbohydrate count each meal to help work out how much insulin I need. Pre - update I could turn phone around & see all meals carbohydrates and then easily work out how much insulin to give. Now the only total is calories. It’s so much more fiddly now and it’s all messy. Really don’t like new app. The simple way it was before was good!!! Really want to just give 2.5* whereas before it would have been almost 5*”	[’insulin’, ’meals carbohydrates’, ’insulin’, ’calories’, ’s all’]	[’Neutral’, ’Neutral’, ’Neutral’, ’Neutral’, ’Neutral’]
“Great for monitoring calories (and exercise). Very comprehensive. Approach to cataloguing your own meals, etc could be a little more intuitive. Handy to have the widget, but does it have to be so massive and take up so much of the phone’s home screen (4x2)?!?”	[’calories’, ’exercise’, ’meals’, ’widget’, ’home screen’]	[’Positive’, ’Positive’, ’Neutral’, ’Positive’, ’Neutral’]
“Recent update removed compact widget, now it takes up a 1/3 of my screen which I absolutely hate. It showed the exact same info but just takes up an unnecessary amount of space. Aside from that it’s good for tracking your calories. Wish their cache would store more of your recently added items, would be a very simple thing to implement.”	[’widget’, ’screen’, ’space’, ’calories’, ’cache’]	[’Negative’, ’Negative’, ’Neutral’, ’Negative’]

Table 5.3: Inference results using model trained on restaurant data.

Review	Aspects	Sentiment
“I think this is a great app and easily the best free option around. However, I’m not a fan of the new look. I’m happy to spend some time getting used to it but two things in particular bother me. 1) The removal of the + which allowed you to quickly add weight and food etc from the home screen. 2) I may be missing it , but I can’t find a simple weight progress update . And I’m sure this used to be easily visible. Something that says current weight - loss so far etc .”	[’screen’, ”, ”, ”]	[’Negative’, ’Negative’, ’Negative’, ’Negative’]
“The app is great and has been a big help with me starting out in the gym a few months ago. The dashboard has recently changed and though it does look better , it is not very smooth to scroll through. The other pages run as smooth as butter when scrolling but the dashboard needs improving quite bad.”	[’app’, ’dashboard’, ’look’, ’scroll’, ’pages run’, ’scrolling’, ’dashboard’]	[’Positive’, ’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’]
“I always liked this app, even though there were times its filtering and history worked poorly. But the recent update has been so badly done that I’ve given up. The dashboard is unhelpful, and many other aspects are nearly unusable. I’ve gone to another app (Loseit). I might come back if the new app gets as foolish as MFP, but MFP will need to return to its former ease of use first”	[’filtering’, ’history’, ’dashboard’, ’app’, ”, ”, ”, ”]	[’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’]
“No not so good since update. I’m a type 1 Diabetic insulin pump user. I have to carbohydrate count each meal to help work out how much insulin I need. Pre - update I could turn phone around & see all meals carbohydrates and then easily work out how much insulin to give. Now the only total is calories. It’s so much more fiddly now and it’s all messy. Really don’t like new app. The simple way it was before was good!!! Really want to just give 2.5* whereas before it would have been almost 5*”	[’insulin’, ’calories’, ”, ”, ”]	[’Negative’, ’Negative’, ’Negative’, ’Negative’, ’Negative’]
“Great for monitoring calories (and exercise). Very comprehensive. Approach to cataloguing your own meals, etc could be a little more intuitive. Handy to have the widget, but does it have to be so massive and take up so much of the phone’s home screen (4x2)?!?”	[’widget’, ’home screen’]	[’Positive’, ’Negative’]
“Recent update removed compact widget, now it takes up a 1/3 of my screen which I absolutely hate. It showed the exact same info but just takes up an unnecessary amount of space. Aside from that it’s good for tracking your calories. Wish their cache would store more of your recently added items, would be a very simple thing to implement.”	[’widget’, ’screen’, ’cache’]	[’Negative’, ’Negative’, ’Negative’]

Table 5.4: Inference results using model trained on laptop data.

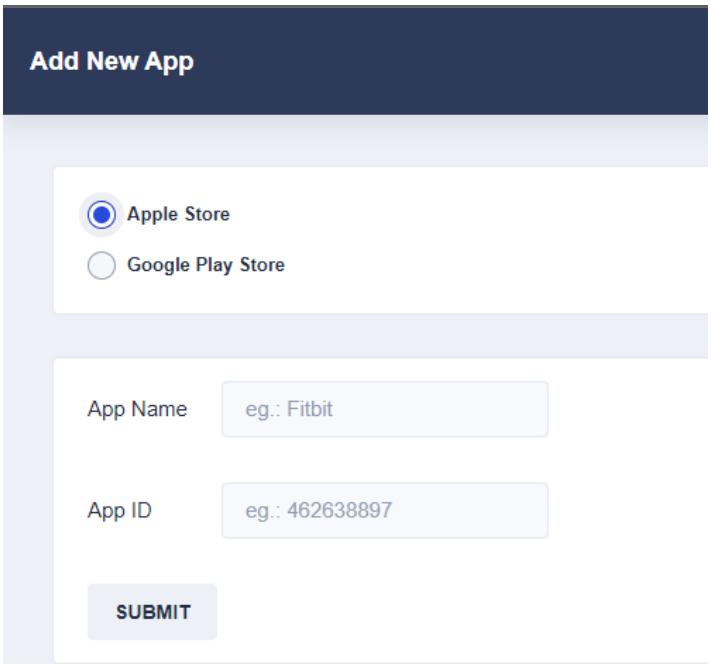
5.2.1 Homepage

When user accesses the homepage it displays a card view with all the apps stored in the system and a card to add a new app.

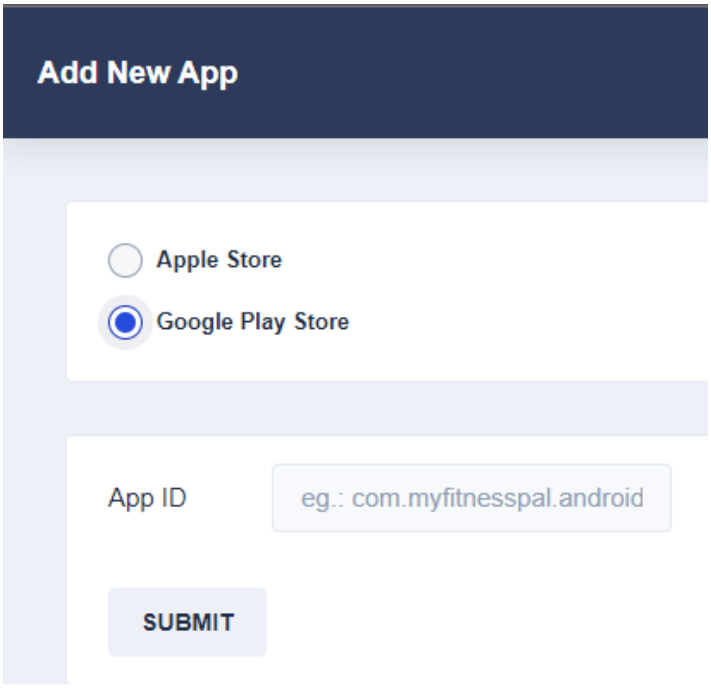
Figure 5.1: Homepage View.

5.2.2 Submit a New App

When adding a new app to the system, the user can choose whether its from Google Play Store or App Store. When submitting the app, the system will classify and extract the aspects. At the end, the user is redirected to the homepage.



(a) Apple Store App Form.



(b) Google Play Store App Form.

Figure 5.2: Forms to submit a new app to the system.

5.2.3 App Aspect Extraction and Classification Details

When selecting an app from the homepage card view, the user is redirected to a page that displays the top 10 positive and negative aspects. The system will fetch new reviews in background. When it ends, the app displays a notification informing how many new reviews were found. After refresh, the tables are updated. If there are no new reviews, the app displays a notification announcing that no new reviews were found.

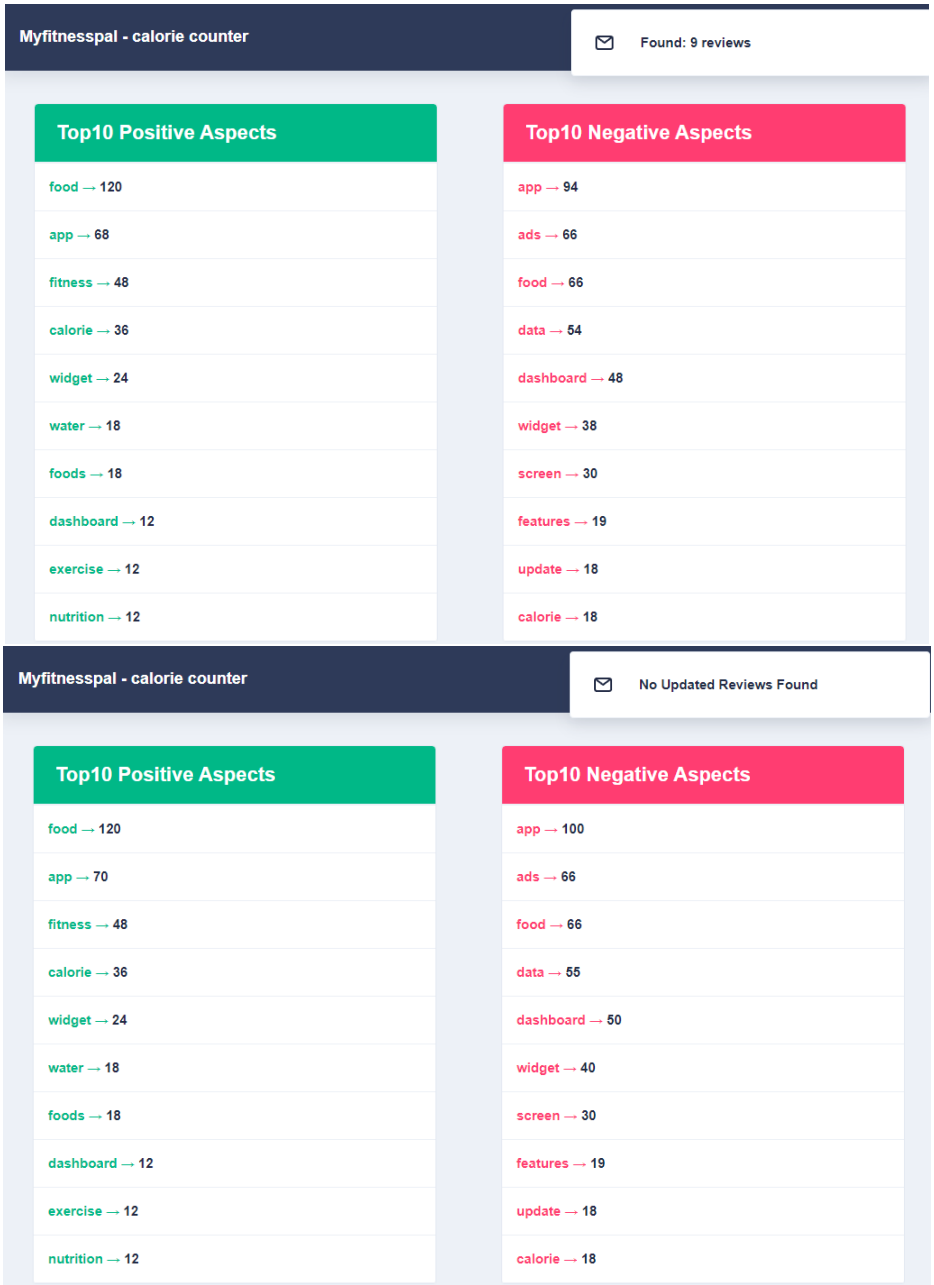


Figure 5.3: App Details View.

CHAPTER 6

Conclusions

This thesis aimed to integrate methods and technologies for collecting information with methods of information extraction and sentiment analysis in a web application that would allow the exploration of data concerning health and well-being apps reviews.

First, a study about sentiment analysis methods was performed. In the first instance, classical methods were described. Then, a search was carried out for actual methods that performed better results. The model best suited the objective was the LCF-ATEPC model due to its results and its entire associated library, pyabsa, which provides the developer training and inference methods.

Then, on the collecting information task, the strategy adopted was to find something that would allow collecting data from where the information was reliable and constantly updated so that the data collected is always a representation of the current state of the apps. The app store scrapers were the perfect match for the project.

All the research resulted in the architecture demonstrated in chapter 4. It was possible to implement a system that allows, through a web application, to add an app and to have the perception of the most relevant aspects, both positive and negative.

The disadvantage of the current system is that it does not have annotated data and has not been trained with datasets within the pretended domain. This fact can lead to some inconsistency in the data since the positive and negative aspects extracted may not fully represent the actual state of the app. However, the advantage of this system is that it has a solid basis for future work and can be easily reused and even more explored. For example, at the web app level, to present other kind of information and add other features, such as annotating data directly from the web app or displaying the comments from where the most relevant aspects were extracted. Besides that, improvements of the classification model should be considered, with annotated and training data within the desired domain.

In conclusion, the proposed objectives were met, and there are improvements to be made in future works to provide a more reliable solution.

Bibliography

[1] W. Medhat, A. Hassan, and H. Korashy, «Sentiment analysis algorithms and applications: A survey», *Ain Shams Engineering Journal*, vol. 5, pp. 1093–1113, 4 Dec. 2014, ISSN: 20904479. DOI: 10.1016/j.asej.2014.04.011.

[2] B. C. Zapata, J. L. Fernández-Alemán, A. Idri, and A. Toval, «Empirical Studies on Usability of mHealth Apps: A Systematic Literature Review», *Journal of Medical Systems*, vol. 39, pp. 1–19, 2 Feb. 2015, ISSN: 1573689X. DOI: 10.1007/s10916-014-0182-2.

[3] A. Muntaner-Mas, A. Martinez-Nicolas, C. J. Lavie, S. N. Blair, R. Ross, R. Arena, and F. B. Ortega, *A Systematic Review of Fitness Apps and Their Potential Clinical and Sports Utility for Objective and Remote Assessment of Cardiorespiratory Fitness*, Apr. 2019. DOI: 10.1007/s40279-019-01084-y.

[4] E. D. Liddy, «SURFACE SURFACE Center for Natural Language Processing School of Information Studies (iSchool) 2001 Natural Language Processing Natural Language Processing Natural Language Processing 1». [Online]. Available: https://surface.syr.edu/cnlp.

[5] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, *Natural language processing: An introduction*, Sep. 2011. DOI: 10.1136/amiajnl-2011-000464.

[6] A. Montoyo, P. Martínez-Barco, and A. Balahur, «Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments», *Decision Support Systems*, vol. 53, no. 4, pp. 675–679, 2012, 1) Computational Approaches to Subjectivity and Sentiment Analysis 2) Service Science in Information Systems Research : Special Issue on PACIS 2010, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2012.05.022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923612001339.

[7] U. Yaqub, V. Atluri, S. A. Chun, and J. Vaidya, «Sentiment based Analysis of Tweets during the US Presidential Elections», vol. Part F128275, Association for Computing Machinery, Jun. 2017, pp. 1–10, ISBN: 9781450353175. DOI: 10.1145/3085228.3085285.

[8] «Social Networking and Constituent Communications: Member Use of Twitter During a Two-Month Period in the 111 th Congress», 2010. [Online]. Available: https://crsreports.congress.gov.

[9] R. Prabowo and M. Thelwall, «Sentiment analysis: A combined approach», *Journal of Informetrics*, vol. 3, pp. 143–157, 2 Apr. 2009, ISSN: 17511577. DOI: 10.1016/j.joi.2009.01.003.

[10] Q. T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, and A. Rehman, «Sentiment Analysis Using Deep Learning Techniques: A Review», 2017. [Online]. Available: www.ijacsa.thesai.org.

[11] «Sentiment Analysis of Restaurant Review with Classification Approach in the Decision Tree-J48 Algorithm», Institute of Electrical and Electronics Engineers Inc., Sep. 2019, pp. 121–126, ISBN: 9781728138329. DOI: 10.1109/ISEMANTIC.2019.8884282.

[12] N. Akhtar, N. Zubair, A. Kumar, and T. Ahmad, «Aspect based Sentiment Oriented Summarization of Hotel Reviews», vol. 115, 2017. DOI: 10.1016/j.procs.2017.09.115.

[13] A. Z. Ahmed and M. Rodríguez-Díaz, «Significant labels in sentiment analysis of online customer reviews of airlines», *Sustainability (Switzerland)*, vol. 12, pp. 1–18, 20 Oct. 2020, ISSN: 20711050. DOI: 10.3390/su12208683.

[14] E. Guzman and W. Maalej, «How do users like this feature? A fine grained sentiment analysis of App reviews», Institute of Electrical and Electronics Engineers Inc., Sep. 2014, pp. 153–162, ISBN: 9781479930333. DOI: 10.1109/RE.2014.6912257.

[15] C. Liu, Q. Zhu, K. A. Holroyd, and E. K. Seng, «Status and trends of mobile-health applications for iOS devices: A developer's perspective», *Journal of Systems and Software*, vol. 84, pp. 2022–2033, 11 Nov. 2011, ISSN: 01641212. DOI: 10.1016/j.jss.2011.06.049.

[16] C. C. M. I. Strategist, «Health & Fitness app adoption up record 47 % so far in Q2 2020», *Sensor Tower Blog*, [Online]. Available: https://sensortower.com/blog/health-and-fitness-app-record-download-growth.

[17] M. Paschou and E. Sakkopoulos, «Personalized assistant apps in healthcare: a Systematic Review», in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2019, pp. 1–8. DOI: 10.1109/IISA.2019.8900728.

[18] M. Camacho-Rivera, H. Vo, X. Huang, J. Lau, A. Lawal, and A. Kawaguchi, «Evaluating asthma mobile apps to improve asthma self-management: User ratings and sentiment analysis of publicly available apps», *JMIR mHealth and uHealth*, vol. 8, 10 Oct. 2020, ISSN: 22915222. DOI: 10.2196/15076.

[19] D. Maynard and A. Funk, «LNCS 7117 - Automatic Detection of Political Opinions in Tweets», 2011. [Online]. Available: http://twittersentiment.appspot.com/.

[20] I. j. R. Avani Jadeja, «Feature Based Sentiment Analysis On Customer Feedback: A Survey», 2013. [Online]. Available: www.zdnet.com.

[21] W. Medhat, A. Hassan, and H. Korashy, «Sentiment analysis algorithms and applications: A survey», *Ain Shams Engineering Journal*, vol. 5, pp. 1093–1113, 4 Dec. 2014, ISSN: 20904479. DOI: 10.1016/j.asej.2014.04.011.

[22] Z. Madhoushi, A. R. Hamdan, and S. Zainudin, «Sentiment analysis techniques in recent works», Institute of Electrical and Electronics Engineers Inc., Sep. 2015, pp. 288–291, ISBN: 9781479985470. DOI: 10.1109/SAI.2015.7237157.

[23] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, «Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis», 2011.

[24] H. Kaur, V. Mangat, and Nidhi, «A survey of sentiment analysis techniques», in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017. DOI: 10.1109/I-SMAC.2017.8058315.

[25] M. D. Devika, C. Sunitha, and A. Ganesh, «Sentiment Analysis: A Comparative Study on Different Approaches», vol. 87, Elsevier B.V., 2016, pp. 44–49. DOI: 10.1016/j.procs.2016.05.124.

[26] A. F. Anees, A. Shaikh, A. Shaikh, and S. Shaikh, «Survey paper on sentiment analysis: Techniques and challenges», *EasyChair2516-2314*, 2020.

[27] F. Xianghua, L. Guo, G. Yanyan, and W. Zhiqiang, «Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon», *Knowledge-Based Systems*, vol. 37, pp. 186–195, 2013, ISSN: 09507051. DOI: 10.1016/j.knosys.2012.08.003.

[28] Y. Ko and J. Seo, «Automatic Text Categorization by Unsupervised Learning».

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[30] *Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review*, Mar. 2019. DOI: 10.1016/j.eswa.2018.10.003.

[31] L. Zhang, S. Wang, and B. Liu, «Deep learning for sentiment analysis: A survey», *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, 4 Jul. 2018, ISSN: 19424795. DOI: 10.1002/widm.1253.

[32] Y. Goldberg, «A Primer on Neural Network Models for Natural Language Processing», 2016, pp. 345–420.

[33] O. Habimana, Y. Li, R. Li, X. Gu, and G. Yu, *Sentiment analysis using deep learning approaches: an overview*, Jan. 2020. DOI: 10.1007/s11432-018-9941-6.

[34] T. Yu, C. Hidey, O. Rambow, and K. McKeown, «Leveraging Sparse and Dense Feature Combinations for Sentiment Classification», Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.03940.

[35] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, «Enhancing deep learning sentiment analysis with ensemble techniques in social applications», *Expert Systems with Applications*, vol. 77, pp. 236–246, Jul. 2017, ISSN: 09574174. DOI: 10.1016/j.eswa.2017.02.002.

[36] Y. Hua, J. Guo, and H. Zhao, «Deep Belief Networks and deep learning», Institute of Electrical and Electronics Engineers Inc., May 2015, pp. 1–4, ISBN: 9781479975334. DOI: 10.1109/ICAIOT.2015.7111524.

[37] S. Liao, J. Wang, R. Yu, K. Sato, and Z. Cheng, «CNN for situations understanding based on sentiment analysis of twitter data», *Procedia Computer Science*, vol. 111, pp. 376–381, 2017, The 8th International Conference on Advances in Information Technology, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.06.037. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917312103.

[38] Y. Yu, C. Hu, X. Si, J. Zheng, and J. Zhang, «Averaged Bi-LSTM networks for RUL prognostics with non-life-cycle labeled dataset», *Neurocomputing*, vol. 402, pp. 134–147, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.03.041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220303945.

[39] P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, *On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems*, 2016. arXiv: 1605.07669 [cs.CL].

[40] P. K. Jain, «84 A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents», 2021. DOI: 10.1145/3457206. [Online]. Available: https://doi.org/10.1145/3457206.

[41] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, «Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers», *IEEE Access*, vol. 8, pp. 131 662–131 682, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3009626.

[42] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, «End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results», *CoRR*, vol. abs/1412.1602, 2014. arXiv: 1412.1602. [Online]. Available: http://arxiv.org/abs/1412.1602.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, «Attention Is All You Need», Jun. 2017. [Online]. Available: http://arxiv.org/abs/1706.03762.

[44] M. Phi, *Illustrated guide to transformers- step by step explanation*, 2020. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding», Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.04805.

[46] S. Alaparthi and M. Mishra, «Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey».

[47] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, «SemEval-2014 Task 4: Aspect Based Sentiment Analysis», in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 27–35. DOI: 10.3115/v1/S14-2004. [Online]. Available: https://aclanthology.org/S14-2004.

[48] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, «SemEval-2015 Task 12: Aspect Based Sentiment Analysis», in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 486–495. DOI: 10.18653/v1/S15-2082. [Online]. Available: https://aclanthology.org/S15-2082.

[49] H. Yang, B. Zeng, M. Xu, and T. Wang, «Back to Reality: Leveraging Pattern-driven Modeling to Enable Affordable Sentiment Dependency Learning», Oct. 2021. [Online]. Available: http://arxiv.org/abs/2110.08604.

[50] H. Yang, B. Zeng, J. Yang, Y. Song, and R. Xu, «A multi-task learning model for Chinese-oriented aspect polarity classification and aspect term extraction», *Neurocomputing*, vol. 419, pp. 344–356, Jan. 2021, ISSN: 18728286. DOI: 10.1016/j.neucom.2020.08.001.

[51] E. H. Silva and R. M. Marcacini, «Aspect-based Sentiment Analysis using BERT with Disentangled Attention», 2021.

[52] J. Dai, H. Yan, T. Sun, P. Liu, and X. Qiu, «Does syntax matter? A strong baseline for Aspect-based Sentiment Analysis with RoBERTa», Apr. 2021. [Online]. Available: http://arxiv.org/abs/2104.04986.

[53] B. Xing and I. W. Tsang, «Understand me, if you refer to Aspect Knowledge: Knowledge-aware Gated Recurrent Memory Network», Aug. 2021. [Online]. Available: http://arxiv.org/abs/2108.02352.

[54] O. Wallaart, F. Frasincar, and. 758x, «A Hybrid Approach for Aspect-Based Sentiment Analysis Using a Lexicalized Domain Ontology and Attentional Neural Models». [Online]. Available: https://github.com/ofwallaart/HAABSA..

[55] N. Reddy, P. Singh, and M. M. Srivastava, «Does BERT Understand Sentiment? Leveraging Comparisons Between Contextual and Non-Contextual Embeddings to Improve Aspect-Based Sentiment Models», Nov. 2020. [Online]. Available: http://arxiv.org/abs/2011.11673.

[56] M. M. Trusca, D. Wassenberg, F. Frasincar, and R. Dekker, «A Hybrid Approach for Aspect-Based Sentiment Analysis Using Deep Contextual Word Embeddings and Hierarchical Attention», Apr. 2020. [Online]. Available: http://arxiv.org/abs/2004.08673.

[57] R. C. Moore and J. Denero, «L 1 AND L 2 REGULARIZATION FOR MULTICLASS HINGE LOSS MODELS».

[58] E. Lim, *App-store-scraper*. [Online]. Available: https://pypi.org/project/app-store-scraper/.