Master Thesis

Fake news detection and analysis

Author:

Ștefan Elena-Ramona

Master in Research and Innovation

Facultat d'Informatica de Barcelona

Universitat Politècnica de Catalunya

Advisor:

Prof. Dr. Jordi Turmo Borrás

GPLN - Natural Language Processing Group

IDEAI - Intelligent Data Science and Artificial Intelligence Research Center

Department of Computer Science

Universitat Politècnica de Catalunya

Co-Advisors:

Prof. Dr. Eng. Mihai Dascălu

S.l. Dr. Eng. Ștefan Rușeți

Department of Computer Science

Faculty of Automatic Control and Computers

University Politehnica of Bucharest

2022

Contents

List of tables

List of figures

Abstract

The evolution of technology has led to the development of environments that allow instantaneous communication and dissemination of information. As a result, false news, article manipulation, lack of trust in media and information bubbles have become high-impact issues. In this context, the need for automatic tools that can classify the content as reliable or not and that can create a trustworthy environment is continually increasing. Current solutions do not entirely solve this problem as the degree of difficulty of the task is high and dependent on factors such as type of language, type of news or subject volatility. The main objective of this thesis is the exploration of this crucial problem of Natural Language Processing, namely false content detection and of how it can be solved as a classification problem with automatic learning. A linguistic approach is taken, experimenting with different types of features and models to build accurate fake news detectors. The experiments are structured in the following three main steps: text pre-processing, feature extraction and classification itself. In addition, they are conducted on a real-world dataset, LIAR, to offer a good overview of which model best overcomes day-to-day situations. Two approaches are chosen: multi-class and binary classification.

In both cases, we prove that out of all the experiments, a simple feed-forward network combined with fine-tuned DistilBERT embeddings reports the highest accuracy – 27.30% on 6-labels classification and 63.61% on 2-labels classification. These results emphasize that transfer learning bring important improvements in this task. In addition, we demonstrate that classic machine learning algorithms like Decision Tree, Naïve Bayes, and Support Vector Machine act similar with the state-of-the-art solutions, even performing better than some recurrent neural networks like LSTM or BiLSTM. This clearly confirms that more complex solutions do not guarantee higher performance. Regarding features, we confirm that there is a connection between the degree of veracity of a text and the frequency of terms, more powerful than their position or order. Yet, context prove to be the most powerful aspect in the characteristic extraction process. Also, indices that describe the author's style must be carefully selected to provide relevant information.

1 Introduction

Information refers to all the facts, knowledge and opinions perceived from another living being, from media or just from reading and it is the crucial component of any communication. Nowadays, a very large part of the data interchange process has moved from the physical environment to the digital one, leading to online communities of considerable size. Given the evolution of technology, instantaneous environments for distribution of information have developed. As a result, the impact of news considerably increased compared to the classic case of discussion between a small number of people. Therefore, it became mandatory to keep this exchange of information under control.

1.1 Context

False content has gained considerable momentum and has become a current issue for both citizens and internal security. The term “fake news” can be defined as “a story invented with the intent to deceive”, according to the New York Times, “purposefully crafted, emotionally charged misleading information” or “a completely invented story, manipulated to resemble credible journalism and to attract maximum attention and implicitly advertising revenue” (Agarwal, Sultana, Malhotra, & Sarkar, 2019). In the classic way, the problem of determining the authenticity of an article can be described as the prediction of the chances that a certain article will intentionally contain false information. Having various patterns and subjects, this task can easily become subjective.

Before analyzing diverse methods for solving this problem, it is important to present the origins of the subject. Thus, although the term “fake news” became popular in 2016, with the presidential election in the United States, the misleading content is not a recent issue. For instance, the document entitled “Zinoviev’s Letter”, was published in 1924 in a famous British newspaper, just a few days before the general election, in order to destabilize the political situation in favor of the Conservative Party (Norton-Taylor, 1999). Another example is a newspaper article that later turned out to be fake, published after the “Hilsborough accident” where ninety-six people were crushed to death due to overcrowding. The article claimed that as people were dying, some drunken colleagues robbed them and beat the police (Torgo, Guimaraes, & Figueira, 2018). Over time, this practice has become increasingly common, now being amplified by technological development.

Thanks to the large number of users, and the overwhelming amount of information available daily, social websites have become a recurring and fast method of creating and easily spreading false news to a very large group of people. The fact that the information is one click away, whether it is on a phone or a computer, makes the process of consuming large amounts of information easy to be done. The most used platforms in this regard are Facebook and Twitter. Studies have shown that over 65% of adults read their daily news from social media, percentage increasing each year (Shearer & Gottfriend, 2017). What is most worrying is that, according to the research, people are not able to detect the lies in the text, being only 4% better than the chance at this problem (Bond & DePaulo, 2006). Malicious users quickly took advantage of the situation and started spreading rumors and

misinformation. For example, OpenAI has announced the launch of a GPT-3 tool that can produce almost indistinguishable text from human writing. According to them, GPT-3 can identify the relationships between concepts and the context of communication. Such a device, coordinated by the wrong hands, would be an extremely dangerous weapon in the process of misinformation, spam, or phishing.

False news, article manipulation, lack of trust in the media, and information bubbles are growing problems with high impact. The most common problem that makes it extremely difficult to detect unauthentic content for both humans and automated tools is that fake news almost never looks the same, fact which slows down the evolution of solutions. A frequent, but not exclusive, classification of these would be the following (Vorhies, 2017):

- Biased messages/comments/opinions - biased reactions to various current events/locations/products
- Fake news/Propaganda - news intentionally written to spread erroneous or misleading information about an event/personality/topic; in most cases it promotes a current, a personality, an extremist, a subjective vision
- Clickbait - articles meant to attract attention, to shock by the title in order to generate a large flow of people on that site and high advertising revenue; in most cases they are completely wrong, based on exaggeration of real events
- Humor/Satire - articles written for fun, not meant to present real information

In this paper, we will focus on the category of fake news that tries to manipulate people by spreading untrustworthy information. Broadly speaking, this issue can be described as a simple task of classifying articles along a continuum of truthfulness, with each news item being associated with a measure of certainty, greater or lesser, depending on the amount of false information intentionally used in it (Conroy, Rubin, & Chen, 2016). Given that most research in this field focuses on written text, the detection of false content is considered a classic problem of Artificial Intelligence (AI), more precisely Natural Language Processing (NLP). Thus, the basic theoretical ideas of NLP are used: the decomposition of language into elementary units, the determination of the relations between them and the deduction of their meaning.

1.2 Objectives

The main objective of this thesis is to explore the natural language issue of false content identification in a specific field, namely media news. The problem is structured as a classification problem with 6, respectively 2 classes. In addition, we intend to analyze various approaches to find the reasons why certain techniques and models have higher performance, highlighting their strengths and weaknesses. This objective is accomplished mainly by means of experimentation with different type of models, features, and pre-processing operations to improve accuracy.

2 State of the Art

The impact of misinformation has prompted both scientific community and big companies (e.g.: Facebook, Google) to look for a way to alleviate the problem. Thus, different ways of assessing the truth value of a text began to appear.

One attempt to eliminate deception dissemination is represented by systems that track social media accounts or false content propagation models to identify bot or spam profiles. One of the first approaches in this direction was in 2010, when the group of Benevenuto used a non-linear Support Vector Machine (SVM) with a Radial Basis Function kernel to detect spam accounts based on attributes regarding the behavior of the user. The model was able to correctly detect 70% of spam accounts and 96% of non-spam accounts from a dataset with over 1.000 entries (Benevenuto, Magno, Rodigues, & Almeida, 2010). The paper "Who is tweeting on Twitter: human, bot, or cyborg?" from the same year was also among the initial solutions which distinguished non-human profiles. The authors proposed a classification system with four parts:

- Entropy component - detects regular posting times of users (sign of automation)
- Machine learning component (Bayesian classification) – detects text patterns
- Account properties component – analyses properties to find bot deviation from normal human distribution
- Decision maker component (Linear Discriminant Analysis) – makes a final decision based on the other three components

This system achieved 96% accuracy in human detection (Chu, Gianvecchio, Jajodia, & Wang, 2010). Other more detailed model can be analyzed in the work of Ma et al. They represented news with propagation trees to follow how a message is altered by users when transmitted over time. For classifying, they appealed to Propagation Tree Kernel to compute the similarity with rumor trees and non-rumor trees depending on structural and linguistic properties (Ma, Gao, & Wong, 2010). The group evaluated the solution on two different datasets and registered accuracies between 73-75%.

Later, the group of Castillo aimed to identify the credibility of Twitter posts by using only user information, subject, and propagation metadata. Along with a decision tree (DT) that shaped a set of rules, they achieved an accuracy of about 86% in terms of detecting bot users (Castillo, Mendoza, & Poblete, 2013). More recently, Antoniadis et al. built models based on tweet features (number of characters/ words/ likes/ retweets/ replies/ mentions/ URLs/ media/ hashtags), user features (number of followers/ followees/ total tweets/ tweets during an event, days registered, followers-followees ratio) and additional features (URLs, media, average tweets per day, positive/negative/average sentiment). Their strategy included Nayve Bayes, K-Nearest Neighbors, Adaptive Boosting, Random Forest, Bootstrap Aggregating, J48, all accomplishing a F1-score over 70% (Antoniadis, Litou, & Kalogeraki, 2015). In 2017, Nayve Bayes classifiers were still a go-to method in this direction, the extracted features being the only variations. Ersahin, Aktaş, Kilinç and Akyol used the Entropy Minimization Discretization technique on numerical features to improve the process

of building rules for fake news detection, maximizing information gain (Ersahin, Aktaş, Kilinç, & Akyol, 2017).

Although the results seem very good, this approach has a major disadvantage. More precisely, to identify a spam account, it must publish enough fake information to build a meaningful profile. Indeed, once identified, the source can be removed, but previously spread news can no longer be stopped. In addition, a new bot account can be created immediately to continue the process indefinitely. Nowadays solutions are more oriented on classifying the articles based mostly on their content. This is the approach that we also choose for this thesis, and which will be detailed further. The most relevant techniques and their related results will be described in the following sections while highlighting some of the most relevant resources.

2.1 Resources

First, we require complete and representative resources. Based on this idea, numerous people started providing data sources that reflect different real-life situations of false content dissemination. Some of the most frequent topics are political news, medical news, advertisement, or celebrity journalism. The interest for the subject is so big that it led to shared tasks and important academic competitions for the NLP community. We collected some of the most famous ones in the following sections. Our focus was on English as it is the most common foreign language and it has a wide use in writing international news and in social media. Besides that, we mention a couple of Spanish resources as results from a very famous competition with the same theme which took place for several consecutive years.

2.1.1 Competitions

The main benefit of competitions is that they provide a common evaluation framework for people who try to advance the state of the art in a particular topic, in our case by racing in getting the best fake content detection system. Despite having a great number of participants that work on a problem, sometimes the progress in the area is not sufficient and some datasets have to be further analyzed. In this section, we present important challenges focused on this shared task of falsity detection in order to see the points of interest.

One of the pioneers in this area of competitions was “Fake news challenge stage 1 (FNC-I)” which was held in 2017. Being one of the first attempts, the main task was simplified by being split into consecutive stages. That edition of the competition – and the only one until now - focused only on the first step, respectively estimating if two texts claim the same thing about a specific topic. More precisely, the two inputs were a title and a body text. After comparing these, they were marked with one of the following labels: “agrees”, “disagrees”, “discusses” and “unrelated”. The purpose was to find trustworthy relations between a valid information and a new one. The evaluation metric was a weighted score which considered both if the two ideas were related or not (25%) and if they agreed on the topic (75%). Out of 80 competitors, the team SOLAT in the SWEN scored 82.02%.

Another remarkable competition is “KDD 2020 TrueFact Workshop: Making a Credible Web for Tomorrow” that took place in 2020. For this shared task, the teams had to design a solution that can distinguish inauthentic claims by classifying them as true or fake.

This time, the dataset contained 6.234 examples of label-text pairs. The competition had 19 teams which were evaluated using accuracy as performance metric. The first classified team achieved a score of 84%.

We can say that 2020 was a great year for fake news competitions. “Fakeddit Multimodal Fake News Detection Challenge 2020” is another example of organized framework which brought important resources for this subject. This competition launched a dataset called Fakeddit which contained over 1 million samples of fake news. Its purpose was wider as it tried to find different models that detect fake news in both text and images. The performance metric used is the same, meaning accuracy, measuring the percentage of text-image pairs the model can recognize as fake. This competition deviates from our scope as we will use only text for our classifiers.

One competition that gained much popularity on the subject and encouraged people to invest time in this problem is IberLEF which adopted this theme for two years in a row. Indeed, its real purpose was to provide a competitive text processing framework to overcome the actual state of the art results. Fake news detection was among the proposed challenges as “MEX-A3T: Fake News and Aggressiveness Analysis” (2020) and “FakeDeS: Fake News Detection in Spanish Shared Task” (2021). In both years, the F1 measure over the class of interest was used to rank participants. In 2020, all competitors used The Spanish Fake News Corpus which was composed of only 971 Mexican Spanish articles collected from different news related websites. The dataset was manually annotated in two balanced classes: true and fake. Besides the text and the label, it also contained the topic, the title, and the source URL. Only nine teams took the challenge of that edition and participated, all of them obtaining a F1-score over 70%. The highest value was achieved by the Idiap-UAM-2 team with a value of 84.44%. At the second edition, the attention was shifted on a hot subject at that time, meaning the pandemic situation. This time, the organizers tried to evaluate the robustness of the solutions by training on a very general set (The Spanish Fake News Corpus 2020) and evaluating on a very specific topic (Covid-19). The new testing dataset was composed of Coronavirus related news from Ibero-American countries classified in the same categories. This time, the results were poorer than those of the previous year. The number of participants elevated to 21 but the increased difficulty of the task led to F1-scores between 48.38% and 76.66%. The GDUFS_DM team achieved the performance of winning the competition with their attention-based solution.

2.1.2 Datasets

Through these competitions or just through simple scientific articles, more and more datasets and web services are becoming available to identify fake content. As it is important to review the most important ones that addressed this topic, we gathered a series of datasets in Table 1. It can be seen that the messages and threads posted on Twitter are an active source of data. On the news side, for most corpora, the main source is PolitiFact.com, and as secondary sources we mention CNN or NYTimes.

There are also a multitude of datasets that cross the barrier of authentic news - inauthentic news, resorting to style and content details and annotating data more precisely,

such as: mostly-true, barely-true, unverified rumour, half-true, false rumor or check-worthy factual sentence. These fine differences between the texts obviously made the task more difficult but lead to more representative results. The length of the datasets varies from only 360 to over 9 millions of examples, fact which influences their performance. As it is easier to recognize patterns from a low number of very specific news, if the corpus is more general then a reduced number of texts may lead to poor results. On the other hand, very large datasets are hard to process as they need important computational resources.

Table 1 Analyze of existing datasets

Lang.	Dataset / Article	Dimension	News type	Labels	Performance metric
ENGLISH	**Predicting information credibility in time-sensitive social media** (Castillo, Mendoza, & Poblete, 2013)	1.873.000 messages	Twitter posts	Credible/ Not-Credible	**89.90% F1-score**
	A Model for Identifying Misinformation in Online Social Networks (Antoniadis, Litou, & Kalogeraki, 2015)	59.660 users & 80.294 messages	Twitter posts	Credible/ Misinformation	**78.00% F1-score**
	Detect rumors in microblog posts using propagation structure via kernel learning (Ma, Gao, & Wong, 2010), (Rum Detect, 2010)	450.150 users and 2.308 messages	Twitter posts	Non-Rumor/ False Rumor/ True Rumor/ Unverified Rumor	**73.20% Accuracy**
	BuzzFeed-Webis Fake News Corpus 2016 (Potthast, Kiesel, Bevendorff, Stein, & Reinartz, 2018), (BuzzFeed-Webis Fake News Corpus, 2016)	1.627 articles	News articles (ABC News, CNN, Politico, Addicting Info, Occupy Democrats, The Other 98%, Eagle Rising, Freedom Daily, Right Wing News)	True/ False/ Mix	**75.00% Accuracy**
	Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. (Rubin, Conroy, Cornwell, & Chen, 2017), (Satirical Fake and Legitimate News Dataset, 2016)	360 articles	News articles (The Onion, The Beaverton, The Toronto Star, The New York Times)	Satirical Online News/ Legitimate Online News	**93.00% Accuracy**
	Toward Automated Fact-Checking: Detecting Check-worthy. Factula Claims by Claim Buster (Hassan, Arslan, Tremayne, & Li, 2017)	20.788 phrases	News articles (PolitiFact, CNN)	Non-Factual Sentence/ Check-worthy Factual Sentence/ Unimportant Factual Sentence	**72.00% Accuracy**

	"Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection (Wang, 2017), (LIAR Dataset, 2017)	12.800 statements	News articles (PolitiFact)	Pants-fire/ False/ Barely-true/ Half-true/ Mostly-true/ True	**27.40% Accuracy**
	Fake News Corpus (FakeNewsCorpus, 2020)	9.408.908 articles from over 745 fields	NYTimes, WebHose English News Articles	Fake/ Satire/ Bias/ Conspiracy/ State/ Junksci/ Hate/ Clickbait/ Unreliable/ Political/ Reliable	-
	Fake and real news dataset (Fake and real news dataset - Classifying the news, 2019)	44.898 articles	News articles	True/ Fake	**99.87% Accuracy**
	Fake News (Fake News - Build a system to identify unreliable news articles, 2018)	114.061 articles	News articles	True/ Fake	**99.03% Accuracy**
	Fake News Detection Challenge KDD 2020 (Fake News Detection Challenge KDD 2020. Develop a machine learning algorithm to detect fake news, 2020)	6.234 articles	News articles	True/ Fake	**84.00% Accuracy**
SPANISH	**The Spanish Fake News Corpus** (MEX-A3T: Fake News and Aggressiveness Analysis, 2020)	971 articles	News articles	True/ Fake	**84.44% F1-score**
	The Spanish Fake News Corpus + Covid-19 (FakeDeS: Fake News Detection in Spanish Shared Task, 2021)	1.544 articles	News articles	True/ Fake	**76.66% F1-score**

2.2 Analysis of News Content

The detection of an article's truthfulness based on its content and meaning is a current difficult NLP problem. Despite being widely used, the solutions that are available nowadays are very specific and do not work in general cases. However, they have safer results than their predecessors because they focus on the problem, not on its source. There are three major complementary directions that lead to promising results:

- Knowledge-based approach - uses a priori knowledge
- Machine Learning approach – uses automatic learning of extracted linguistic patterns from news content
- Hybrid approach – combines Knowledge-based and Machine Learning techniques

2.2.1 Knowledge-based Approaches

As expected, the most human-appropriate way to detect false news is to try to verify the truthfulness of the statements based on another ones. That's why researchers start using a priori recognized knowledge in their solutions. This technique of retrieving information uses an existing body of collective human knowledge to determine the truth value of new statements. The main advantage of this direction is that beside a label, it may also offer an explanation. There are two main categories for the knowledge-based methods that will be detailed in the following sections.

A Human Oriented Fact Checking

Initially, people have chosen a manual approach based on their knowledge in various fields. Depending on the author of the final label, this category can be split in: Expert Oriented Fact Checking and Crowd Sourcing Oriented Fact Checking.

Mainly, the first method requires experts to evaluate the accuracy of a news through research and study of the subject from nonpartisan data sources. The process is straightforward, meaning that a fact is labeled as certain after comparing its accuracy to one of an already fact-checked news. In 2014, Vlachos and Riedel described the entire process of expert-oriented fact checking which is decomposed in four steps: extract statements, define relevant questions, obtain answers from valid sources, and establish a verdict (Vlachos & Riedel, 2014). Important fact checking services like Snopes (https://www.snopes.com/) and PolitiFact (https://www.politifact.com/) offer quality services in this direction. Expert oriented fact checking requires a huge amount of work and time which cost when false information is spreading. Moreover, due to frequent changes of the news publication nature, the generation of huge amount of content and the diversity of formats and genres, this solution does not fit for new fake content.

Regarding the other category, crowd sourcing offers a unique opportunity to users, meaning: the possibility of discussing the truth label assigned. In this way, people have the chance to evaluate piece after piece and to signal eventual mistakes of annotation. One available website which provides this at the moment is Fiskkit (https://www.fiskkit.com/).

The downside here is that these solutions entirely rely on the wisdom of the public users (Ahmed, Hinkelmann, & Corradini, 2019).

B Computational Oriented Fact Checking

As fact checking solutions governed by human beings are time consuming, automatic systems that identify fabricated content using external structured information (e.g.: Freebase, Google's Knowledge Graph, DBpedia) represent the alternative. For very specific areas, knowledge graphs and ontology-based solutions can lead to promising results.

In 2016, the group of Shi et al. taggled the task of fake news detection in the paper "Fact Checking in Heterogeneous Information Networks". The authors defined this problem as a link-prediction task in a knowledge graph. Their knowledge graph base was composed of a high number of triples - (head entity, relation, tail entity) – which represented facts. The model used both entity and predicate information to test validity. It extracted discriminative paths from DBpedia and SemMedDB to validate the truthfulness of a statement (Shi & Weninger, 2016). For example, for the statement "Barack Obama is a Muslim", the extracted path would have been Barack Obama - Columbia University - Association of American Universities - Canada - Stephen Harper - Calgary - Naheed Nenshi - Islam (Ciampaglia, et al., 2015). The evaluation step included thousands of claims from different domains.

Later, the group of Pan came with a solution which generates three knowledge graphs from: a fake news article base, DBpedia and a true reliable news article base. Then, they used a single B-TransE model to embed the entities and the relations. A binary TransE model was trained on each of them, and the results were compared, the best one being an 80% F1-score (Pan, et al., 2018). This model, as any other from this category, assumed that all needed information was in the graph, but this is not a realistic hypothesis, as even the largest base is incomplete. The group of Etzioni presented a prediction algorithm that use knowledge and semantic web data to determine erroneous information in documents on the Internet (Etzioni, Banko, Soderland, & Weld, 2008). The paper also introduced a new extraction system, Open IE, that was capable of deriving tuples after only a pass over the corpus and without any human intervention.

Lin et al. presented a new method based on discriminant subgraph structures. The authors generalized graph fact checking rules (GFCs) into ontological graph fact checking rules (OGFCs) by adding ontological closeness and topological constraints. Moreover, they described a supervised pattern discovery algorithm to find this kind of rules. This design generated relevant subgraph patterns and dynamically selected patterns from a stream with a small update cost per pattern. They tested their proposal against a real-world knowledge base to evaluate the efficiency (Lin, Song, & Wu , 2018) .

There are several other "network effect" variables that can be used to calculate probabilities of truth. Recently, the group of Gupta proposed a study of fact checking using information crawled from Wikipedia and organized in tables. They studied frequent topics and themes in false content, emotions transmitted to the readers and metrics from network analytics. In the undirected graphs that they built; the nodes represented bigrams while edges were indicating if two nodes coexist in a single article. These strategies were applied over

2.049 untrusty texts and 12.490 real news on which they obtain promising results (Guptaa, Lib, Farnoushc, & Jiang, 2022).

In 2020, at the start of the pandemic crises, deceptive information was a real problem. Adrian Groza used Description Logics (DL) and a COVID-19 ontology to detect not-trusted information. He converted both trustworthy and untrustworthy content to DL using FRED converter, while the reasoning process was made with Racer. The approach was straightforward: if there existed a conflict between a new myth and the scientific content or between it and the ontology structure, then the piece of news was labeled as fake (Groza, 2020).

Overall, the success of knowledge-based models has been measured by their ability to assign high truth values to true statements, and the outcome may even reach 95% accuracy on specific domains and datasets (Ciampaglia, et al., 2015). However, there are some limitations in this approach, including the fact that statements must be in a pre-existing knowledge base. Also, the solution is static, and the information need to be constantly updated.

2.2.2 Machine Learning Approach

Machine Learning (ML) approaches focus on finding similar patterns that occur in false texts. Nowadays, this direction is the most plausible and common method for detecting the authenticity of a piece of news. Over time, various ML solutions were proposed and analyzed, so we just focus on those relevant for our approach. As mentioned earlier, the issue of fake content detection is in most cases treated as a classification problem.

A Classical Models

Some of the initial approaches for untrustworthy content detection include basic algorithms from supervised learning area. Their versatility and ease of understanding make them facile to be used in combination with different linguistic and structural features extracted from the text. These solutions achieve very good results in predicting the low credibility in texts and establish a promising state of the art.

In 2009, Mihalcea and Strapparava used Naïve Bayes (NB) and Support Vector Machine on a dataset created with fake news intentionally written by people. More exactly, real news was similarly rewritten, in a journalistic manner, but false. The paper proved that Naïve Bayes classifies according to the accumulated evidence of the correlation between a certain variable (e.g.: syntax) and the others present in the model by achieving an average accuracy of 70.8%. The SVM model had a similar result, a performance accuracy of 70.1% was found in terms of detecting erroneous information (Mihalcea & Strapparava, 2009). Later, the paper “Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News” revolutionized by taking into account satirical news features (absurdity, humor, grammar, negative affect and punctuation). With TF-IDF and SVM, they improved the performance obtained by Mihalcea and registered an 82% accuracy, proving that deep syntax and grammatical patterns may be good detectable signs of falsity (Rubin, Conroy, Cornwell, & Chen, 2017).

Wang tried to introduce SVMs models into multi-class classification of fake news articles. In 2017, he proposed a dataset that contained six hierarchical labels of veracity and used it with word embeddings extracted with Word2Vec and classical ML algorithms like SVM or Logistic Regression (LR). Even though their results did not exceed 25.5%, they were very similar to performance metrics from more complicated classifiers applied on their dataset (Wang, 2017). The group of Alhindi presented similar solutions on the same corpus with the same models, SVM and LR, but with poorer results (Alhindi, Petridis, & Muresan, 2018). After they combined text with extra features such as: justification of the label or metadata, their system achieved even 37% accuracy for 6-classes and 67% for the binary case. More recently, Brașoveanu and Andonie presented the results of other classical algorithms like Naïve Bayes, Decision Tree, and Random Forest with embeddings on the same complicated dataset (Brasoveanu & Andonie, 2019). The latter was the solution which had the best performance score out of these three, 24.9% accuracy.

In 2019, Perez-Rosas et al. created two datasets and used SVM for misinformation detection. The authors used various grouping methods and point distance functions for the model to actively influence the accuracy obtained. Also, the group tested different linguistic features like n-grams (unigrams, bigrams), punctuation (commas, periods, dashes, question marks, exclamation marks), psycholinguistic features (LIWC lexicon to extract the proportions of words that fall into psycholinguistic categories) and readability (number of characters, number of paragraphs). To obtain a deeper analysis of the structure of language, namely the syntax, they extracted a set of features derived from production rules based on context free grammars trees (CFG) using the Standford Parser. This meant that beyond the simple use of words, features were selected by transforming sentences into a set of rewriting rules that describe the structure of the syntax (e.g.: noun and verb phrases were rewritten by their syntactically constituent parts). The sets of rules eventually formed an analysis parser tree. The best model on this dataset achieved an accuracy of 76% and less with 5% on a dataset with fake news taken from the web, proving the usefulness of linguistic features in fake news detection (Perez-Rosas, Kleinberg, Lefevre, & Mihalcea, 2019).

Despite having very good results, classical ML algorithms may encounter problems in the learning process if the number of features is large or the data is very similar between classes.

B Deep Learning

Classifiers based on neural networks solve the limitations of those mentioned above and perform better. Mainly, in working with texts, the Recurrent Neural Networks (RNN) had a great boost. Their advantage was that, at each step, they considered the internal state of the previous one. Thus, a word influenced the words nearby, and the weights were adjusted considering this aspect in the backpropagation process. NLP took another step forward with the advent of Long Short-Term Memory (LSTM) architecture which removed the disadvantage of RNN, namely the influence of only nearby words.

In 2017, Rashkin et al. proposed a GloVe and LSTM solution that took the sequence of words and classified it. As they worked with data crawled from PolitiFact.com, a fact-

checking website that categorizes the text in six classes, the authors proposed both a multi-class classifier and a binary-classifier. The final architecture that used frequency features achieved on the test dataset a F1-score of 20%, respectively 56%. Overall, when LIWC features were introduced, their baseline solutions with TF-IDF and Naïve Bayes or Maximum Entropy had similar or even better performance metrics than LSTM for both cases (Rashkin, Choi, Jang, Volkova, & Choi, 2017).

Different variants of this architecture like: Tree LSTM, BiLSTM or Gated recurrent units (GRU) had appeared and gave even better results. All these designs were a revelation at the time of their emergence, surpassing the convolutional neural network (CNN) on the NLP side and becoming the new benchmark (baseline). One exception is the paper written by Wang in 2017. He presented a CNN architecture that obtained the highest performance in the multi-label classification task, surpassing all types of RNNs. Taking into account only the content of the news and representing it into word embeddings, he achieved an accuracy of approximately 27% (Wang, 2017).

Also, there were several innovating mixed approaches. In 2018, the paper "Multi-source multi-class fake news detection" detailed CNN and LSTM methods to merge different text-based features in order to detect low-veracity content (Karimi, Roy, Saba-Sadiya, & Tang, 2018). Last year, Aslam et al. proposed an architecture that uses BiLSTM and GRU layers to detect the fake and real news from the LIAR dataset. Although the original dataset paper mentioned that BiLSTM tends to overfit this corpus, they managed to find a version of the algorithm that scores approximately 90% accuracy for binary classification (Aslam, Khan, Alotaibi, Aldaej, & Aldubaikil, 2021).

Ruchansky et al. proposed a model called CSI – Capture, Score, Integrate - composed of three modules which: capture temporal pattern of user activity (Recurrent Neural Network), learn characteristics based on user behavior (a fully connected layer) and classify the article (Decision Tree, Support Vector Machine, Long Short-Term Memory, Gated recurrent units). This solution tried to combine the text itself with information about users' response and reaction. It was tested on real-world corpuses (Twitter, Weibo) and achieved higher results than the existing state of the art, 89.20% accuracy, respectively 95.30% (Ruchansky, Seo, & Liu, 2017).

The latest approach that workes very well and seems to dethrone LSTM neural networks is represented by Transformers networks. They have an attention mechanism that is able to retain long-term dependencies in the case of very long statements and capture the meaning of an entire sentence. In 2019, the news from PolitiFact came again to researchers' attention through the LIAR dataset. Brasoveanu and Andonie combined machine learning and semantics in a high-performant model. They tried a series of classic and deep learning models together with the text itself, speaker information, context, semantic features (sentiment polarity, sentiment subjectivity, entities, links, relations) and syntactic features. They managed to propose architectures with BiLSTM Attention and GRU Attention that avoid the problem of overfitting and obtain between 50-55% accuracy on multi-classification. Their best model used all additional information and achieved an improved performance of 64.4% with CapsNet (Brasoveanu & Andonie, 2019).

In “Overview of FakeDeS at IberLEF 2021: Fake News Detection in Spanish Shared Task”, the winner solution described an architecture based on BERT and Sample Memory with an attention mechanism which scored 76.57% accuracy (Gomez-Adorno, Posadas-Duran, Enguix, & Capetillo, 2021). In 2022, Raza and Ding launched a framework, FND-NS, based on a Transformer architecture. The authors adapted the BART model for fake news detection and trained models only with news content or combined with additional social context. The model was tested on two datasets, NELA-GT-19 and Fakeddit, and scored 74.8% accuracy (Raza & Ding, 2022).

There are also other approaches than have not been very exploited until now. For instance, the paper “Detecting Fake News with Sentiment Analysis and Network Metadata” integrated sentiment analysis, an area of research that identify opinions or emotions expressed directly or indirectly in texts, in the problem of fake news detection. The idea pursued is intuitive, namely: the authors of inauthentic articles use emotional, unintentional communication, subjective judgments, or assessments of emotional state at the time of writing. Thus, syntactic models can distinguish between sentiment and fact-based arguments by associating a learned argumentation style class. Specifically, those who wrote false and negative articles used excessive terms of emotion compared to those who wrote truthful reviews, thus exaggerating the feelings they wanted to convey. The creators looked in general for polarity and subjectivity. Their ML sentiment solution combined with a Random Forest classifier achieved a F1-score which exceeds 88% (Shrestha, 2018). Although it obtained high performance values, it tended to have a low generality, leading to poor results when detecting false information in real cases.

More recently, Anoop et al. targeted untrustworthy content detection on different kinds of affective characteristics that appeared on fake and true health news articles. Emotion features were combined with classical and deep learning models, proving that this kind of information led to improved accuracy for all classifiers (Anoop, Deepak, & Lajish, 2020). For the Covid-19 pandemic, the authors found a pattern of emotional content in narratives that supported the use of this method in stopping misinformation dissemination. This approach is based on the use and analysis of language and is very promising, especially in hybrid combinations.

2.3 Hybrid Approaches

New most studies combine machine learning and knowledge techniques in innovating architectures in order to obtain better results for certain datasets. All the previous techniques can be combined into hybrid systems that solve the weaknesses of each individual method.

In 2021, in the paper “Knowledge Enhanced Multi-modal Fake News Detection”, the authors detected dishonest content through a subgraph classification task. They converted each news into a knowledge graph. Analog, each subgraph was a news item. Finally, Han et. al trained a graph neural network to categorize each subgraph (Han, Silva, Luo, & Karunaseker, 2021). The method achieved accuracy higher than 85% for two different datasets.

Hu et al. (2021) came with CompareNet, an end-to-end graph neural model. Starting from the news content and its topic, they built a directed heterogeneous document graph for each article using Latent Dirichlet Allocation (LDA) and TAGME3 tool. Based on it, the authors proposed a heterogeneous graph attention network to learn news representation for every topic and contextual entity representations that encode the semantics of the news content. The contextual representations were later compared to the corresponding entities from the knowledge base. Finally, the topic-enriched news representation combined with the entity comparison features were fed into a classical fake news classifier (Hu, et al., 2021). Experimental results on two benchmark datasets demonstrate that CompareNet significantly outperformed state-of-the-art methods. In 2021, Shakeel and Jain described a solution with a Fast-TransE model and machine learning algorithms (SVM and LR). This system achieved an F1-score of approximately 82%, a very good result for an approach that incorporates knowledge graphs (Shakeel & Jain, 2021).

Mayank et al. (2022) proposed a solution which combined the encoded news content with a Knowledge Graph (KG). Its architecture included three parts:

- News encoder (2-layer stacked BiLSTM): applies contextual encoding on the title
- Entity encoder: finds named entities in title and encodes them using KG
- Classification layers (multi-layer perceptron): use the other two components to perform classification

The entity encoder represented the hybrid part with its sub modules: Named Entity Recognition (RoBERTa), Named Entity Disambiguation (connects the identified entity with the most similar one from Wikidata Knowledge Graph), Knowledge Graph embedding (applies encoding ComplEx algorithm on KG to represent it in complex space) and Entity encoder aggregation layer (aggregates the entities' representation). The model which used only news titles was evaluated on two different datasets and achieved an F1-score of 88%, respectively 78% (Mayank, Sharma, & Sharma, 2022).

Starting with 2017, an automatic tool, ClaimBuster, is available to people, being an example of how good fit are machine learning techniques together with natural language processing and knowledge databases. Its strong point is that it is powerful enough to analyze in real time social contexts, speeches, and interviews. After finding a set of facts, this system compares them with already verified ones from a knowledge repository, establishes a label and delivers the result (Hassan N. , Arslan, Li, & Tremayne, 2017).

In these types of models, the use of metadata can improve the results in establishing the degree of veracity. The most frequent approaches include using hyperlinks, associated metadata about context or information about the author (Chu, Gianvecchio, Jajodia, & Wang, 2010).

3 Method

This thesis explores and compares different solutions for inauthentic content detection. Our approach is divided into three main stages: pre-processing data, extracting features and classification of the input. In the next sections, we will present what different methods we tested for each one of the steps.

3.1 Dataset

We focus on applying supervised learning techniques. For that, we use an annotated body of text, LIAR Dataset (Wang, 2017). It contains short English statements extracted from the PolitiFact.com website, which are labeled with a certain level of trust based on a detailed explanation. Having public and accessible data, any necessary information is one click away. Moreover, the dataset can be enlarged at any time by collecting more data from this fact-checking site, which can only benefit our analysis.

LIAR contains a total of 12.800 entries, a size large enough to support complex models. It has an extensive number of classes, which allows the analysis of a more detailed classification. Each one of the labels claims if the statement is accurate or not, if there is any data missing and eventually the ridiculousness of the claim (Holan, 2018).

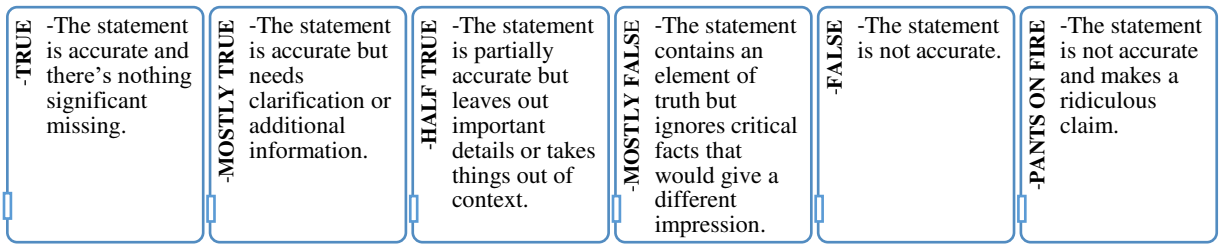

Figure 1 Explanations of labels

At the same time, the large number of classes raises the difficulty of the problem and automatically the performances obtained are lower, but closer to reality. The methods that are analyzed will be tested on 6-labels classification, but also on 2-labels classification. In the latest case, we group the classes "true", "mostly-true" and "half-true" as a unique class "TRUE", while the others, "mostly-false", "false", "pants-fire" as "FAKE".

The annotation of the information is done manually, by specialists with knowledge in the field. More exactly, one reporter researches a fact-check, suggests a rating, and writes a motivation with strong arguments that sustains its label. After that, the reporter and an assigned editor review the report, its authenticity, its weak and strong points, and fix eventually leak of details or contradictions. When they agree on the rating, two additional editors are included in the process. In order to choose the final label, the following questions must be answered by the three editors and the reporter (Holan, 2018):

- "Is the statement literally true?"
- "Is there another way to read the statement? Is the statement open to interpretation?"
- "Did the speaker provide evidence? Did the speaker prove the statement to be true?"
- "How have we handled similar statements in the past? What is PolitiFact's jurisprudence?"

Finally, the three editors vote on the rating, the majority is always wining. After some more edits, the final report is published on the website.

Another aspect that represents an interesting point and motivates the choice of this corpus is the fact that besides the actual statements, LIAR contains some information about the context. In some approaches, this additional data may be used as extra features to improve the performance of the solution and even to explain the proposed label. The supplementary columns that may be used in future experiments are as follows:

- Subject (the subject of the statement)
- Speaker (source of statement)
- Speaker job title (source job title)
- State info (state of source)
- Party affiliation
- Barely true counts (the number of barely-true statements made by that source)
- False counts (the number of false statements made by that source)
- Half true counts (the number of half-true statements made by that source)
- Mostly true counts (the number of mostly-true statements made by that source)
- Pants on fire counts (the number of pants-fire statements made by that source)
- Context (location where the statement was made)

Before building an actual model, we did some dataset analysis to better understand its structure and distribution, and to see how diverse and biased it is. The corpus is already split in the usual parts: training, validation, and testing. As a distribution of data in all three parts, it is relatively balanced between 5 classes: "barely-true", "false", "half-true", "mostly-true" and "true", the only exception being the "pants-fire" class which has significantly fewer examples.

Table 2 Label distribution for multi-label classification

	Training	**Validation**	**Testing**
True	1.676	169	208
Mostly-true	1.962	251	241
Half-true	2.114	248	265
Barely-true	1.654	237	212
False	1.995	263	249
Pants-fire	839	116	92

Table 3 Label distribution for binary classification

	Training	Validation	Testing
True (true, mostly-true, half-true)	5.752	714	668
Fake (pants-fire, false, barely-true)	4.488	553	616

Regarding the extra metadata, we evaluated the set of possible values for each column and observed, in most cases, a high cardinality in the training data and a slightly lower number in the validation corpus. Also, for some columns, the validation partition includes just a part of the set of possible values of the same column from the training slide, plus new unseen values. Therefore, in the validation step, the model encounters information which does not know how to interpret, aspect which hinders the learning process.

From the point of view of the frequency of each value of each category, we draw some conclusions. Firstly, the main persons cited in the dataset are politicians (e.g.: republicans, democrats, independents, activists), their statements representing more than half of the dataset. Secondly, from the point of view of the diversity of topics covered, it emerged that the most common statements fall into the categories: “economy”, “health-care”, “taxes”, “federal-budget”, “education”, “jobs”, while the least common in fields like “autism”, “death penalty”, “food”, “homeless”, “fires”. As expected, the dataset focuses on serious topics, treated in today's society and not on artificial or superficial subjects.

Table 4 Cardinality of metadata columns

Column	Partition	Number of unique values per column
Speaker	Training	2.911
	Validation	661
	Common	446
Context	Training	3.874
	Validation	729
	Common	364
Job	Training	1.108
	Validation	285
	Common	208
Party	Training	23
	Validation	15
	Common	15
Subject	Training	143
	Validation	132
	Common	132
State	Training	69
	Validation	46
	Common	46

This dataset has gain popularity over time, being used in different scientific reports and solutions. Our purpose is to determine the falsity of a news depending on its content, no other metadata. The reason for this decision is that, in most real-life cases, additional information will not be available for new articles. Therefore, our point of comparison is represented by papers that adopt the same approach.

We made a detailed selection of the highest performances we found for different solutions that were used over time. In the original dataset paper, several tested methods and their associated results are presented for 6-labels classification. The scientific article aimed to analyze what models and techniques of natural language processing fit on this corpus. Two important directions were highlighted, namely: text analysis itself and text analysis along with other available information (Wang, 2017). For the first case which interests us, the best results on the testing corpus were obtained for a convolutional network (27%), respectively for a simpler algorithm, Support Vector Machines (25.50%).

The group of Alhindi presented similar solutions but with lower performance metrics (Alhindi, Petridis, & Muresan, 2018). Their innovation was in models that integrate the justification of the label and improve results, but we don't compare to that direction. Brasoveanu and Andonie obtained similar results for classic designs and recurrent neural networks, while their attention-based solutions reached accuracy between 40-50%. Unfortunately, they did not give enough details about their complex implementations to allow us to directly compare with them – we marked this in Table 5 through "?" symbol. Moreover, they proved that introducing more attributes and relations can lead to an accuracy of almost 65% (Brasoveanu & Andonie, 2019).

Various features are tested over time, from surface level indices to n-grams or word embeddings. They are combined either with simple ML architectures or more complex solutions, but overall, Word2Vec led to the highest accuracy for the test partition.

Table 5 State of the art results for multi-classification

Paper	Features	Model	Validation (%)	Testing (%)
(Wang, 2017)	Surface-level linguistic patterns extracted with LIBSHORTTEXT toolkit (binary feature/ word count/ term frequency/ TF-IDF)	SVM	25.80	25.50
	Word embeddings (Word2vec pretrained embeddings from Google News)	BiLSTM	22.30	23.30
		CNN	26.00	27.00
(Alhindi, Petridis, & Muresan, 2018)	Unigram features	LR	23.00	25.00

(Brasoveanu & Andonie, 2019)	Word embeddings (Word2vec, GloVe, FastText)	Multinomial Naïve Bayes	-	23.00
		SGDClassifier	-	22.90
		Random Forest	-	24.90
		Decision Tree	-	22.60
		Basic LSTM	-	22.50
		BiLSTM Attention?	-	*40.80*
		GRU Attention?	-	*46.00*
		CapsNet?	-	55.50

There are multiple papers that address this dataset from a binary point of view. Most of them group three out of six classes as true and the rest as fake. From Table 6, it can be observed that classical ML algorithms and recurrent neural networks achieve similar results in all papers, between 50-60%. The group of Aslam proposed a solution that combines BiLSTM with GRU and obtained a very good score (89.90%). In 2002, Yazdi et al. described a system which selects the most relevant and different features to be used through K-means. Afterwards, simple algorithms are used to achieve more than 90% accuracy. Unfortunately, these last two papers do not provide enough information about how they obtained the two classes from the six original ones, how they split and grouped the classes, so their results are hard to be reproduced or even directly compared with - aspect marked with the same "?" symbol in Table 6.

For the binary case, there are more directions both in terms of classifiers and their associated features than for the first case. From the table below, we can see that lexical and sentimental traits did not bring enough information for the models to learn differences between classes. While n-grams and static word embeddings had similar results, contextualized embeddings set the highest performance of 62%.

Table 6 State of the art results for binary-classification

Paper	**Features**	**Model**	**Validation (%)**	**Testing (%)**
(Alhindi, Petridis, & Muresan, 2018)	Unigram features	LR	58.00	61.00
		SVM (linear)	56.00	59.00
	Word embeddings (GloVe)	BiLSTM	59.00	60.00
(Khan, Khondaker,	Lexical (word count, average word length,	Decision Tree	-	51.00

Afroz, Uddin, & Iqbal, 2021)	count of numbers, count of parts of speech, count of exclamation marks) & Sentiment (positive/negative polarity)	AdaBoost	-	56.00
	n-grams (TF-IDF of word-based uni-gram and bi-gram)	Naïve Bayes	-	60.00
	Lexical categories Empath (Empath tool - e.g.: violence, crime, war)	k-NN	-	54.00
	Word embeddings (GloVe)	CNN	-	58.00
		LSTM	-	54.00
		C-LSTM	-	54.00
		HAN	-	57.00
		Conv-HAN	-	59.00
	Word embeddings (RoBERTa)	Feed-Forward	-	62.00
(Aslam, Khan, Alotaibi, Aldaej, & Aldubaikil, 2021)	Word embeddings (FastText "cc.en.300.vec" pretrained vector)	*BiLSTM – GRU?*	-	*89.90*
(Yazdi, et al., 2020)	N-grams (average of 10 executions of feature selections – uni-grams)	*K-means + SVM?*	-	*94.19*
		K-means + Decision Tree?	-	*92.58*
		K-means + Naïve Bayes?	-	*91.64*

3.2 Text Pre-processing

As mentioned before, in this document, we analyze different methods of fake news detection from a machine learning point of view. At the base of this process is the way the machine interprets the text, meaning how words are converted to a numerical format. In general, in order to have a good representation for the feature vector, certain pre-processing operations are required. Their combination is debatable, depending on the problem itself and the available data as in some cases inappropriate operation may lead to loss of information. The processes that we applied on our input in our experiments are among the most frequent and are summarized in Figure 2.

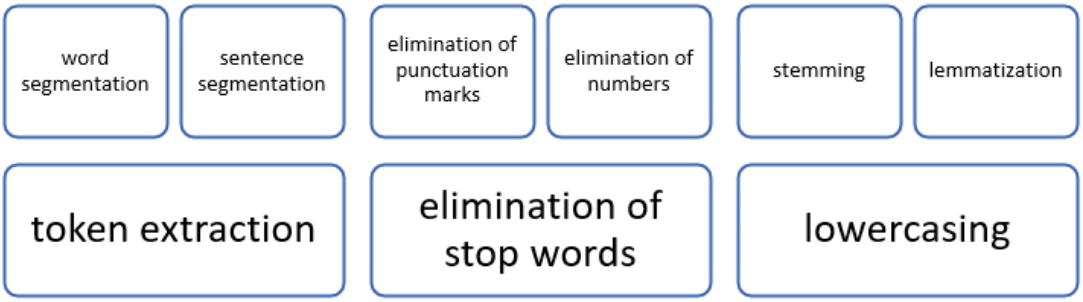

Figure 2 Pre-processing operations applied on LIAR Dataset

As our dataset contains short statements, the input need segmentation only at sentence and word level. Also, we apply tokenization which supposes the extraction of units that contain individual information, such as: “San Francisco”, “school”, “girl”. They can be represented by a single word or by several. We cut out all the words without informational gain, that includes the most used keywords in a language. As our corpus is written in English, tokens as “and”, “I”, “so”, “of”, “in”, “a”, “an” are removed. Our hypothesis is that very common words will not change the degree of truthfulness of the whole statement. Their elimination is common in natural language processing problems which do not involve the analysis of feelings as it is our case.

After that, we want to focus only on alpha characters, so we delete any other noise like numbers or punctuation marks. Likewise, we afford to do that despite the same limitation for emotions, subjectivism, sentimental involvement which also applies for these elements. In plus, some punctuation marks may be confusing as they are used in different situations. For instance, the dot has multiple usages: at the end of the sentence, in numbers (3.14), in other punctuation marks (…) or in acronyms (U.S.A.). Another example would be the hyphen which may be a sign for a dialogue replica (- Hello!), a union of two words (pre-processing) or an explanation (Barack Obama – the president of U.S.A. – voted against….). Discarding these items will help the classifiers in the process of learning.

In order to have a uniform input, the next step of normalization is lowercasing. Finally, stemming and lemmatization are used to reduce the number of different words in the text and to unify derived or articulated words. By applying them, the words are reduced to their basic form (root). We experiment with both procedures to find which one achieves better results. The difference between them is that lemmatization draws a valid root, reducing

the word to its dictionary form (e.g.: has → be). Unfortunately, it involves a longer process because each word must be POS-labelled at first. In contrast, stemming does not always extract a valid root identical to the morphological form of the word (e.g.: believing → believ). In plus, sometimes, more letters than would be desirable are removed but this form is usually sufficient. We applied the Porter's stemmer, which removes common suffixes and prefixes (e.g.: "-ing", "-ed", "-ness", "-less", "un-").

To exemplify the pre-processing steps, we take a sentence from the test partition, and we go through all the stages:

"Building a wall on the U.S.-Mexico border will take literally years."

↓

"build wall usmexico border take liter year".

3.3 Features

Once the text is in a normalized form, different features can be extracted. The process of extracting characteristics from the news is an incremental one. It aims to choose various information and combine them in order to best describe the original text. In the process of experimenting with different variants to find the solution that offers the best accuracy for the problem, four directions were tested: bag of n-grams with TF-IDF, static word embeddings (Word2Vec, GloVe), contextualized word embeddings (BERT, DistilBERT) and stylometric features.

3.3.1 Bag of n-grams

The simplest method of converting text into numerical format is the bag of n-grams, a way that represents each sequence of n tokens as an individual unit, independent and equally significant with the others. This approach is the most natural way to start the analysis, as it is well known that fake articles tend to use different words than real ones, generally words with a powerful emotional impact. Thus, we try to find a correlation between the importance of words from a text and its degree of confidence. For this paper, we choose to analyze the statements as bags of 1-grams and 2-grams. In order to remove any useless noise from the text, we applied the operations described in the previous section: numbers removal, punctuation marks removal, lowercasing and stop words removal. The structural features that were extracted afterwards were at words, lemmas, and stems level.

This approach involved creating a vocabulary with sequences of n tokens extracted from all the documents. Each document was represented as a feature vector that has a frequency associated with each item from the vocabulary. The frequency was computed using Term Frequency - Inverse Document Frequency (TF-IDF). As the name suggests, the TF-IDF of an element is calculated as the product of its number of occurrences in the document (term frequency), and how common the term is in all documents (inverse document frequency). The advantage of this method is that it limits the importance of very frequent syntagms as

they have a lower informative content than those that appear in a smaller portion of the document set.

Bag of n-grams comes with several disadvantages such as: vocabulary terms are isolated from the context, the meaning and position of the grams become non-existent, and the relationships between words and sentences are ignored. For TF-IDF, the detriment is that produces rare and large vectors.

3.3.2 Static word embeddings

The first hypothesis we started with was that the frequency of tokens is more important than the relationships between words, sentences, or context. This is a rather restrictive approach, so for the second method of extracting features we take into account the similarity between words and use word embeddings. So, if in the previous section, the words were represented as rare and long vectors, in this subchapter they are represented as dense vectors of small dimensions. Over time, dense vectors have been shown to work better than rare ones in most NLP tasks, including fake news detection. Some of the reasons include: an easier and more efficient training as the number of weights is smaller, a smaller space of parameters that helps to generalize and avoid overfitting and better management of synonymy.

A Word2Vec

In this regard, we use Word2Vec, a technique based on a 2-layer neural feed-forward network described in the paper "Efficient Estimation of Word Representations in Vector Space" (Mikolov, Chen, Corrado, & Dean, 2013). The model has two architectural variants: continuous-bag-of-words (predicts the current word based on a neighboring word window) and skip-gram (uses the current word to predict the neighboring word window). The number of words considered in the window has increased over time and has led to better results.

Word2Vec is based on training a classifier on a binary prediction task. The goal is not the prediction itself, but the learned weights that will represent word embeddings. The obtained representations are positioned in the vector space so that the words that share common contexts in the corpus are in the immediate vicinity of each other. Word2Vec offers speed, efficiency of training and online availability along with pre-trained weights. A weak point for this type of word embedding is that the representation of features is a static process, where any word in the vocabulary has a fixed representation associated with it, regardless of its meaning.

For pre-processing and normalizing the text that precursors the application of Word2Vec, we applied: sentence segmentation, lowercasing, stop words removal and token extraction. Further, we decided to train on LIAR dataset our own model using Gemsim library. We experimented with both skip-gram and continuous-bag-of-words cases. The parameters that we used for the Word2Vec model are:

- min_count = 1 (we kept all words from the corpus)

- windows = 5 (maximum distance between the current and predicted word within a sentence)
- vector_size = 100 (word vectors length)

Final text embeddings were also limited to a dimension of 100, while null items were used to pad shorter statements.

B GloVe

Another method of obtaining static embeddings for words that we apply on our LIAR dataset is GloVe. This technique is an unsupervised learning algorithm based on the co-occurrence probability ratio between words. It was discovered by researchers at Stanford University who published the paper "GloVe: Global Vectors for Word Representation" (Pennington, Socher, & Manning, 2014). GloVe is already introduced in various tasks which implies finding relationships between words like synonyms or antonyms. Overall, its usage is more frequent than that of Word2Vec, as it led to better results. Still, this method has the same shortage, meaningly it generates a unique vector for every word, no matter how many different meanings it may have.

GloVe is based on training a classifier on aggregated word pair statistics from a corpus. To build the statistics and therefore the co-occurrence matrix, just one pass through the corpus is needed. Also, in this process, only the non-zero entries of the matrix are used. The interesting aspect of this method is that it is focused on the relationships between words, not just their unique occurrence. Every item from the matrix shows how often a particular pair of words occurred together in the given dataset. That means that the authors' main intuition when they built the model was that the co-occurrence probabilities may encode some form of meaning.

In our case, for pre-processing and normalizing the text that precursors the application of GloVe, we applied the same actions as in the previous case: sentence segmentation, lowercasing, stop words removal and token extraction. Further, we decided to use a pre-trained model on the Wikipedia 2014 and Gigaword5 datasets which contains 6B tokens for uncased words. The dimension of the final embeddings was also limited at 100. Texts representations with lower dimensions were completed with 0 elements.

3.3.3 Contextualized word embeddings

Contextualized embeddings are more powerful than static ones as they take into account semantics and associate a certain representation to a word depending on its meaning in the context. This way, same word may have different formats. We obtained our dynamic word embeddings using transformers-based techniques. Transformers have two mechanisms: an encoder of the entered text and a decoder that produces the prediction. The encoder is considered bidirectional as it reads all words at once. Also, these systems have an attention mechanism that learns the contextual relationships between words in a text, allowing a model to consider each word from the input before making a decision. Over time it was proved that a language model trained on this type of features can understand the language context much better.

A Bidirectional Encoder Representations from Transformers (BERT)

For the third series of experiments, we adopted the alternative of contextualized words representations through a numeric format obtained with BERT. It is one of the newest architectures used in NLP that is based on the transformer theory. As the goal of BERT is to encode words and generate embeddings, only the encoding mechanism from the traditional architecture of transformer is needed. BERT was created precisely to generate deep two-way representations of unlabeled text by jointly conditioning both contexts (right & left) in all layers. It was trained on two supervised tasks, created from the Wikipedia corpus in an unsupervised way: predicting previously random masked words and determining the sequence of two sentences in a text passage. Thus, the result is a pre-trained encoder that incorporates words while considering the context.

We used two smaller versions of BERT, as the original one included 110M parameters and we did not have enough computational resources to fine-tune such a large model. Also, we decided to not build a new vocabulary based on the dataset but used the pre-trained models "bert_en_uncased_L-4_H-512_A-8" (BERT-Small) and "bert_en_uncased_L-8_H-512_A-8" (BERT-Medium) for BertTokenizer, along with the corresponding numerical mapping. These models had only 32M, respectively 49M parameters and were fine-tuned for our specific dataset for 20 epochs in total. We decided to follow the same pattern and set the maximum length of the embeddings at 100 and pad with null elements.

B DistilBERT

Another solution for dynamic representations that we opted for is DistilBERT, still a smaller and cheaper version of BERT, but quite bigger than BERT-Small or BERT-Medium. Its purpose is to increase computation efficiency, so the authors focused on reducing the number of layers. This alternative general pre-trained solution reduces the size of BERT by 40%, having 66M parameters and increases the speed with 60% while keeping 97% of its capabilities. Overall DistilBERT keeps the general architecture of its base model but at a smaller scale. The number of layers is diminished by a factor of 2 while the pooler and token-type embeddings are removed.

We had the same approach, meaning that we used the pre-trained version "distilbert-base-uncased" for DistilBertTokenizerFast from HuggingFace, along with the corresponding numerical mapping. The model was fine-tuned afterwards for our case for 20 epochs. Numerical representations of the input were kept to maximum 100, being padded with 0 if they were shorter.

3.3.4 Stylometric features

The last approach of extracting features does not take into account the frequency of words, their importance or their semantic meaning, but the author's writing style that emerges from the text. In general, a lot of fake content generators use language strategically to avoid being discovered. However, not all linguistic aspects are easy to control in writing, such as the frequency of adverbs, pronoun patterns, the frequency of conjunctions, or the use of negative words. For example, it has been concluded that people tend to use more first-person

pronouns in false news. Also, longer sentences are used involuntarily when trying to convince the reader of a generally false aspect. Therefore, an attempt was made to identify some indicators of the writing style and of the format, in order to see if they could represent a valid direction of study and a possible solution for this task.

The starting point for this approach is the article "ReaderBench: A Multi-lingual Framework for AnalyzingText Complexity" (Dascălu, et al., 2017), a work that presents different indices of complexity of the text. Thus, the aim is to find a link between the way a text is written and its degree of truth. If more complex texts contain more information and, inherently, more diverse concepts, the assumption used is that true texts tend to have high levels of complexity. In this regard, we use a series of features to train a new series of models and analyze the outcomes.

For calculating complexity indices, we choose the ReaderBench tool, a complex software product that focuses on in-depth analysis of texts from several points of view. Specifically, it is built as an easy-to-use framework that integrates advanced natural language extraction, processing, and analysis techniques. (Dascălu, et al., 2022). The purpose of the ReaderBench project is much broader, being an educational system for students and tutors that combines individual learning methods with computer-supported collaborative learning (CSCL). In addition, this system is a high-performance system with a high degree of reliability, being already used in several projects on the market. Its adoption on such a large scale indicates the high quality and usefulness of these indices and encourage us to analyze its results in false content detection.

The range of indices is quite wide, bringing together metrics related to various areas of study such as readability, semantics, morphology, or discourse structure. Their number exceeds 200, and if we refer to various sections such as word, sentence, paragraph or entire document, their number increases significantly. The system classifies these indices into five categories according to their complexity and scope (Dascălu, et al., 2017):

1. **Surface Indices.** They are determined only by the form of the text and are mostly lengths, frequencies, or entropies of sentences / words. They are the easiest to calculate and use, but at the same time they provide information only at the structure level, not at the level of morphology or discourse. We consider these clues useful because they can give us an overview of the diversity of present concepts and the level of information. All these data can be indicators of veracity.
2. **Word Complexity Indices.** This category goes beyond the superficial analysis of words and focuses on its internal structure. Specifically, the number of syllables, the differences from the lemma or the root, the number of possible meanings derived from WordNet and the specificity deduced from its depth in the lexicalized ontology are analyzed. The larger are these metrics, the more complex the word. Prefixes and suffixes increase the difficulty of using a word, while multiple meanings make it difficult to deduce the ideas of a text. Also, a high presence of named entities gives both veracity to the text and heterogeneity, requiring more cognitive resources to understand the message.

3. **Syntactic and Morphologic Indices.** This set of metrics analyzes the text from a broader perspective, namely at the sentence level. Thus, a complex and truthful text is considered to have several dependencies between words and uses several main parts of speech as nouns or verbs. These elements primarily bring textual structure, a structure that gains the reader's trust, but also gives a serious tone to the content. At the same time, the distribution of pronouns can be an indication of the degree of truth of the text.
4. **Semantic Cohesion Indices.** Cohesion can be defined by the various ways in which the components of a text are interrelated. Types of links can be grammatical, lexical, semantic, metric, or alliterative. They play a very important role in the process of understanding a text, being basic indices in the ReaderBench structure. This tool uses various semantic models (semantic distances in WordNets, LSA, LDA, Word2Vec), lexical chains, and co-reference chains to calculate these strong indices.
5. **Discourse Structure Indices.** The indices from this category follow the evolution of the points of view in a text by analyzing the specific connectives of the speech and the polyphonic model. In our case, the input is represented by entries consisting of several statements. For this length, these indices are not very significant, but they will be used at least for some initial models until a rigorous filtering of metrics.

Table 7 Text complexity indices (D = Document, P = Paragraph, S = Sentence, W = word)

Category	Description	Notation	Multiple classes available	Level			
				D	P	S	W
SURFACE	No. words	Wd	No	X	X	X	
	No. unique words	UnqWd	No	X	X	X	
	No. commas	Comma	No	X	X	X	
	No. punctuation marks	Point	No	X	X	X	
	No. sentences	Sent	No	X	X		
	Word entropy	WdEntr	No	X	X	X	
WORD COMPLEXITY	No. characters in a word	Chars	No				X
	Distance in number of characters between inflicted form and its lemma	LemmaDiff	No				X
	No. occurrences of the same lemma	Repetitions	No	X	X	X	
	No. of named entities	NmdEnt	No	X	X	X	
	Maximum depth in the hypernym tree from root to word sense	MaxDepth HypTree	No				X
	Average depth in the hypernym tree from root to word sense	AvgDepth HypTree	No				X

	Paths to the root of the hypernym tree based on all word sense	PathsHyp Tree	No				X
	Word polysemy count	Polysemy	No				X
	No. syllables in word	Syllabus	No				X
	Age of Acquisition scores	AoA	No	X	X	X	X
	Age of Exposure scores	AoE	No	X	X	X	X
	Word valence	Valence	No	X	X	X	X
SYNTACTIC	No. dependencies of a certain type	Dep	Yes, for all dependencies	X	X	X	
	Depth of the parsing tree	ParseDepth	No				X
MORPHOLOGIC	No. words with specific part of speech	POS	Yes, for all PoS	X	X	X	
	No. unique words with specific part of speech	UnqPOS	Yes, for all PoS	X	X	X	
	Nr. specific types of pronouns	Pron	Yes, for all pronoun types	X	X	X	
SEMANTIC COHESION	Cohesion between adjacent sentences	AdjSentCoh	No	X	X		
	Cohesion between adjacent paragraphs	AdjParCoh	No	X			
	Cohesion between paragraphs	InterParCoh	No	X			
	Cohesion between sentences inside a paragraph	IntraParCoh	No	X	X		
	Cohesion between first and last text element	StartEnd Coh	No	X	X		
	Cohesion between first and middle text element	StartMiddle Coh	No	X	X		
	Cohesion between middle text element and last one	MiddleEnd Coh	No	X	X		
	Transition cohesion between adjacent sentence and paragraph	TransCoh	No	X			
DISCOURSE STRUCTURE	No. of connectors on predefined categories	Connector	No	X	X	X	

All these indices were calculated for both training, validation, and testing. An earlier stage of pre-processing was omitted to avoid the possible loss of information. In addition, the system performs a precursor step in word processing before calculating each metric, depending on its type and nature.

3.4 Classical Machine Learning Models

A Decision Trees

Decision Tree classifier is one of the simplest supervised learning algorithms. It is a classic tree structure where the root node has no input edges, while all other nodes have one. Thus, if a node has outputs edges it is called an internal node, otherwise it is called a leaf. Each internal node represents a condition on a feature and divides the instance space into two or more subspaces depending on a particular discrete function of the input attribute values. Leaf nodes are classifier labels. Alternatively, the leaf may have a probability vector that indicates the probability of the target attribute having a certain value. For each input, the path to a terminal node that determines the class prediction is found. These trees can be easily transformed into if-then-else rules, simply by joining the tests along each path to form the condition part and taking the prediction of the leaf as a class value (Rokach & Maimon, 2005). To build our classifier, we experimented with different values for the parameters of the model:

- criterion: [Gini impurity, entropy]
- max_depth: [5, 10, 15, 20, 25]
- splitter: [best, random]
- random_state: 42

The Decision Tree model which offered the highest accuracy was the one which had a maximum depth of 20, the criterion “best” used to choose the split at each node, entropy as measurement of the quality of a split and a factor of randomness of 42 for the estimator.

Some disadvantages of this algorithm are the following: even if DT is flexible, it cannot determine the significance of the characteristics, it is unstable and prone to abnormal values (outliers) and it can neglect some key values in the training data, causing low accuracy. In addition, the classification process adopted is sequential, not simultaneous, with a clear dependence between levels, which may affect the results.

B Naïve Bayes

The Naïve Bayes classifier has a very good complexity-performance ratio, being one of the most popular basic methods for classifying text with term frequencies as input. This is a statistical model, belonging to the family of probabilistic methods, which tries to determine the probability that a certain document belongs to a certain class, considering the features. Finally, a document is associated with the class that gets the highest score. The main problem with this algorithm is that of “zero-frequency” which involves assigning a zero score when evaluating a new feature that was not encountered in training. In general, this limitation is

removed by applying soothing methods, assuming that if a trait-label pair did not appear in the training set, it does not mean that it is impossible for it to appear later.

Another limitation that is worth mentioning is the hypothesis from which this algorithm starts, namely the fact that the features are independent. Thus, if we consider two adjacent words as two characteristics, then their probabilities will be multiplied as if they were independent, although this is not always the case.

Our final model had the following parameters:

- alpha = 1 (soothing parameter)
- fit_prior = True (to learn class prior probabilities)

C Support Vector Machines

Support Vector Machines is a linear classifier that starts from the premise that the set of classes is linearly separable, trying to find a plan that separates them. As a working principle, it assigns to each example a confidence score for each class. SVM handles abnormal values and a large number of features very well, which is an important advantage. The limitations imposed by this algorithm appear if there is no clear separation between classes.

The process of finding the best SVM algorithm involved a long series of experiments with the parameters of the model. More precisely, we trained classifiers with each of the following values and evaluated them on the validation set in order to find the best one:

- kernel: [linear, polynomial, sigmoid, radial basis function]
- C: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
- Degree: [2, 3, 4]
- Gamma: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]

After a series of preliminary tests, we concluded that for the LIAR dataset, linear and quadratic kernel gave the highest accuracy. So, for the final model, we chose two SVM models:

- linear kernel, regularization parameter equal to 0.7
- quadratic kernel, regularization equal to 1 and kernel coefficient equal to 1

3.5 Neural Networks

A Long Short - Term Memory

Classifiers using LSTM divide the issue of context management into two sub-issues: deleting information that is no longer useful and adding to the new context information that could potentially be used in subsequent decisions. This network manages to solve both tasks by using gates that control the flow of data entering and leaving each unit at the layer level. These gates are implemented with the help of additional weights that operate sequentially on the input, the previously hidden layer, and the previous context layer.

The LSTM architecture is distinguished by three gates, where each gate has the same design that involves a feed-forward layer with sigmoid activation, followed by a point multiplication with the previous layer (Li & Wu, 2015). One of the gates is the forget gate, which erase information from context that is no longer useful. In other words, decide what information to forget from the previous step. It calculates a weighted sum between the previously hidden state layer and the current entry, which it passes through a sigmoid function. The result is then multiplied by the context vector. The second gate is to add information (update gate) and get the new context vector. In short, decide what information should be kept from the input. The final output gate (result gate) decides what information is needed for the current hidden state.

We experimented with different values for the dimension of the layers: 100, 300 and 1000. Our final architecture for which we obtained the highest accuracy is a relatively simple one that includes:

- an input Embedding layer with input_dim equal to the size of the vocabulary and output_dim = 1000
- an LSTM layer of size 1000
- a fully connected hidden layer with 1000 neurons with ReLu activation function
- a DropOut layer that randomly sets 0.5 input units to 0 at each update during training
- a fully connected output layer with 2/6 neurons with softmax / sigmoid activation function

For optimizations, we used Adam, and a learning step set to 0.0001. The model was trained for a period of 10 epochs, after which we stopped because it was not improving and started overfitting.

B Bidirectional Long Short-Term Memory

Another alternative classification model for LSTM that has been used is Bidirectional Long Short-Term Memory. Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification issues. This architecture drives two instead of one LSTM on the input sequence. The first LSTM is instructed on the input sequence while the second is on an inverted copy of it. This technique can provide additional context to the network and can lead to a faster and more complete learning. The architecture of the model is similar with the previous one:

- an input Embedding layer with input_dim equal to the size of the vocabulary and output_dim = 1000
- an BiLSTM layer of size 1000
- a fully connected hidden layer with 1000 neurons with ReLu activation function
- a fully connected output layer with 2/6 neurons with softmax / sigmoid activation function

The process of training was identical to that of LSTM, meaning that the same Adam optimizer was used along with a 0.0001 learning rate. The model was trained for a period of 20 epochs.

Based on the research we made for the state of the art, we found out that BiLSTMs are often used with attention mechanism to improve results. Therefore, we tried to also approach this direction. In general, attention can be described as a vector of importance weights which resumes how strongly a word is correlated with another. In our case, we experimented with an additional multi-head self-attention layer. This mechanism compares every sequence from the sentence to all the others, recalibrating the embedding in order to take into account contextual relevance. In this way, we can learn the correlation between a part of the sentence and all that precedes or succeeds it. Multi-head runs through the attention mechanisms multiple times in parallel. The outputs are then concatenated and transformed to the desired dimension. In this case, the model was further parametrized as:

- an input Embedding layer with input_dim equal to the size of the vocabulary and output_dim = 1000
- an BiLSTM layer of size 1000 with recurrent dropout set of 0.2 and dropout of 0.25
- a GlobalMaxPool1D layer
- a Dropout layer with factor of 0.2
- a MultiHead Self-Attention layer with head size of 128 and number of heads of 12
- a fully connected output layer with 2/6 neurons with softmax / sigmoid activation function

We used the same Adam optimizer along with a 0.0001 learning rate. The model was trained for a period of 20 epochs.

C Feed-forward network

Feed-forward neural networks are used to learn more complex non-linear models. They consist of an input layer, an output layer, and one or more hidden layers. The loss function, however, is known only for the output layer. As a result, the error in the last layer propagates back to the neurons in the previous layers to update their weights in the same way. The algorithm is a well-known backpropagation, in which it is considered that the neuron j is responsible for a part of the error associated with each of the neurons in the next layer with which it is connected. Our simple network involved:

- an input Embedding layer with input_dim equal to the size of the vocabulary and output_dim = 1000
- a DropOut layer that randomly sets 0.1 input units to 0 at each refresh during training
- a fully connected output layer with 2/6 neurons with softmax / sigmoid as activation function

In addition, an Adam optimizer with a 3e-5 learning rate was used. Due to computational limitations, training has only been done for 20 epochs.

4 Results

For each solution, we build a multi-label classifier with the original labels from the dataset and a binary one that treats a simpler case. Therefore, we follow the same pattern as the papers from the state of the art and group the classes “true”, “mostly-true” and “half-true” as “TRUE”, respectively “false”, “pants-fire”, “barely-true” as “FAKE”. We evaluate each model on both validation and testing partition, to see if there are major differences. Also, we used the validation slice to tune the hyper-parameters of each model, when was the case. The results obtained in both cases are presented in the following tables, being grouped according to the type of text features used.

4.1 Classification based on TF-IDF

For the first series of experiments, we opt for classical machine learning algorithms: Decision Tree, Naive Bayes, and Support Vector Machines, along with the features extracted by TF-IDF. In order to find the most relevant series of characteristics, we try this frequency-based approach on different inputs: words, stems and lemmas. Also, we look either at unique entities or two entities at the time, respectively unigrams, and bigrams. All the results from our tests are summarized in Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, and Table 15.

After analyzing the case of multi-class classification, we draw the following conclusions. In most cases, the best results were obtained with the Support Vector Machines model, either with linear or quadratic kernel, while the weakest with the Decision Tree algorithm. Only for the case of unigrams and bigrams of words concatenated with unigrams of lemmas, the Naive Bayes model got a lower accuracy on the test partition. Using a linear kernel for the SVM algorithm led to better results than those of the previous two models in all input cases. Overall, for 6-labels classification, the highest performance was obtained with this type of model which scored an accuracy of 25.97%. The instance of quadratic kernel was not good enough to separate the six classes, beating the linear one only when using unigrams of stems or unigrams and bigrams of words concatenated with unigrams of lemmas.

If we compare these metrics with the state of the art of this dataset, our best frequency-based model, SVM, had a performance approximately equal to those or even higher. It bet both classical solutions (SVM, LR, NB, SGDClasifier, Random Forest, DT) and deep-learning ones (LSTM, BiLSTM), even if they were based on more informative features. The model was better than the “random choice” by about 10%. Moreover, our most accurate Naive Bayes (24.15%) and Decision Tree (23.36%) algorithms exceeded the performance of the same models from previous published papers, but also of the LSTM models. The solutions that declared a higher accuracy (40-50%) used word embeddings, and attention-based architectures, aspect that we will consider in future sections. It is important to mention that they do not give enough details about their implementation, so the execution context may differ, making direct comparison not possible. From an analysis of metrics at each label level, we saw that “pants-fire” class had the lowest values in all models, which was expected

considering that it is the class with the fewest statements in the data set. Simple methods like the Naïve Bayes or Decision Tree algorithms failed to learn characteristic aspects for this class, obtaining all 4 metrics almost less than 10%.

For binary classification, the outcome looked similar, meaning that the highest performance that we managed to accomplish was 62.19% from the same linear SVM model. As in the previous case, DT paired with any type of traits achieved the lowest accuracy of all four models. SVM algorithm continued to dominate the other models through its linear or quadratic approach, boosting performance to at least 60% in all cases. The only exemptions were when we introduce unigrams and bigrams of lemmas, either on their own or concatenated with other features, case when the NB algorithm bet the rest and reached even 61.64%. From the point of view of the kernels applied, the tables show that adding bigrams to the input always boosted up the performance of the models that used the linear one. On the other hand, unigrams with polynomial kernel surpassed the others in most cases.

Our best model for binary classification outperformed all models from the state of the art, either classical ML algorithms such as LR, SVM, DT, AdaBoost, NB, k-NN, or neural network models like LSTM, CNN, C-LSTM, Feed-Forward, HAN, BiLSTM, Conv-HAN. It's interesting to highlight that our simplest solution distinguished between the two classes almost like the one which used semantic-based traits, RoBERTa, with a simple feed-forward network (62.19% vs 62.00%). Also, it bet arbitrary choice by 12%. In this case, the best versions of DT (58.48%) and (61.80%) exceeded all solutions that used simple classifiers with lexical features from the state of the art and obtained comparable values with those based on word embeddings and neural networks.

Regarding the type of features, we analyzed words, stems, and lemmas to see which one can provide the highest informational gain for these models. For each case, we followed an incremental process, which started with looking at simple elements and then adding pairs. Regarding words, adding bigrams increased accuracy for the DT multi-class model and for the NB and linear SVM bi-class models. For stems, improvements remained only for the same binary models, while for lemmas, the NB models slightly improved their performance in both cases. Overall, the models trained with TF-IDF of lemmas had a better performance than those with stems. That's the reason why we continued the series of experiments with concatenating words with lemmas in order to see if we can improve our values. Therefore, we observed that more features can improve some models, at least for the case of multiple classes. For instance, the DT model managed to deduce more complex rules and achieve 23.36% accuracy when using three different types of features. Overall, for the test partition, the highest accuracy was obtained when we kept only words as input of the models, not any abbreviated or simplified form of it.

The metrics from the table show that there is a link between the frequency of the words used and the degree of veracity of the text, but not so strong as to return very good results. Despite having similar or even better performance metrics than those from the state of the art, the models were not accurate enough to perform in real life situations. Overall, for all three algorithms, the adopted pre-processing operations may not have been sufficient to bring the dataset statements to the best standardized form. Also, the extracted features may not have

been sufficiently relevant and easy to interpret by these models. In addition, it is very likely that the results were limited both by the fact that the meaning of the words is ignored and by the fact that the dependencies between them are not captured. In the next subchapter we present the results of some models that aim to remove these limitations to see if the obtained accuracy will be higher than 25.97%, respectively 62.19%.

Table 8 Results for TF-IDF of unigrams of words

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
TF-IDF of unigrams of words	DT	Validation	22.51	22.03	22.51	18.04	57.48	60.20	57.48	52.65
		Testing	22.10	23.60	22.10	17.50	58.48	57.92	58.48	52.84
	NB	Validation	23.60	21.41	23.60	20.27	59.19	61.08	59.19	56.13
		Testing	24.15	22.86	24.15	20.87	60.06	59.85	60.06	56.05
	SVM linear	Validation	24.07	26.28	24.07	22.58	59.97	60.15	59.97	59.21
		Testing	***25.97***	***26.92***	***25.97***	***24.69***	61.56	61.00	61.56	60.74
	SVM quadratic	Validation	25.62	31.82	25.62	23.37	61.99	63.32	61.99	60.29
		Testing	25.02	31.57	25.02	22.95	*61.88*	*61.73*	*61.88*	*59.28*

Table 9 Results for TF-IDF of unigrams & bigrams of words

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
TF-IDF of unigrams & bigrams of words	DT	Validation	21.34	21.91	21.34	18.11	55.61	58.25	55.61	49.11
		Testing	22.73	23.59	22.73	19.20	57.46	56.33	57.46	50.47
	NB	Validation	23.91	23.01	23.91	19.68	58.57	63.24	58.57	52.98
		Testing	23.13	20.35	23.13	18.33	60.93	63.31	60.93	54.52
	SVM linear	Validation	24.84	28.11	24.84	22.78	62.15	63.05	62.15	60.85
		Testing	*24.70*	*31.00*	*24.70*	*22.52*	***62.19***	***61.79***	***62.19***	***60.45***
	SVM quadratic	Validation	25.31	37.92	25.31	21.66	59.89	63.72	59.89	55.61
		Testing	23.28	29.25	23.28	18.92	60.54	61.23	60.54	55.46

Table 10 Results for TF-IDF of unigrams of stems

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
TF-IDF of unigrams of stems	DT	Validation	21.88	21.16	21.88	15.50	57.48	60.14	57.48	52.70
		Testing	21.70	24.11	21.70	16.24	58.17	57.39	58.17	52.41
	NB	Validation	24.30	31.16	24.30	20.94	58.96	60.89	58.96	55.78
		Testing	23.52	20.96	23.52	19.85	60.30	60.10	60.30	56.53
	SVM linear	Validation	24.45	27.58	24.45	23.31	62.15	62.51	62.15	61.40
		Testing	23.99	24.28	23.99	22.93	60.54	59.87	60.54	59.51
	SVM quadratic	Validation	26.87	32.95	25.87	25.11	61.60	62.54	61.60	60.18
		Testing	*24.70*	*30.75*	*24.70*	*23.03*	*61.17*	*60.61*	*61.17*	*59.27*

Table 11 Results for TF-IDF of unigrams & bigrams of stems

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
TF-IDF of unigrams & bigrams of stems	DT	Validation	21.57	20.12	21.57	15.75	56.15	57.74	56.15	51.58
		Testing	21.63	22.23	21.63	16.31	57.06	55.47	57.06	52.02
	NB	Validation	24.22	23.79	24.22	20.00	57.94	62.45	57.94	52.04
		Testing	23.52	20.51	23.52	18.47	61.80	65.54	61.80	55.23
	SVM linear	Validation	24.61	28.23	24.61	22.97	61.99	62.62	61.99	60.94
		Testing	*23.84*	*24.57*	*23.84*	*21.71*	*61.96*	*61.48*	*61.96*	*60.33*
	SVM quadratic	Validation	23.60	32.91	23.60	20.45	59.03	61.69	59.03	55.20
		Testing	22.89	30.43	22.89	19.17	60.69	61.14	60.69	56.13

Table 12 Results for TF-IDF of unigrams of lemmas

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
TF-IDF of unigrams of lemmas	DT	Validation	21.11	20.17	21.11	15.71	57.48	60.37	57.48	52.49
		Testing	22.65	24.14	22.65	17.19	57.70	56.51	57.70	52.31
	NB	Validation	24.38	31.60	24.38	20.96	59.19	61.20	59.19	56.01
		Testing	23.84	22.53	23.84	20.27	60.62	60.57	60.62	56.82
	SVM linear	Validation	23.91	27.53	23.91	22.63	60.59	60.90	60.59	59.72
		Testing	*25.41*	*26.64*	*25.41*	*24.23*	*61.72*	*61.15*	*61.72*	*60.60*
	SVM quadratic	Validation	26.95	33.43	26.95	24.91	60.98	61.92	60.98	59.43
		Testing	25.34	31.94	25.34	23.49	*61.72*	*61.35*	*61.72*	*59.57*

Table 13 Results for TF-IDF of unigrams & bigrams of lemmas

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
TF-IDF of unigrams & bigrams of lemmas	DT	Validation	22.27	23.11	22.27	17.90	56.07	57.95	56.07	51.03
		Testing	22.65	24.24	22.65	17.64	57.70	56.47	57.70	52.67
	NB	Validation	24.53	23.40	24.53	20.22	58.18	62.75	58.18	52.39
		Testing	24.15	22.65	24.15	19.27	*61.64*	*64.50*	*61.64*	*55.51*
	SVM linear	Validation	25.70	29.49	25.70	23.83	63.01	63.95	63.01	61.80
		Testing	*25.26*	*27.72*	*25.26*	*23.33*	61.17	60.58	61.17	59.45
	SVM quadratic	Validation	24.53	35.31	24.53	21.13	59.03	62.33	59.03	54.68
		Testing	23.28	31.14	23.28	19.57	60.62	61.18	60.62	55.80

Table 14 Results for TF-IDF of unigrams & bigrams of words and unigrams lemmas

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
TF-IDF of unigrams & bigrams of words and unigrams of lemmas	DT	Validation	20.95	21.12	20.95	16.14	55.53	57.80	55.53	49.38
		Testing	23.36	25.75	23.36	18.23	57.06	55.54	57.06	50.05
	NB	Validation	24.77	32.43	24.77	21.49	59.03	61.69	59.03	55.20
		Testing	23.28	21.96	23.28	19.77	60.93	61.35	60.93	56.63
	SVM linear	Validation	24.14	24.64	24.14	23.95	59.35	59.36	59.35	58.84
		Testing	23.76	23.51	23.76	23.36	*61.40*	*60.86*	*61.40*	*60.72*
	SVM quadratic	Validation	26.64	30.26	26.64	24.90	61.92	62.61	61.92	60.78
		Testing	*24.94*	*28.91*	*24.94*	*23.53*	60.62	59.94	60.62	58.98

Table 15 Results for TF-IDF of unigrams & bigrams of words and unigrams & bigrams lemmas

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
TF-IDF of unigrams & bigrams of words and unigrams & bigrams of lemmas	DT	Validation	22.27	21.68	22.27	17.58	55.84	57.83	55.84	50.40
		Testing	23.05	22.74	23.05	17.91	57.54	56.26	57.54	52.01
	NB	Validation	25.47	33.35	25.47	22.32	60.20	63.04	60.20	56.76
		Testing	23.68	22.58	23.68	20.22	*61.48*	*62.46*	*61.48*	*56.91*
	SVM linear	Validation	25.47	26.21	25.47	25.32	60.83	60.87	60.83	60.40
		Testing	*24.63*	*24.23*	*24.63*	*24.14*	60.46	59.85	60.46	59.70
	SVM quadratic	Validation	26.01	30.15	26.01	23.76	60.83	62.36	60.83	58.67
		Testing	23.28	27.18	23.28	21.16	60.30	59.76	60.30	57.48

4.2 Classification based on word embeddings

The results of the classification using the LSTM and BiLSTM classifiers, together with features extracted by Word2Vec and GloVe are shown in Table 16. In addition, they are compared with those obtained using a feed-forward neural network with two versions of BERT and DistilBERT. A combination of these two approaches is also tested by training a BiLSTM classifier using characteristics selected through a transformer architecture.

From the point of view of features, we obtained a straightforward order of performances for binary classification. From the table, we concluded that Word2Vec word embeddings combined with neural networks led to the lowest accuracies, between 55.56% and 58.01%. Among these models, the skip gram approach was a better fit with BiLSTM architecture, while continuous bag of words transformed words into a numerical format that was easier to interpret for a simple LSTM classifier. As in general GloVe comes with significant improvements, this was also our case. Despite having more unknown words in our dataset vocabulary, these word embeddings associated with BiLSTM architectures led to performance metrics of approximately 58.50%. One thing to notice is that despite adding the self-attention layer, the model kept having similar results. This emphasizes that the attention mechanism that we used was not appropriate for our task and we need to reconsider this choice in the future. Furthermore, for the LSTM case, accuracy was bigger (59.19%) than that of the solutions that provided a bidirectional view. All transformer-based techniques used in feature extraction surpassed the models that used static word embeddings, an expected result as they take into account semantics and represent the new baseline of this problem nowadays. In our case, the best accuracy was 63.61%, value achieved by associating DistilBERT with a simple feed-forward network for classification. BERT-Small and BERT-Medium traits led to lower performances (58.48%, respectively 60.06%), but they still overcame those of Word2Vec or GloVe. A possible explanation for this is that their word embeddings were not as precise and as relevant as those of DistilBERT, as they were achieved through architectures with smaller number of parameters.

If we compare with the state-of-the-art results from Chapter 3, our best embeddings-based model had a higher performance than all our reference points. It scored better accuracy than both classical solutions (SVM, LR, NB, SGDClasifier, Random Forest, DT) and deep-learning ones (LSTM, BiLSTM, C-LSTM, Feed-Forward, CNN, Conv-HAN). We concluded that DistilBERT could extract traits that were more relevant for the fake news detection task than Word2Vec, GloVe, n-grams, lexical features, sentiment indices or even RoBERTa. The model improved the highest metrics from previous section with 2%, while it was better than the “random choice” by about 13%. Moreover, our most accurate LSTM (59.19%) and BiLSTM (58.40%) algorithms exceeded the performance of the same models from previous published papers, but not those from previous section.

For multi-label classification, the differences between our models were very small, as most of them scored an accuracy around 23-24%. This time, there was no clear distinction between solutions which used Word2Vec or GloVe. So, we can intuit that in this case none of these two characteristics extraction algorithms provided enough informative traits for our

task, at least in combination with these classifiers. One important aspect to notice, is that the self-attention layer added to the BiLSTM architecture, had a slightly more powerful impact than in the case of binary grouping, improving precedent results till 24.30%. With very small differences, BERT-Small and BERT-Medium overcame the solutions based on static embeddings (24.54%, respectively 24.39%), situation constant in both our study cases (6/2 labels). DistilBERT combined with a Feed-forward network continued to achieve the best performance. It had 27.30% accuracy, being the model, which classified news from the LIAR dataset with the best precision until this point. The same features extraction technique combined with a BiLSTM architecture offered poorer results, precisely 25.33%, but they were still better than the rest of this section.

Our solution for multi-label classification surpassed the results from the state of the art: LR, SVM, NB, DT, Random Forest, LSTM CNN, BiLSTM. As in the other case, DistilBERT worked better than surface-level linguistic patterns, Word2Vec, GloVe n-grams or FastText. In addition, it overcame the accuracy of the most precise SVM model from previous section with 1.5%.

Summarizing, a constant order of performance for some models can be deduced in both cases, multi-class classification, and binary classification. Specifically, we have: LSTM (W2V-skip) < BiLSTM (W2V-cbow) < BiLSTM (W2V-skip) < BiLSTM Attention < FF (BERT-small) < BiLSTM (DistilBERT) < FF (DistilBERT). It can be concluded that for the LIAR dataset, contextualized word embeddings surpassed static word embeddings in all cases. This instantly emphasized that BERT could deal with more complex text inputs unlike Word2Vec, or GloVe and that context and semantics are important pieces in the process of false content detection. Moreover, GloVe tended to lead to higher results than Word2Vec, expected aspect as its architecture is more complex. The highest performant model for the task of fake news detection was in both cases a fine-tuned model that used the weights of DistilBERT as embeddings and classified them through a simple feed-forward network, achieving a performance of 27.30%, respectively 63.61%.

These solutions outperformed all classical algorithms and recurrent neural networks solutions from the state-of-the-art papers presented in previous chapter. For both multi-class and binary classification there are a couple of algorithms in the state of the art (ex.: attention-based, BiLSTM-GRU) which still have higher accuracy, but they do not provide enough implementation details to be reproduced or compared with, so they are out of the scope of this thesis. If we look at the first category of models that used TF-IDF for features and classical machine learning algorithms, the results with static embeddings and recurrent neural networks are similar or slightly poorer, even if the architectures used are stronger. Thus, we can conclude that for the statements from PolitiFact.com, the frequency of words is more relevant than the position and order. Although, the meaning of the words proved to be useful when deducing the true value of a text, as semantic characteristics increased performance.

Table 16 Results for word embeddings

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	**Precision (%)**	**Recall (%)**	**F1 (%)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 (%)**
Word2Vec – continuous bag of words	LSTM	Validation	25.15	23.56	25.16	22.22	58.25	58.26	58.26	58.26
		Testing	24.38	21.69	24.39	21.04	57.45	57.63	57.46	57.53
	BiLSTM	Validation	23.75	25.02	23.75	21.52	55.99	56.21	56.00	54.01
		Testing	23.36	21.10	23.36	20.68	56.66	55.31	56.67	54.46
Word2Vec - skip gram	LSTM	Validation	22.58	22.58	22.59	22.46	55.52	55.62	55.53	55.55
		Testing	22.80	23.09	22.81	22.80	55.56	55.88	55.56	55.68
	BiLSTM	Validation	25.46	22.52	25.47	19.65	54.75	56.55	54.75	48.16
		Testing	23.44	17.98	23.44	17.97	58.01	57.13	58.01	52.17
GloVe	LSTM	Validation	23.67	23.72	23.68	23.36	58.09	58.01	58.10	57.95
		Testing	23.52	23.63	23.52	23.20	59.19	59.00	59.19	59.07
	BiLSTM	Validation	22.97	22.75	22.98	22.73	56.77	56.66	56.78	56.49
		Testing	23.36	23.27	23.36	23.11	58.40	58.16	58.41	58.25
	BiLSTM Attention	Validation	24.06	24.09	24.07	23.83	57.94	58.08	57.94	57.96
		Testing	24.30	24.84	24.31	24.33	58.40	58.93	58.41	58.56
BERT- Small	Feed Forward	Validation	27.41	27.09	27.41	27.02	62.77	62.73	62.77	62.73
		Testing	24.54	24.32	23.55	24.03	58.48	58.50	58.48	58.49
BERT-Medium	Feed Forward	Validation	27.02	27.39	27.02	26.90	61.05	61.01	61.06	60.91
		Testing	24.39	24.69	24.39	24.26	60.06	59.62	60.06	59.68
DistilBERT	Feed Forward	Validation	25.38	24.66	25.39	23.68	61.44	62.07	61.45	60.32
		Testing	***27.30***	***27.69***	***27.31***	***26.30***	***63.61***	***63.26***	***63.61***	***62.38***
	BiLSTM	Validation	25.38	25.97	25.39	24.16	60.20	60.15	60.20	59.97
		Testing	25.33	25.40	25.34	24.93	61.32	61.03	61.33	61.10

4.3 Classification based on stylometric features

The results of classification using standard machine learning classifiers together with the stylometric features expressed through text complexity indices are presented in Table 17. To obtain these results, we used indices from all 5 classes described in Section 4.2.4, together with all their variations and all possible levels (word/sentence/paragraph/document) as representative features, instead of the text itself. We started from the hypothesis that all these extracted characteristics describe the author's writing style, and they may betray his negligence in composing false content. Therefore, they may lead to an improvement in the performance of the models analyzed up to this point. Thus, in order to confirm this, we tested these traits first on their own and afterwards concatenated with other features.

From Table 17, it can be observed that we managed to build a group of classifiers with less promising results. For multi-label classification, the model which achieved the maximum accuracy was the quadratic SVM, while for the second case, NB surpassed the other three. We obtained a maximum value of 20.33%, respectively 56.89% on the test partition, values better than the random choice of a label with only 4-6%. It's important to highlight how different the models behaved in our two cases of study, being impossible to establish a general order of performance. Despite having the best accuracy for the binary case, NB performed as good as arbitrary choice in the multi-class case (17.49%). Moreover, DT algorithm managed to find some basic rules for the six classes, exceeding linear SVM, but not for the case of two labels when scored only 51.61%.

To see if these models were really that weak, we analyzed the confusion matrixes. We found out that even though they classify the texts in such a wrong way, they generally confuse adjacent classes. For example, many news items labeled "barely-true" have been categorized as "half-true" or "false", neighboring classes. Similarly, "half-true" texts have been marked as "barely-true" or "mostly-true". The "true" class is the only one for which there has been more confusion. Instead, the binary model distinguished slightly better values for the "true" class, correctly classifying several entries, which means that it found several distinctive features for this category compared to the "fake" one. Taking this aspect into account, plus the fact that the dataset has a high difficulty and the differences between the performance of our best model and those of these were less than 10%, we can say that the models managed to learn some characteristics for each class. Therefore, these indices may be helpful criteria in establishing veracity, but not on their own.

For the second series of experiments, we used the concatenation of features extracted by TF-IDF with text complexity indices. We kept only features which refer to words, as they were the ones for which we scored the highest accuracy in Section 5.1. In this sense, the same DT, NB and SVM classifiers were used, the results being shown in Table 18. This new set of traits caused a decrease in performance for all models with one exception. As shown there, in all cases, the best results were obtained with the DT model for both series of features: unigrams and unigrams concatenated with bigrams, while the weakest with the linear SVM algorithm, as opposed to the first series of tests. The maximum performance metrics were 22.02% and 58.32%, for the case of using unigrams and bigrams of words and stylometric

features. For multi-label classification, using bigrams of words associated with ReaderBench indices affected the results in a good manner for three out of four models. Also, the performance metrics indicated better values in all cases for the "true" class, which emphasizes the fact that it was easier to find common words and style patterns to distinguish texts with this tag. The history repeated in the second case of binary grouping, meaning that NB models obtained higher performances than linear or quadratic SVM (56.43% vs. 54.05% and 55.32%). Also, this time, bigrams had a positive impact only for DT and linear SVM classifiers. With this input, Decision Tree managed to understand the information behind text complexity indices and even improved the accuracy from previous section for the 2-labels (57.46%).

Comparing our results with those in the state of the art, we can see that the single use of the text itself was easier to distinguish, but we managed to obtain similar values. For instance, for SVMs, the highest values obtained for multiple label classification were weaker than those in the reference article (20.43% vs. 25.50%), being better than the "random choice" by only about 4%. This implies that the combination of the two inputs was not compatible with this model. The Naïve Bayes algorithm exceeded the performance of the SVM models (21.31%), but not that of the models in (Wang, 2017). DT model managed to best interpret the input used in this experiment, but not enough to obtain an accuracy close to the SVM model in the reference papers or to our previous tests. For the binary classification, Decision Tree managed to obtain a higher accuracy than its approach in the state of the art (58.32% vs. 51.00%) and similar score with other classical machine learning methods (59-61.00%). Even if we added additional data, SVMs could not separate the two classes in an ideal manner but scored an accuracy approximately equal to that from the state of the art (55.32% vs. 55.50%). Also, NB from the state of the art achieved 60%, while our solution was at a maximum of 56.43%.

Furthermore, we introduced these stylometric features also in our neural network approaches. The results of the classification using the LSTM and BiLSTM classifiers, together with the features extracted by Word2Vec/GloVe and the stylometric elements are shown in Table 18. In addition, they are compared with those obtained using BERT/DistilBERT and text complexity indices with a feed-forward neural network. Although a strict constant order of performance of all models cannot be deduced in both cases, multi-class and binary classification, a pattern can be generalized: BiLSTM < LSTM < NN. If we apply the same principle on feature extraction methods, we can conclude that: GloVe < Word2Vec < BERT < DistilBERT.

For multi-label classification, adding various text indices was not a good choice, as our models decreased in accuracy compared to those from previous section. The only exceptions were when we opted for Word2Vec-skip gram, case when performance increased to 23.28% (LSTM), respectively 24.38% (BiLSTM) and when we used BERT-Small, metric reaching 26.28%. More than that, this time, the use of skip-gram features with complexity elements improved results compared with the continuous bag of words approach of the same algorithm (22.96% - LSTM, 18.86% - BiLSTM). This time, Glove was part of solutions with very poor metrics, while BERT-Medium kept its performance (24.23% vs. 24.39%). The

fine-tuned model with DistilBERT and the Feed-Forward network remained the one with the highest performance, 26.91%. Regarding the classifiers, adding bidirectional perspective to Long-Short Term Memory improved results only for skip-gram, aspect that repeated in most of our experiments. Adding an attention layer at our BiLSTM architecture had a negative impact, decreasing accuracy to 20.99%.

Even though we treated a simpler case, binary segmentation, the situation was identical. Performances did not improve for majority, compared to those from Section 5.2. This time, the exemption was the solution with BERT-Small, text complexity indices and a feed-forward network which improved its performance to more than 60%. Also, Word2Vec-skip gram models felt below the ones based on continuous bag of words, in terms of accuracy, but they continued to be a better match for BiLSTM models. The combo of GloVe, ReaderBench elements and LSTM led to a performance of 57.69%, value larger than any other solution based on static embeddings from this section. This algorithm of traits extraction was also paired with BiLSTM, but the results got worse. Adding an attention layer caused the accuracy to drop below "random choice" with more than 5%. Even when paired with additional information, contextualized word representations had an advantage in fake news detection. BERT-Small and BERT-Medium had results around 60%, good values for this dataset. The top performance was obtained through the same solution, DistilBERT, stylometric elements and a simple feed-forward, 63.14%.

Summarizing, the results obtained with (Bi)LSTMs were not as good as would have been expected considering that the architectures are more complex than in the previous case, and Word2Vec and Glove capture more features than just the frequency. Compared to the models which did not use text complexity metrics, the results had worsened with at least 2%, which means that the modified input does not significantly help the model in the learning process. The approach that included attention in the classification mechanism managed to score the lowest accuracy, fact which highlights the incompatibility between stylometric elements and self-attention. An important aspect that improved was the reduction of over-specializing tendency that we encountered in our preceding case. For feed-forward neural networks, the performance metrics obtained by adding stylometric elements were the best, comparable to those in the previous section (26.91% vs. 27.30% and 63.14% vs. 63.61%). The model with DistilBERT, text indices and NN brought improvements to all the other models presented in this section. Thus, this feature extraction technique was the most efficient for the LIAR data set. Moreover, it seemed that BiLSTM model almost unanimously classified the texts as "barely-true", while LSTM and Feed-Forward had more balanced results.

If we draw a line and compare these new values with previous section, there is no strong evidence that text complexity indices offered relevant information for our problem, as most of the models obtained similar or lower performances when adding this kind of information. However, by contrast with the state of the art, values were not as poor as expected. For instance, for LSTMs, the highest values obtained for multi-label classification were still better than those in the reference article (23.28% vs. 22.50%), beating "random choice" by about 7%. This implies that the combination of the two inputs helped this neural

network extract patterns for the six classes. In plus, BiLSTM also exceeded the performance of the same approach (24.38%) and some classical machine learning algorithms, but not that of CNNs or SVMs. Feed-forward network model managed to best interpret the input used in this experiment, enough to obtain an accuracy close to the CNN model in the LIAR paper. For the binary classification, LSTM managed to obtain a higher accuracy than its approach in the state of the art (57.69% vs. 54%) and similar score with CNN, HAN, Conv-HAN or C-LSTM (55-58%). When grouping text indices with static word embeddings, BiLSTM could not overcome the value from the reference papers, achieving at most 56.43%. Its results were in the same range as simpler algorithms like AdaBoost or k-NN, exceeding only solutions that implied simple Long-Short Term Memory. All the solutions that integrate feed-forward networks led to metrics comparable with those from the state of the art.

Overall, the stylometric indices failed to raise the accuracy of the models on the test dataset compared to the best results obtained when only TF-IDF/ Word2Vec/ GloVe/ BERT/ DistilBERT were used to extract features. In fact, the elements of text complexity seem to bring additional features that are not distinct enough to assure a very precise learning, at least in the large number used for these experiments, confusing the model. One explanation for the weaker results could be that the extracted stylometric traits may not be relevant enough to be used together, requiring an additional analysis step to indicate exactly which indices are important for our problem and which ones weight the most in the classification process. We started to investigate this idea, adding one index at the time to our best models and follow up the changes in performance. Unfortunately, there were no important improvements from one stage to another. Therefore, we did not include those experiments in this thesis.

Table 17 Results for stylometric features

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
RB text complexity indices	DT	Validation	17.30	17.51	17.30	17.27	53.27	53.21	53.27	53.22
		Testing	19.55	20.01	19.55	19.69	51.61	51.60	51.61	51.61
	NB	Validation	18.08	23.15	18.08	17.44	54.74	54.62	54.74	54.68
		Testing	17.49	21.15	17.50	16.94	***56.89***	***56.79***	***56.89***	***56.69***
	SVM linear	Validation	19.16	18.97	19.16	18.93	54.64	54.49	54.64	54.32
		Testing	19.26	19.14	19.26	19.03	54.35	53.81	54.35	53.99
	SVM quadratic	Validation	18.38	17.93	18.38	18.07	54.54	54.48	54.55	54.49
		Testing	***20.33***	***20.44***	***20.33***	***20.20***	56.30	56.50	56.30	56.39

Table 18 Results for stylometric features combined with textual features

Feature	Model	Dataset partition	Multi-class classification				Binary classification			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
TF-IDF of unigrams of words and RB text complexity indices	DT	Validation	22.50	22.56	22.51	21.69	58.64	58.84	58.64	57.64
		Testing	*21.55*	*20.56*	*21.55*	*20.35*	*58.24*	57.47	58.25	57.36
	NB	Validation	23.36	22.96	23.36	20.15	59.19	59.35	59.19	58.35
		Testing	20.83	26.23	20.84	18.85	56.43	55.66	56.43	55.69
	SVM linear	Validation	18.96	18.80	18.96	18.74	53.95	53.81	53.96	53.70
		Testing	19.45	19.46	19.45	19.26	53.07	52.39	53.08	52.61
	SVM quadratic	Validation	17.79	17.37	17.79	17.49	54.93	54.83	54.94	54.79
		Testing	20.23	20.25	20.23	20.08	55.32	55.34	55.33	55.33
TF-IDF of unigrams & bigrams of words and RB text complexity indices	DT	Validation	22.04	21.96	22.04	21.01	59.11	59.04	59.11	58.90
		Testing	*22.02*	*20.06*	*22.02*	*20.26*	*58.32*	*58.29*	*58.33*	*58.31*
	NB	Validation	25.23	18.28	25.32	18.95	57.24	58.64	57.24	53.77

		Testing	21.31	12.73	21.31	15.86	55.40	53.32	55.41	51.68
	SVM linear	Validation	19.35	19.25	19.35	19.16	53.17	53.03	53.18	52.98
		Testing	19.25	19.21	19.26	19.04	54.05	53.39	54.06	53.60
	SVM quadratic	Validation	17.69	17.29	17.69	17.41	55.52	55.42	55.52	55.39
		Testing	20.43	20.45	20.43	20.28	54.64	54.59	54.64	54.62
Word2Vec – continuous bag of words and RB text complexity indices	LSTM	Validation	19.85	19.37	19.86	19.09	54.67	54.56	54.67	54.55
		Testing	22.96	22.72	22.97	22.23	57.06	56.86	57.06	56.94
	BiLSTM	Validation	19.93	20.18	19.94	19.64	54.82	54.70	54.83	54.65
		Testing	18.86	19.83	18.86	18.82	56.43	56.16	56.43	56.26
Word2Vec – skip gram and RB text complexity indices	LSTM	Validation	23.67	23.43	23.68	23.43	55.48	55.71	55.49	55.58
		Testing	23.28	23.14	23.28	23.15	55.45	55.54	55.45	55.47
	BiLSTM	Validation	22.97	23.07	22.98	22.88	54.67	54.54	54.67	54.50
		Testing	*24.38*	*24.22*	*24.39*	*24.21*	56.35	55.72	56.35	55.82
GloVe and RB text complexity indices	LSTM	Validation	22.04	23.69	22.04	21.19	57.47	57.44	57.48	56.82
		Testing	22.25	22.28	22.26	21.47	*57.69*	*56.96*	*57.70*	*56.94*
	BiLSTM	Validation	21.80	22.53	21.81	21.75	55.06	54.99	55.06	55.00
		Testing	22.17	22.39	22.18	22.08	56.27	56.19	56.27	56.23
	BiLSTM Attention	Validation	19.54	21.93	19.55	13.30	47.97	23.02	47.98	31.11
		Testing	20.99	24.83	20.99	14.99	43.64	19.05	43.65	26.52
BERT-Small and RB text complexity indices	Feed-Forward	Validation	25.46	26.21	25.47	25.14	59.89	59.83	59.89	59.82
		Testing	26.28	26.70	26.28	26.07	60.61	60.47	60.62	60.53
BERT-Medium and RB text complexity indices	Feed-Forward	Validation	24.06	25.13	24.07	23.89	60.12	60.34	60.12	60.12
		Testing	24.23	24.76	24.23	23.93	59.66	60.27	59.67	59.82
DistilBERT and RB text complexity indices	Feed-Forward	Validation	26.24	26.60	26.25	25.98	61.37	61.75	61.37	60.52
		Testing	***26.91***	***27.05***	***26.91***	***26.63***	***63.14***	***62.69***	***63.14***	***62.13***
	BiLSTM	Validation	24.22	24.73	24.22	23.28	61.29	61.24	61.29	61.20
		Testing	21.70	22.95	21.70	20.92	57.22	57.00	57.09	57.09

5 Discussion

In the previous chapter, we summed up a collection of fake news detection solutions which learn various patterns of untrustworthy content. Despite having similar results with those from Section 3, there are multiple aspects that restrict their final performance, aspects that we will try to discuss further. A series of arguments can be put forward to explain the existing limitations and based on them, we can establish further steps that may diminish the impact.

Firstly, the weak results could be explained by the choice of the data set. A well-balanced, unbiased, diverse, and real-life-inspired corpus can greatly influence the outcome of the experiments. In our case, some classes have less examples comparing with the others, hindering the learning process. Also, they are very specialized, focusing just on specific events from reality. Moreover, LIAR has six classes, and it contains many short real-world statements from various contexts and different authors, being very difficult to place them in a pattern. Therefore, the models cannot learn general patterns for the fake news detection task. Further, we analyzed the structure of LIAR and found out different limitations like typos, atypically organized or incomplete sentences that may justify the mistake prone models. Such examples that make the task difficult are:

- typing errors:

*Says Dan Sullivan approved a light sentence for a sex **offenderwho** got out of prison **andis** now charged with a gruesome murder and sexual assault*

(offender who; and is)

*Jeff Greene on why **hewent** Cuba.*

(he went)

***Americanschools** are more segregated than they were in the 1960s.*

(American schools)

- extra punctuation marks:

Barack Obama ""pays for every dime "" of his spending plans.

- missing punctuation marks:

*If you **dont** buy cigarettes at your local supermarket, your grocery bill **wont** go up a dime.*

(don’t; won’t)

- incomplete sentence:

On gay marriage.

On abortion rights.

- confusing sentence:

Says he was a Marine during Vietnam.

(Who is “he’?)

These constraints affect all the features we have been tested: frequency, word embeddings and text complexity indices. For instance, some of the ReaderBench metrics - surface indices, word complexity indices, morphological indices - are strictly related to the shape of the text. A corpus with grammatical errors, incomplete or absent words, extra or less punctuation marks, unclear and non-fluent texts, can negatively influence the values of these indices. In turn, stylometric metrics influence the final label determined by the classifiers. Thus, if they are not correctly calculated, they will lose their information and relevance and they will lead to erroneously learned patterns. In addition, the typos from the input especially influence the features extracted through TF-IDF, Word2Vec, GloVe, BERT and DistilBERT, as they lead to words which do not exist in the dictionary, with unidentified stem or lemma. Unknown words raise difficulties in both computing frequency and vector space representation.

One possible solution for improving the format of the dataset statements is to apply some precursor operations for correction. For example, confronting words with a dictionary and finding the closest replacement through minimal number of character changes may assure the removal of lexical-grammatical errors. A more complex option is using a predicting model that can find the most probable word that fits in a sentence in case of discovering a token that does not belong to the glossary. Moreover, texts may be validated through a checker that signals punctuation marks mistakes or incoherent sentences. This additional editing of the corpus may also influence the pre-processing stage, as clean text is easier to be standardized than the noisy one. For a different format of the text, we may need to change the sequence of operations we applied by adding, removing, or even replacing some steps.

Figure 3 Example different indices with identical information: wd_sent, wd_par, wd_doc

Secondly, another aspect of the dataset that influences the experiments based on indices of complexity is the length of the input. Specifically, LIAR is a corpus that contains in general short texts with a few statements, not complex documents, or paragraphs. This implies that certain text indices will have identical values, regardless of the level of granularity chosen. For instance, if we have a single sentence entry, the document, paragraph, and sentence will be equal. Thus, the information sent to the models may be consistent, but duplicated and without essence. Figure 3 shows the average number of words for an example

at the sentence, paragraph, and document level. It can be seen that although we have three indices, the histograms of two are identical, and the third has very similar information. In the future, to avoid sending extra data to the model, we should filter out all the stylometric indicators that do not help the learning process.

The problem of input length can also affect the embeddings of the words. We used a fixed size of 100 for embeddings vectors in all our experiments. Although the majority of texts had less than 100 tokens, 7 out of over 12.000 news were truncated as they had dimensions between 186 and 512. Most of them were padded with zeros as the average length is 20. The problem appears when we lose too many information or when we add too much irrelevant one - noise. If we consider those 7 elements as outliers and remove them, then the constant size that we chose may have not been the most appropriate one, a smaller one being more helpful. More detailed experiments for finding the best embedding dimension are in our short-term plans.

Another issue which came from using the ReaderBench tool is that a significant percentage of the calculated indices were null for all classes, adding no informational gain. This can be determined by several aspects. One of them is the simplistic structure of the dataset manifested through the small length of the texts. As a result, it is almost impossible to extract clues related to discourse structure or semantic cohesion from short news, aspect which automatically leads to zero values for all classes. Another reason is the lack of diversity of the entries. For example, out of about 12.800 entries, there was no example containing dependencies of type “classifier” or “copula”, which means that these metrics do not bring knowledge to the models. Thereby, because of the little information that these indices brought, we limited the number of experiments that used them.

Fourthly, embeddings representations may also be influenced by the strategy we adopted for words that do not belong to the vocabulary. We chose a quite simple approach in all cases, meaning that we added a new word to the vocabulary, called “UNK”, with a fixed numerical array of zeros; then we associated it with every unknown word. Taking into account that on average approximately 2.000/13.000 (15%) unique words are not in the glossary of our feature extraction methods, this aspect may explain the limited results of these methods. GloVe had the largest number of null embeddings, which led in general to confusing results. One method that may improve news numerical representation is using a predicting algorithm for the out-of-vocabulary terms. In this way, we may find useful embeddings based on morphology and context.

The analysis of the performance of the previously presented models can continue beyond the metrics used: Precision, Recall, Accuracy and F1-Score. More than these easy-to-understand numerical values for a computer, it is interesting for a person to analyze why a certain classifier gave a certain verdict. If we only knew the criteria taken into account, we could understand which are the reasons for which the result is not as expected. In this direction, we analyzed the correlation between the obtained labels and the text complexity indices, trying to deduce some of the learned rules. Although more than 1000 indices were used, not all of them contributed to the differentiation between classes. In addition to those with zero values mentioned above, we also found sufficient indices that had similar values for

the two classes. In fact, if we analyze this direction further, we can see that the values obtained for these indices are generally small, and their variation between classes is mostly at the level of decimals. However, even with these small changes, some variables play a more important role than others. To highlight this, we calculated the Gini scores of importance and summarized the highest values in Figure 4. The importance of Gini or Mean Decrease in Impurity (MDI) calculates the importance of each trait, in our case of each complexity index, as the sum over the number of divisions in a forest of trees that include the characteristic, proportional to the number of samples it divides. A higher value indicates a higher importance of the feature.

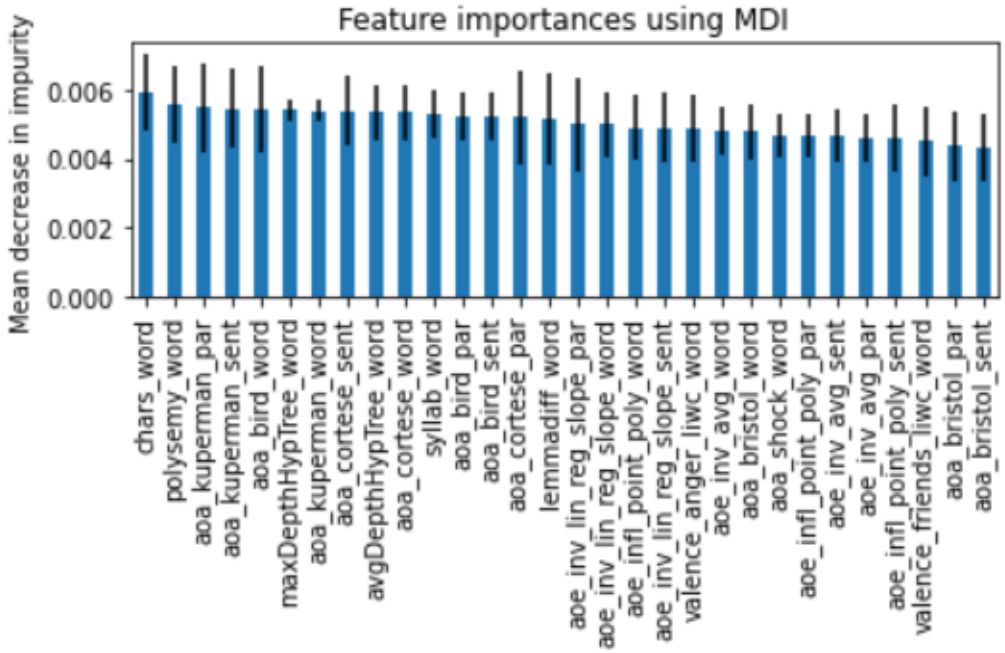

Figure 4 Text complexity indices with the highest Gini score

Given that very small variations were obtained for this dataset, the model failed to learn the defining features of a certain class. However, there were several indices that recorded positive values only for a certain class. This can also be an explanation for some of the labels obtained because once those indices were met, they were associated with that class. Thus, Table 19 highlights such examples, mainly in terms of valence and named entities, so indices of word complexity. We can conclude that the news with false information was associated with mostly negative feelings: despair, disappointment, dissatisfaction, humiliation, sadness, with a few exceptions: hope, surprise. On the other hand, articles that present real facts were written in a positive tone, leaving traces of emotion, touch, serenity, joy, and relief. Also, our theory was that evoking real-world elements like persons, locations, organizations, languages through their proper name increases the reader's confidence. This idea was confirmed as truth texts tended to use named entities, unlike the opposite class.

Table 19 Examples of text complexity indices specific for a class

fake	true
valence_desperation_galc_sent	valence_beingtouched_galc_sent
valence_hope_galc_sent	valence_enjoyment_galc_sent
valence_dissapointment_galc_sent	valence_relief_galc_sent
valence_surprise_galc_sent	valence_serenity_galc_sent
valence_skipt_lasswell_sent	valence_rcends_lasswell_sent
valence_sadness_galc_sent	valence_ingest_liwc_sent
valence_humility_galc_sent	nmdEnt_language_sent
valence_dissatisfaction_galc_sent	nmdEnt_location_sent

6 Conclusions and Future Work

The purpose of this thesis was to analyze the problem of detecting false content through a linguistic approach based just on the content of the texts. Therefore, we proposed different approaches of selecting the characteristics of news and feed them to an automatic model to learn from. This direction has allowed experimentation with several solutions, focusing on pre-processing data, extracting features and classification. At the beginning, we tried to implement some traditional machine learning algorithms as they lead to good results, and later we shifted to more complex solutions, based on neural networks. In addition, several traits extraction methods were tried: term frequency, static and contextualized word embeddings. Moreover, we came up with the idea that the author's intention to voluntarily write misinformation will be reflected in his writing style and that we may be able to deduce a series of explanations of the final labels if we identify them. So, we extracted over 2000 text complexity indices for each text entry and integrated them in the previous solutions. We treated the problem from two different points of views: 2-label classification and a more complex one, 6-label classification.

In the end, we kept the best results per feature type and classification case and compared them for an overall perspective in Table 20 and Table 21. The highest classification results were obtained in both cases using a feed forward network combined with a fine-tuned DistilBERT, which achieved for the test partition an accuracy of 27.30% on 6-labels, respectively 63.61% on 2-labels. This outcome highlights from the start that contextualized word embeddings through they semantic-based approach have the biggest potential for detecting patterns of false content.

Table 20 Overall results for multi-class classification

Features	**Model**	**Test accuracy (%)**
TF-IDF of unigrams of words	SVM linear	23.36
TF-IDF of unigrams & bigrams of words	SVM linear	24.70
TF-IDF of unigrams of stems	SVM quadratic	24.70
TF-IDF of unigrams & bigrams of stems	SVM linear	23.84
TF-IDF of unigrams of lemmas	SVM linear	25.41
TF-IDF of unigrams & bigrams of lemmas	SVM linear	25.26
TF-IDF of unigrams & bigrams of words and unigrams of lemmas	SVM quadratic	24.94
TF-IDF of unigrams & bigrams of words and unigrams & bigrams of lemmas	SVM linear	24.63
Word2Vec – continuous bag of words	LSTM	24.38
Word2Vec - skip gram	BiLSTM	23.44
GloVe	BiLSTM Attention	24.30

BERT- Small	Feed Forward	24.54
BERT-Medium	Feed Forward	24.39
DistilBERT	**Feed Forward**	**27.30**
RB text complexity indices	SVM quadratic	20.33
TF-IDF of unigrams of words and RB text complexity indices	DT	21.55
TF-IDF of unigrams & bigrams of words and RB text complexity indices	DT	22.02
Word2Vec – continuous bag of words and RB text complexity indices	LSTM	22.96
Word2Vec – skip gram and RB text complexity indices	BiLSTM	24.38
GloVe and RB text complexity indices	LSTM	22.25
BERT-Small and RB text complexity indices	Feed Forward	26.28
BERT-Medium and RB text complexity indices	Feed Forward	24.23
DistilBERT and RB text complexity indices	Feed Forward	26.91

Table 21 Overall results for binary classification

Features	**Model**	**Test accuracy (%)**
TF-IDF of unigrams of words	SVM quadratic	61.88
TF-IDF of unigrams & bigrams of words	SVM linear	62.19
TF-IDF of unigrams of stems	SVM quadratic	61.17
TF-IDF of unigrams & bigrams of stems	SVM linear	61.96
TF-IDF of unigrams of lemmas	SVM linear	61.72
TF-IDF of unigrams & bigrams of lemmas	NB	61.64
TF-IDF of unigrams & bigrams of words and unigrams of lemmas	SVM linear	61.40
TF-IDF of unigrams & bigrams of words and unigrams & bigrams of lemmas	NB	61.48
Word2Vec – continuous bag of words	LSTM	57.45
Word2Vec - skip gram	BiLSTM	58.01
GloVe	LSTM	59.19
BERT- Small	Feed Forward	58.48
BERT-Medium	Feed Forward	60.06
DistilBERT	**Feed Forward**	**63.61**
RB text complexity indices	NB	56.89
TF-IDF of unigrams of words and RB text complexity indices	DT	58.24
TF-IDF of unigrams & bigrams of words and RB text complexity indices	DT	58.32
Word2Vec – continuous bag of words and RB text complexity indices	LSTM	57.06
Word2Vec – skip gram and RB text complexity indices	BiLSTM	56.35
GloVe and RB text complexity indices	LSTM	57.69

BERT-Small and RB text complexity indices	Feed-Forward	60.61
BERT-Medium and RB text complexity indices	Feed-Forward	59.66
DistilBERT and RB text complexity indices	Feed-Forward	63.14

For most features extracted with TF-IDF, the best score was obtained with a SVM, either linear or quadratic. There were only two exceptions that reached better accuracy with Naïve Bayes. Overall, the solutions based on frequency had values close to the state of the art, around 24-25%, respectively 60-61%. This emphasizes that for our task, frequency of terms represents informative traits. Due to the promising results obtained with TF-IDF and the basic machine learning algorithms, it can be said that there is a link between the frequency of words used and the degree of veracity of the text. Furthermore, the fact that the meaning and order of the words were ignored in the first approaches and that those algorithms did not capture the dependencies between the words, did not affect the results as much as expected. Also, as it turns out from our experiments, there are not large differences between simpler or more complex classifiers, as the most important element is represented by the features used in each case. All our models have performances close to the state of the art, 27% and 62%.

Another point to note is that the performance of recurrent neural networks has not been as good as expected. They obtained poorer results and tended to overfit. So, we can conclude that the task of predicting the veracity of a text using this type of approach is not so promising, at least not for this dataset. In both cases, Word2Vec-continuous bag of words algorithm scored the highest accuracy with a LSTM architecture, while skip-gram with BiLSTM. GloVe had different performances, reaching 59.19% when fed to LSTM for the 2 labels case and 24.30% when paired with BiLSTM Attention for the 6 labels one. For static word embeddings, we obtained inferior metrics compared to those from the first series of experiments, proving that using more complex information do not boost a model up if not paired with the right classifier. On the contrary, for the LIAR dataset, they bring just noise, as the features decreased model performance when attached to the list of inputs. Overall, our corpus proved to be more suitable for n-grams than Word2Vec or GloVe.

However, all previously mentioned models lose context and therefore semantic. It is common knowledge that the meaning of a word depends on the neighboring words, and it should be represented according to them. This is obvious by the increase in performance that appears when employing features extracted through transformers. Therefore, BERT managed to get better results than static word embeddings and similar with TF-IDF, while DistilBERT surpassed all of them.

Our experiments also showed that the text complexity indicators are not decisive factors for the degree of truth of a text, as they do not bring many improvements. When used on their own, style cues led to the weakest metrics in this series of experiments. Moreover, the hypothesis that the author's writing style can be expressed through additional traits that could fine-tune a model was proven wrong, as the data expressed by the stylometric features brought almost no improvement on the model. However, we were able to determine some connections between the features used at the input and the labels obtained at the output, to

provide a number of explanations to the user. Thus, it could be deduced that the model with the best results learned a series of rules to differentiate between the two main classes: true and false. In this process, the indices of word complexity had the greatest contribution through words valence.

Besides short-term improvements that we mentioned in the previous chapters, there are a multitude of directions that may lead to systems that behave much better in real-life situations. The first one is introducing in the input additional meta-data about the world. Despite the disadvantage that for daily use of such systems, in general we do not have access to such data, we can deduct from the state of the art that extra information can boost up the performance of the models. So, as LIAR has multiple columns that we did not use until this point, we intend to test this path and see how the metrics evolve.

The architectures tested so far are supervised solutions that focus on extracting the features that are further used in classification. Another direction we propose is to approach the problem in an unsupervised way and analyze the structure of the corpus data. Specifically, we want to see if semantically similar news would be in the same class depending on the clusters obtained. Thus, by analyzing the purity and cohesion of a cluster, we would be able to measure the level of trust that the news in certain fields have and to extract some trends from the existing data.

Nowadays, for fake news detection, the field of Explainable AI is one of interest, so we want to continue to research this direction. First, improvements can be made to previous models, to the point where a clear correlation is determined between text complexity indices and the degree of truth by studying the weights of the models. Another alternative could be the ProSeNet (Prototype Sequence Network) architecture which generates explanations by comparing the input with typical training cases or LIME library which uses the components of an interpretable model built in the vicinity of the instance it wants to explain. Also, we can build a complementary dataset with verified information and use it as context and justification. Finally, it will be desired to compare all the options to determine which of them offers a more complete explanation and which one is closer to human thinking. It is also interesting to follow the behavior of the models implemented in different experiments such as: the transferability of model learning between several datasets or the transferability of model learning between several foreign languages.

For the long run, the aim of this project is to also analyze these algorithms for the Romanian language, which may even need the construction of such a corpus. Thus, we want to see how good the metrics would be for a language that has many syntactic and semantic particularities. The difficulty of the task of detecting fake texts would increase for the Romanian language due to the much more limited resources and the impact of the pre-processing text step. For instance, at the time of segmentation into tokens, this language can raise significant problems, such as: the elimination of letters that appear in word compressions (“într-adevar”) or regionalisms (“acu’”) or the hyphenation of an unaccented forms of pronouns with a verb (“l-am”, “m-ai”, “schimbându-se”). All these are interesting challenges that may lead to innovational solutions.

Bibliography

Agarwal, V., Sultana, P. H., Malhotra, S., & Sarkar, A. (2019). Analysis of Classifiers for Fake News Detection. *Procedia Computer Science. 165*, pp. 377-383. Elsevier B.V.

Ahmed, S., Hinkelmann, K., & Corradini, F. (2019). *Combining Machine Learning with Knowledge Engineering to detect Fake News in Social Networks-a survey*. Retrieved from arXiv: https://arxiv.org/ftp/arxiv/papers/2201/2201.08032.pdf

Alhindi, T., Petridis, S., & Muresan, S. (2018). Where is your Evidence: Improving Fact-checking by Justification Modeling. *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, 85–90.

Anoop, K., Deepak, P., & Lajish, V. K. (2020). Emotion Cognizance Improves Health Fake News Identification. *Proceedings of the 24th Symposium on International Database Engineering & Applications*.

Antoniadis, S., Litou, I., & Kalogeraki, V. (2015). A Model for Identifying Misinformation in Online Social Networks. *OTM 2015 Conferences* (pp. 473-482). Springer International Publishing Switzerland.

Aslam, N., Khan, I. U., Alotaibi, F. S., Aldaej, L. A., & Aldubaikil, A. K. (2021). Fake Detect: A Deep Learning Ensemble Model for Fake News Detection. *Complexity*.

Benevenuto, F., Magno, G., Rodigues, T., & Almeida, V. (2010). Detecting spammers on Twitter. *4th International AAAI Conference on Weblogs and Social Media (ICWSM), 6.*

Bond, C. F., & DePaulo, B. M. (2006). Accuracy of Deception Judgments. *Personality and social psychology review : an official journal of the Society for Personality and Social Psychology,, 10*, 214-234.

Brasoveanu, A., & Andonie, R. (2019). Semantic Fake News Detection: A Machine Learning Perspective. *Advances in Computational Intelligence*, (pp. 656–667).

BuzzFeed-Webis Fake News Corpus. (2016). Retrieved from Zenodo: https://zenodo.org/record/1239675

Castillo, C., Mendoza, M., & Poblete, B. (2013). Predicting information credibility in time-sensitive social media. *Electronic Networking Applications and Policy, 23*.

Chu, Z., Gianvecchio, H., Jajodia, S., & Wang, H. (2010). Who is Tweeting on Twitter: Human, Bot, or Cyborg? *Twenty-Sixth Annual Computer Security Applications Conference*, (pp. 21-30).

Ciampaglia, G. L., Shiralkar, P., Rocha, L. M., Bollen, J., Menczer, F., & Flammini, A. (2015). *Computational Fact Checking from Knowledge Networks.* Plos ONE 10.

Conroy, N. K., Rubin, V. L., & Chen, Y. (2016). Deception detection for news: Three types of fakes. *52.* Proceedings of the Association for Information Science and Technology.

Dascălu, M., Trăușan-Matu, Ș., Guțu, G., Ruseti, Ș., Paraschiv, I. C., Dessus, P., . . . Crossley, S. A. (2017). ReaderBench: A Multi-lingual Framework for AnalyzingText Complexity.

Dascălu, M., Trăușan-Matu, Ș., Guțu, G., Ruseti, Ș., Paraschiv, I. C., Dessus, P., . . . Crossley, S. A. (2022). *ABOUT*. Retrieved january 2022, from ReaderBench: http://www.readerbench.com/

Ersahin, B., Aktaş, Ö., Kilinç, D., & Akyol, C. (2017). Twitter fake account detection. *International Conference on Computer Science and Engineering (UBMK)*, (pp. 388-392).

Etzioni, O., Banko, M., Soderland, S., & Weld, D. S. (2008). *Open Information Extraction from the Web.* Communications of the ACM.

Fake and real news dataset - Classifying the news. (2019). Retrieved from Kaggle: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

Fake News - Build a system to identify unreliable news articles. (2018). Retrieved from Kaggle: https://www.kaggle.com/c/fake-news/data

Fake News Detection Challenge KDD 2020. Develop a machine learning algorithm to detect fake news. (2020). Retrieved from Kaggle: https://www.kaggle.com/competitions/fakenewskdd2020/data

FakeDeS: Fake News Detection in Spanish Shared Task. (2021). Retrieved from https://sites.google.com/view/fakedes/home

FakeNewsCorpus. (2020). Retrieved from GitHub: https://github.com/several27/FakeNewsCorpus/releases/tag/v1.0

Gomez-Adorno, H., Posadas-Duran, J. P., Enguix, G. B., & Capetillo, C. P. (2021). Overview of FakeDeS at IberLEF 2021: Fake News Detection in Spanish Shared Task. *Procesamiento del Lenguaje Natural*, 223-231.

Groza, A. (2020). *Detecting fake news for the new coronavirus by reasoning on the Covid-19 ontology.* Retrieved from arXiv: https://arxiv.org/pdf/2004.12330.pdf

Guptaa, A., Lib, H., Farnoushc, A., & Jiang, W. (2022). Understanding patterns of COVID infodemic: A systematic and pragmatic approach to curb fake news. *Journal of Business Research*, 670-683.

Han, Y., Silva, A., Luo, L., & Karunaseker, S. (2021). *Knowledge Enhanced Multi-modal Fake News Detection.* Retrieved from arXiv: https://arxiv.org/pdf/2108.04418.pdf

Hassan, N., Arslan, F., Li, C., & Tremayne, M. (2017). Toward automated fact-checking: detecting check-worthy factual claims. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1803–1812.

Hassan, N., Arslan, F., Tremayne, M., & Li, C. (2017). Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. *23rd ACM SIGKDD International Conference*, (pp. 1803-1812).

Holan, A. D. (2018). *The Principles of the Truth-O-Meter: PolitiFact's methodology for independent fact-checking*. Retrieved from PolitiFact: https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/

Hu, L., Yang, T., Zhang, L., Zhong, W., Tang, D., Shi, C., . . . Zhou, M. (2021). Compare to The Knowledge: Graph Neural Fake News Detection with External Knowledge. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 754–763.

Karimi, H., Roy, P., Saba-Sadiya, S., & Tang, J. (2018). Multi-source multi-class fake news detection. *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1546–1557.

Khan, J. Y., Khondaker, M. I., Afroz, S., Uddin, G., & Iqbal, A. (2021). *A Benchmark Study of Machine Learning Models for Online Fake News Detection.* Retrieved from arXiv: https://arxiv.org/pdf/1905.04749.pdf

Li, X., & Wu, X. (2015). *Constructing Long Short - Term Memory based deep recurrent neural networks for large vocabulary speech recognition.* Retrieved from Arvix: https://arxiv.org/pdf/1410.4281.pdf

LIAR Dataset. (2017). Retrieved from CS UCSB: https://www.cs.ucsb.edu/~william/data/liar_dataset.zip

Lin, P., Song, Q., & Wu , Y. (2018). Fact Checking in Knowledge Graphs with Ontological Subgraph Patterns. *Data Science and Engineering*, pp. 341–358.

Ma, J., Gao, M., & Wong, K.-F. (2010). Detect rumors in microblog posts using propagation structure via kernel learning. *55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 708-717). Association for Computational Linguistics.

Mayank, M., Sharma, S., & Sharma, R. (2022). *DEAP-FAKED: Knowledge Graph based Approach.* Retrieved from arXiv: https://arxiv.org/abs/2107.10648

MEX-A3T: Fake News and Aggressiveness Analysis. (2020). Retrieved from https://sites.google.com/view/mex-a3t/

Mihalcea, R., & Strapparava, C. (2009). The Lie Detector: Explorations in the Automatic Recognition of Deceptive Language. *Proceedings of the ACL-IJCNLP 2009 Conference Short Paper* (pp. 309–312). Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.* Retrieved from Arvix: https://arxiv.org/abs/1301.3781v3

Norton-Taylor, R. (1999). *The Colour of Justice.* Oberon Books Ltd.

Pan, J. Z., Pavlova, S., Li, C., Li, N., Li, Y., & Liu, J. (2018). *Content Based Fake News Detection Using.* Springer International Publishing.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics.

Perez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2019). Automatic Detection of Fake News. *27th International Conference on Computational Linguistics.*

Potthast, M., Kiesel, J., Bevendorff, J., Stein, B., & Reinartz, K. (2018). A stylometric inquiry into hyperpartisan and fake news. *56th Annual Meeting of the Association for Computational Linguistics*, *1: Long Papers*, pp. 231-240.

Potts, C. (2011, November 8-9). Sentiment Symposium Tutorial. San Francisco.

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2931–2937). Association for Computational Linguistics.

Raza, S., & Ding, C. (2022). Fake news detection based on news content and social contexts: a transformer-based approach. *International Journal of Data Science and Analytics* .

Rokach, L., & Maimon, O. (2005). *The Data Mining and Knowledge Discovery Handbook.*

Rubin, V., Conroy, N., Cornwell, S., & Chen, Y. (2017). *Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News.* NAACL-HLT.

Ruchansky, N., Seo, S., & Liu, Y. (2017). *CSI: A Hybrid Deep Model for Fake News Detection.* Retrieved from arXiv: https://arxiv.org/pdf/1703.06959.pdf

Rum Detect. (2010). Retrieved from Dropbox: https://www.dropbox.com/s/7ewzdrbelpmrnxu/rumdetect2017.zip?dl=0

Satirical Fake and Legitimate News Dataset. (2016). Retrieved from Victoria Rubin: https://victoriarubin.fims.uwo.ca/news-verification/data-to-go/

Shakeel, D., & Jain, N. (2021). *Fake news detection and fact verification using knowledge graphs and machine learning.*

Shearer, E., & Gottfriend, J. (2017). *News use across social media platforms.* Retrieved from Journalism: https://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/

Shi, B., & Weninger, T. (2016). Fact Checking in Heterogeneous Information Networks. *WWW '16 Companion: Proceedings of the 25th International Conference Companion on the World Wide Web*, (pp. 101-102).

Shrestha, M. (2018). *Detecting Fake News with Sentiment Analysis and Network.* Retrieved from https://portfolios.cs.earlham.edu/wp-content/uploads/2018/12/Fake_News_Capstone.pdf

Torgo, L., Guimaraes, N., & Figueira, A. R. (2018). Current State of the Art to Detect Fake News in Social Media: Global Trendings and Next Challenges. *In Proceedings of the 14th International Conference on Web Information Systems and Technologies*, (pp. 332-339).

Vlachos, A., & Riedel, S. (2014). Fact Checking: Task definition and dataset construction. *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pp. 18–22.

Vorhies, W. (2017, May 1). *Using Algorithms to Detect Fake News - The State of the Art*. Retrieved from Data Science Central: https://www.datasciencecentral.com/profiles/blogs/using-algorithms-to-detect-fake-news-the-state-of-the-art

Wang, W. Y. (2017). *"Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection.* Retrieved from Arvix: https://arxiv.org/pdf/1705.00648.pdf

Wang, W. Y. (2017). “Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection. *55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, (pp. 422–426).

Yazdi, K. M., Yazdi, A. M., Khodayi, S., Hou, J., Zhou, W., & Saedy, S. (2020). Improving Fake News Detection Using K-means and Support Vector Machine Approaches. *Open Science Index, Electronics and Communication Engineering*.