Truth Detection using Natural Language Processing

Sergiu-Ionuț Craioveanu

2021

Contents

Abstract

Natural Language Processing is an interdisciplinary field crossing artificial intelligence, computer science, cognitive science, and information processing [1]. It presents itself with the potential of truly impacting people's lives, being of great interest to all types of scientists and researchers, from large corporations to academia. The objective is to reach and eventually surpass human-level performance in analyzing language tasks. In this thesis, a case-study is formulated upon political affirmations within the US, where affirmations vary in their degree of veracity. We aim to understand the current development stage, and thus formulate two research questions: (1) How does deceptive language vary from its truthful counterpart? (2) How can we leverage state-of-the-art machine learning or deep learning models in an attempt to correctly classify statements based on veracity? Our focus is on the particular task of truth detection within text, but to fully grasp the progress made up until this point, we also analyze other closely related tasks. Based on our results, we manage to uncover unique insights, whilst also identifying potential opportunities for improvement.

Keywords

Lie Detection, Deception Detection, Natural Language Processing, NLP, Rumor Detection, Fake News Detection, Fact-Checking, Stance Detection, Sentiment Analysis, Post-Truth Analysis, BERT, Recurrent Neural Network, RNN, Long Short-Term Memory, LSTM, Transformers, Reformers, Performers.

1. Introduction

Being able to dissimulate one's intentions, and implicitly to detect such dissimulation, is part of a large-scale game played along the millennia. Throughout history, those with the power to manipulate could usually impose their will upon the masses. This meant that he who was persuasive enough, could change society, more often than not irrespective of moral codes – and only those who would see through the façade could ever have a chance of combatting lies – with truth or other lies.

Even though society has evolved to a present which we call the "Era of Information", it would seem that we have inadvertently slipped into the "Era of Misinformation", in which fake news spread six times faster than real information [2]. This is a worrying fact and should be a warning sign regarding the trend in which our society is heading towards. A problem present is the lack of proper ability to verify news and information spread on social media, practically the most significant medium in which humans interact virtually. Human-led organizations tasked with verifying information spread on social media are overwhelmed by the sheer quantity of information present on these platforms – especially as nowadays, many ill-intended articles are automated (created by programmed bots).

To fight fire with fire, if there is a way to automate the process of writing manipulative articles, there should also be a way of automatically detecting which articles are properly and purposefully written. It is essential to understand which have the purpose of spreading a certain kind of information, valuing sincerity to the detriment of manipulative attempts – and implicitly telling the truth from the lies.

Natural Language Processing has a vital role to play in most automated language tasks. The development of NLP throughout history can be split into three major waves: rationalism, empiricism, and deep learning. The rationalist approach called for manual rulemaking to integrate information into the NLP structure in the first wave, based on the premise that language knowledge is set in advance by generic inheritance in the human mind. The empirical method of the second wave concluded that rich sensory feedback and measurable language data were sufficient for the mind to learn the intricate structure of language. As a consequence, probabilistic models for discovering patterns in broad language corpora have been created. The third wave, deep learning, makes use of hierarchical nonlinear structures, also known as neural networks, that are influenced by nature [1]. These networks aim to learn representations from language data in such a way that simulates the human mind.

In this paper, we will explore the task of Truth Detection, along with closely related issues and applications. We will also discuss the main innovations brought to the domain of Natural Language Processing in the last few years, which have allowed for better results in all areas of this domain.

2. Related Tasks

2.1. Fake News Detection

The task of fake news detection is concerned with assessing the truthfulness of claims in the news. The impact if such a task were successfully approached would be huge. It would essentially combat the spread of misinformation or attempts at manipulation, stopping issues from extrapolating to the division of countries into political extremist groups, leading to violence, hate, and overall inferior society. It would also offer the community the chance to properly inform themselves, giving rise to a more educated and unified nation that can make decisions based on the same ground truth.

Usually, teams of humans review news outlets, articles, and publications, cast a vote, and manually classify the respective paper as to whether it is fake news or not. Unfortunately, the process of spreading fake news happens at a much faster rate than teams of humans could ever surveil, as mentioned in the introduction, giving birth to an imbalance in our society that impacts all areas of our daily lives.

The sheer amount of information generates immense competition for people's attention. "Information consumes the attention of its recipients" [3]. Modern technologies amplify cognitive biases in a harmful way through intelligent algorithms present in search engines and social media [3]. Unfortunately, cognitive biases are products of our evolutionary past meant to protect us and help us make sense of our surroundings. As a result, it means that our ability to exploit the weaknesses of the mind surpasses our ability to evolve mentally [3].

From a technical standpoint, the issue of fake news classification is usually formulated as a binary classification problem [4]. Unfortunately, that is too a simplistic approach, as some articles can be real or fake to a certain extent. A slightly more appropriate solution is to enable multi-class classification, based on degrees of truthfulness.

2.2. Rumor Detection

Rumor detection remains the most indeterminate task amongst the ones enumerated throughout the thesis, partly due to the struggle of defining the concept. The majority of literature defines rumors as "unverified and instrumentally relevant information statements in circulation" [5]. Somewhat consistent with the previous definition, the Oxford English Dictionary defines a rumor as "a currently circulating story or report of uncertain or doubtful truth" [5].

Another difficult aspect surrounding rumors are represented by the attempt of categorizing them. One method of differentiating rumors is by veracity value (true, false, or unresolved) or its degree of credibility (high or low) [5]. An additional method of categorizing rumors is that which provides the following types: (1) rumors that lead to wishful thinking; (2) rumors that increase anxiety

or fear; (3) rumors that generate hatred [5].

Rumors can also be understood and interpreted in relation to time: new rumors and long-standing rumors which are discussed for long periods [5]. Although the names are self-implied, the former refers to rumors that emerge as breaking news. The latter regards issues that are hard to establish or require a long time to determine the truth value.

2.3. Fact-Checking

Fact-checking is the process of determining whether or not statements made by public figures, such as politicians, are true [4]. This task is often confused with fake news detection due to the fact that both relate to the truthfulness of certain claims. In general, fake news detection focuses on news events, whereas fact-checking is a larger, more general topic [4].

Journalism is characterized as a "discipline of verification," distinct from "entertainment, propaganda, fiction, or art."[6]. The concept of fact-checking and verification are often confounded, but they are distinct and yet complementary. Verification is described as a "scientific-like approach to obtaining the truth and also the right facts," which includes checking the source, date, and location of data [6]. Instead, fact-checking “addresses the claim’s logic, coherence, and context” [6]. To summarize, verification is an initial and important step in the issue of fact-checking of the considered information.

Fact-checking has also been misused in the context of fake news, where its meaning is not necessarily related to the candor of claims. Even more so, the connection to terms such as “misinformation” and “disinformation” is also worthy of taking into consideration [6]. The distinction between the two would be that the former relates to incomplete information, missing perhaps. In contrast, the latter implies an ill-intent to manipulate and alter a view upon a situation/individual.

2.4. Stance Detection

Stance detection is a task in which an author of a text is assessed as to whether he is supporting a specific entity/side. It differs from fake news detection in that the task does not check for truth, but rather consistency [4]. In general, stance detection is split into three categories: “for”, “against”, “observing” [7]. “For” pertains support in view of a claim, “Against” implies a contrary view, and “observing” is merely a repeat of particular a claim.

The problem of stance detection might be concerned with the source of information, the headline, and the overall result as far as the implicit stance goes. Additionally, elements of truth detection might be involved to gauge whether a certain claim and implicitly the support brought to it are faithful.

2.5. Sentiment Analysis

Text sentiment analysis technology mines text in order to gauge the feeling expressed, in accordance with the key purpose of the evaluation. We further distinguish the scopes of text sentiment analysis, which can be divided into three levels: for words, sentences, and chapters, respectively. Furthermore, based on the method of classification, we may encounter binary, ternary, or even multi-sentiment classification [8].

We can also distinguish three types of sentiment analysis methods, based on sentiment dictionary, on machine learning, and deep learning. Although machine learning and deep learning are used interchangeably, deep learning is a sub-domain of machine learning, which now yields greater results than any other method up until this point. More shall be discussed in the further sections, where we analyze the more recent breakthroughs in NLP [8].

3. Deceptive Language

According to [9], "We define deception as a deliberate attempt to mislead others". This entails the overall idea that summarizes the concept. To better distinguish all the nuances of ill-intended speech, we quote further: "Falsehoods communicated by people who are mistaken or self-deceived are not lies, but literal truths designed to mislead are lies" [9].

Many factors influence what we perceive as deceptive language, especially based on context – forms of deception present in casual conversation will differ from that of criminal interrogation. Even so, we may distinguish six linguistic categories present in deceptive behavior, such as quantity, immediacy, specificity, accessibility, complexity, and redundancy [10].

Quantity refers to total words in a text or mean words per sentence. Immediacy refers to the mention of tentative statements and personal pronouns. Specificity relies on spatial, temporal, and questions within a text. Accessibility addresses familiarity, meaningfulness, and concreteness of words. Complexity implies negation and sentential complexity (mean words before the verb of the main clause). Redundancy measures referential overlap in context with given information [10].

The above-mentioned concepts relating to deceptive language are human interpretations regarding the ever-growing human knowledge base regarding the process of lying. However great, the process of lie detection performed by a human will always be prone to errors due to our nature. This allows for technology to compensate for our misgivings. Further, we discuss truth detection and innovations brought to the deep-learning world which have decreased the gap between "tech" and human-level performance, as far as text analyzing is concerned.

4. Truth Detection – One Piece of the Puzzle

The task of Truth Detection is rarely ever studied by itself, but always with a greater purpose in mind. As such, it is tightly linked with other closely related language tasks, such as fake news detection or stance detection. These issues might also pertain to a component that may distinguish attempts at disinformation/manipulation.

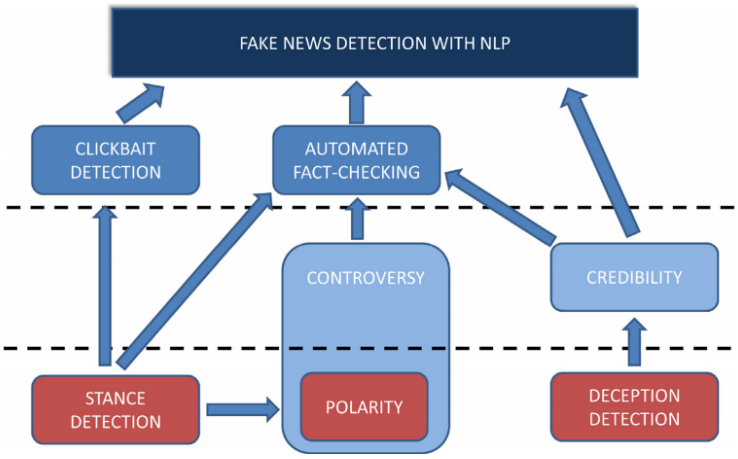

Figure 1: A diagram to show the complexity of a fake news detection application [12]

One application attempts to exploit ground-truth facts through an adversarial approach [11]. Generative Adversarial Networks (GAN) are a framework through which two networks compete, one in its attempt to deceive the other (model G –the *generative* model, creates lies), whilst the other tries to improve the ability to tell if it is being deceived (model D – the *discriminative* model, spots lies) [12]. Through competition, the two networks may improve simultaneously, creating a great 'deceiver', and an excellent 'lie spotter'. For an NLP task to indeed perform within such a network, certain preliminaries are necessary: entity recognition (object, person or organization), entity mention (a reference to an entity), event triggers (usually a verb or noun that expresses a specific event), and event argument (entity mention combined with a temporal expression, that serves as a participant to a specific role in the event mention) [11].

Another application attempts to classify facts based on varying shades of truth, regarding fake news and political fact-checking [13]. The paper also seeks to tackle the notion of "post-truth," which "refers to a distorting phenomenon in which objective facts are less effective in influencing public sentiment than appeals to emotion and personal belief," according to the authors [13]. This application is far more complex than what was previously discussed, proposing a complex and intelligent ensemble of models, each with a particular task, in order

to form a whole. In essence, the task of fake news detection is split into small, approachable segments, together with models that address that specific subtask.

Each model used was trained on specific datasets, best suited to the task at hand. Even so, we notice that many models present here use Machine Learning methods, as opposed to the newer and innovative deep-learning models. This fact alone, leaves room for significant improvements, especially in the domain of Deception Detection, which may heavily rely on the models understanding the connection between words to distinguish potentially deceptive formulations.

Table 1
Summary of the main Fake News features for each subtask as well as the number of resources and systems reported in the review, where ML=Machine Learning, DL=Deep Learning and KG=Knowledge Graphs.

Subtask	FN features	No. resources	No. systems
Deception detection	Scarcity	19	11 (9 ML/2 DL)
Stance detection	Relevant topic	5	Traditional media: 9 (3 ML/5 DL/ 1 Hybrid) Social Media: 4 (2 ML/2 DL)
Polarity	Impact	1	4
Controversy	Viralization		(2 ML/2 Graph models)
Fact checking	Relevant topic Scarcity Impact Viralization	8	13 (References: 5/KG: 4/Context: 4)
Clickbait detection	Relevant topic Impact Viralization	5	9 (5 ML/3 DL/1 Hybrid)
Credibility	Impact Scarcity Viralization	31	Traditional media: 4 Blogs: 3 Social media:11 (18 ML)

Table 1: Table describing implementational approaches for the fake-news detection application [12]

To fully understand the number of models used and the complexity of the ensemble, combined with the extent of expertise necessary to create such a tool, the figure below offers a break-down of the approaches undertaken.

5. Neural Network Mechanisms in Natural Language Processing

If algorithms can better “understand” human languages, they can also extract more information, and “offer” better predictions/solutions in regard to a certain end goal. The more they can learn, the better they can generalize.

The end goal is to construct language models that can address various linguistic tasks successfully. Ideally, we would like to have one size fits all type of model, that is able to address all types of tasks with super-human performance. In essence, to obtain better performance on our language models, we implicitly wish to devise better ways of developing them. A language model may only be as good as its building blocks allow for it to be. Innovation may come in the form of incremental improvements or conceptual breakthroughs. As may be obvious, the latter type tends to more strongly impact fields of research, and this one is no exception.

5.1. Previous Approaches

Previous attempts at solving NLP related tasks included variants of linear regression, logistic regression, or support vector machines [4], which were built to solve language classification problems. Nevertheless, they were expert systems built by scientists or the like, using intelligent feature engineering with the end purpose of obtaining the best results possible in regard to specific data (and implicitly datasets). This type of approach was closed for generalization and a transfer of results from one language-related task to another was generally impossible.

This category of systems are results of “The First AI Era”, specific to years before 2012 (even though reminisces of such approaches continued in later years as well). The disparity between "The First Era" and "The Modern Era" is that the amount of processing power needed to train a neural network on ImageNet to obtain the same accuracy has decreased by a factor of two every 16 months since 2012 [14].

5.2. Recurrent Neural Networks

A recurrent neural network is a sequence learning model that connects nodes between hidden layers and can learn sequence features dynamically [8]. Traditional neural networks are ineffective in dealing with sequence learning because they build no correlation between the beginning and end of a specific sequence. In simpler terms, RNN’s can build upon previous steps in time. As opposed to traditional neural networks, RNN’s work on an encoder-decoder architecture. Simply put, encoders will summarize all of the data from the input phrase, and decoders will use the encoder’s output to generate the appropriate output [15].

RNNs provide a great stepping stone for word prediction and machine translation.

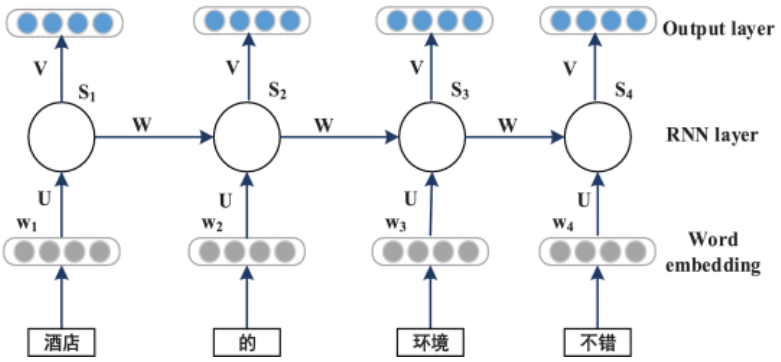

Figure 2: Machine translation using RNN [8]

In Figure 2, the sentence "The environment hotel is good" in Chinese is being translated by inputting each word sequentially into the RNN (word segmentation) and building word vectors for sentence c. Each word vector is then fed into the RNN, influencing the outcome of the next word vector prediction/translation based on our current word.

5.3. Long Short-Term Memory Model

RNNs are slow to train and also suffer from a problem called vanishing and exploding gradients. In essence, if sequences are too long, information gets lost (mainly from the beginning of sentences).

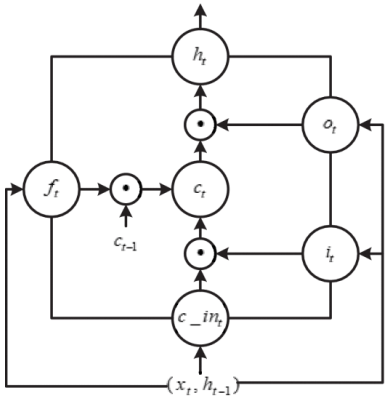

Figure 3: LSTM block Structure [8]

Long Short-Term Memory Networks (LSTMNs) were created to address these problems with RNNs. These work by enabling a previously hidden state to pass to the present cell while skipping the majority of the processing that the current cell would do [15]. This particular mechanism allows for more selectiveness in

retaining or discarding information. It implicitly instils neurons with memory which can be leveraged on a per need basis. This allows for better information capture over longer sequences. Concretely, LSTM introduces the Input Gate i, Output Gate o, Forget gate f, and Memory cell [8]. The Forget fate determines the information to forget in the Memory cell.

While RNNs are considered slow to train, it turns out LSTMs take even longer to train, which can be explained by more computation power needed per unit. Unfortunately, since each word is passed sequentially into the architecture, it cannot take advantage of today's GPU parallel processing.

5.4. Attention Mechanisms

Attention mechanisms were introduced to address certain limitations of RNNs and LSTMs. In the process of decoding, the decoder's hidden state is multiplied by the encoder's hidden state so as to produce a distribution of relevance or importance. At an implementational level, this implies the leverage of a context vector that contains the weighted sum of all the hidden states [16]. In simpler terms, the context vector influences aspects of the output, as to offer some parts of a sentence more importance in the detriment of others. It is called attention because it imitates the human process of focusing on a certain part of the text.

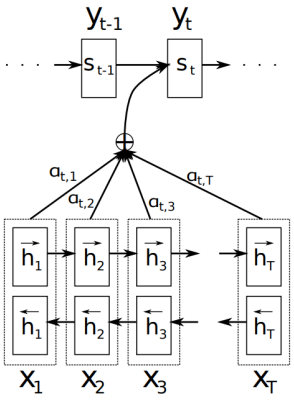

Figure 4: Attention block structure [16]

While the attention mechanism mitigated some of RNN's shortcomings, words were still fed to the network in a sequential fashion, preventing these architectures from reaping the benefits of today's hardware's parallel processing capabilities [15].

5.5. Transformers

With the discovery of Transformers, RNN's were inherently replaced altogether. It is important to note that Transformers are a "conceptual" breakthrough,

which usually drives innovation in most deep learning domains, and NLP is no different. RNNs are also an example. LSTMs are only improvements brought to an existing concept, but these improvements are not enough to bring out new and impressive results.

The Transformer architecture employs both an encoder and a decoder, but no RNNs [17]. The main distinction from previous architectures is that now the encoder's input can be passed as a full sentence (as opposed to one word at a time for RNNs). Analogously, the decoder's inputs consist of the entire sentence (shifted right). We sequentially pass all of the words to the sentence and determine the word embeddings [15].

Transformers are often regarded as the discovery of the decade in terms of NLP, so they are worthy of extra attention (no pun intended). To begin, we feed all of the sequences' word embeddings into the network inputs. Embeddings are vector representations of words that tie in words with similar meanings (specifically, similar words in a defined embedding/vector space) [15].

The positional encoding layers are used to retain the position of a specific word, with different word placements implying different meanings [17]. Self-attention, or how each word in a series is related to other words in the same sequence, is the focus of the multi-headed attention block. [17]. An attention vector is created within the attention block to reflect self-attention, preserving associations between the word in a sentence [17]. Relationships are determined by the scaled dot-product of two vectors, and the result indicates whether they are closely related or not.

The multi-headed attention block is called as such because a weighted average of attention vectors is used to determine the final result [17]. Each attention vector is independent of the other, and this allows for the use of parallelization (important for the use of GPUs and computing equipment). The Feed-Forward Network (FFN) is used to allow the output of one type of block to be used as the input for another [17]. After each attention block and FFN block, Addition and Normalization Layers are applied to normalize. They also enable backpropagation via residual connections [17]. The decoder block functions in a similar fashion.

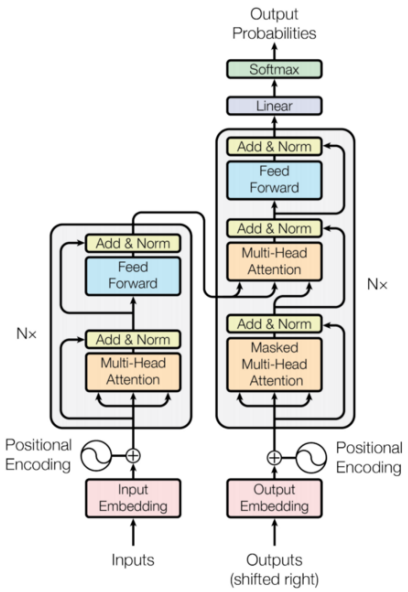

Figure 5: Transformer block structure [18]

The decoder's self-attention block generates the attention vectors for the target sequence after receiving the target sequence of words encoded in an embedding and the positional encodings. Finally, a linear layer and a SoftMax function are used to produce the probability distribution of all subsequent terms, as well as the term with the highest probability [15].

5.6. Reformers

A Transformer's limitation is precisely what gives it powers – the attention module. Using the attention module, it compares and contrasts all possible pairs of words to better understand the connections between them. If we were to assess X words, we would have X2 operations. If that X was 100.000, then we would have 10 billion pairs [18].

Another issue is the common practice of saving each model layer's output. The memory needed for storing the output of neural network layers for long context windows may imply sizes of terabytes in models with thousands of layers [18]. The practical implication of such limitations is that Transformer models with several layers can only be utilized to create only a few lines text as output, depending on the specific use-case.

Reformers [19] are a Transformer model that addresses both, the attention, and the memory problem, respectively. It was conceived in such a way in which it manages one million-word context windows using only 16GB of memory [18]. The Reformer employs locality-sensitive hashing (LSH) to simplify the task of

attending lengthy sequences and reversible residual layers to make better use of available memory [18][19].

5.7. Performers

Several fast and more space-efficient proxies, such as memory caching approaches, have been proposed for situations where long-range attention is required; however sparse attention is a far more typical solution. Sparse attention decreases attention mechanism calculation time and memory needs by computing only a subset of similarity scores from a sequence rather than all possible couples, resulting in a sparse matrix rather than a complete matrix [20]. Sparse-matrix attention is used in Reformers [19].

Even so, there are a number of drawbacks to sparse-attention strategies, such as: They need efficient sparse-matrix multiplication operations, seldom give robust theoretical assurances for their representation power, are particularly tuned for Transformer models and generative pre-training, and stack additional attention layers to compensate for sparse representations. This poses significant obstacles in the utilization in combination with other-pretrained models, thus necessitating retraining, which in turn implies a larger energy consumption [20].

The Performer [21] was introduced to address the shortcomings mentioned above. It is based upon a Transformer design, but with linearly scaling attention mechanisms, allowing for quicker training whilst also being able to process longer lengths, as required by certain image and text datasets [20][21]. It employs a highly efficient (linear) generalized attention framework that allows for the use of various attention mechanisms, which in turn produces scalable, low-variance, and unbiased attention mechanism estimates [20]. Performers achieve high accuracy guarantees while maintaining linear space and time complexity [20][21].

6. Natural Language Processing Language Models

It is important to note that as language models advance, results obtained on datasets concerning certain tasks also improve. It would seem that they are tightly knit together, and the better the language model, the better results can be obtained in specific tasks. These advancements are driven by new ways of constructing language models, optimizations, but also access to more computing power in accordance with Moore's Law.

A "good" language model will generally perform well in most language tasks, only to be superseded by specialized models (in regard to that particular task). This is an important mention because before the era of deep learning, algorithms were more statistical and could only be applied towards a single end goal in mind. Breakthroughs such as the ones discussed above have offered something very special – the ability to generalize.

Below, we address some of the more impactful language models from the past few years, and discuss how they manage to leverage Transformers layers to obtain better performances with less computational power.

6.1. BERT - Bidirectional Encoder Representations from Transformers

BERT is a model built by Google AI, innovating by applying the bidirectional training of Transformer to language modelling (LM) [22]. As mentioned above, Transformers use encoders to read text input and decoders to produce predictions for a certain task. Only the encoder mechanism is needed because BERT's goal is to construct a language model. While the Transformer encoder reads the complete string of words at once, meanwhile directional variants (such as RNNs) read text input in a sequential manner [15]. Although this behavior is labelled as "bidirectional," it is really more "non-directional." This enables a model to understand the meaning of a term by looking at its surroundings [23].

Some interesting particularities of BERT are that 15% of sequences fed into BERT are replaced with a MASK token [24]. Based on the context of other, non-masked keywords in the sequence, the model attempts to forecast the original value of the masked words. [23]. Masked LM (MLM) is the name of this procedure. Another distinguishing characteristic of BERT's training process is that it learns to predict whether the second sentence in a pair is the next sentence in the original text by receiving pairs of sentences as input [23]. During instruction, half of the inputs are a pair, with the second sentence being the following input sentence [23]. Intuitively, this process is called Next Sentence Prediction (NSP) [23].

6.2. RoBERTa – Robustly Optimized BERT

RoBERTa is a robustly optimized variant of BERT, produced at Facebook, with 1000% more data and compute power [25]. RoBERTa removes the Next Sequence Prediction (NSP) task from BERT's pre-training and replaces it with dynamic masking, which switches the masked token during training epochs [25]. The training technique was also found to be more effective with larger training batches.

6.3. ALBERT – A Lite BERT

ALBERT [26] is a language model released by Google, built upon BERT, which successfully leveraged the Transformer technology for language tasks such as machine translation or question answering. These state-of-the-art models include hundreds of millions or billions of parameters, commonly referred to as weights and biases. The amount of memory required to store all this information brings about certain limitations for specialized hardware such as GPUs (Graphical Processing Units) and TPUs (Tensor Processing Units) [27]. There are 3 innovative aspects that stand out in ALBERT, as opposed to BERT: factorized embedding parametrization, cross-layer parameter sharing, and introducing an inter-sentence coherence loss, respectively [28].

Firstly, researchers were able to increase the hidden layer size without considerably increasing the vocabulary embedding parameter size by projecting one-hot encoded vectors into a lower-dimensional embedding space [26][28]. Secondly, researchers opted to share all parameters across layers to avoid the direct correlation between network depth and parameter growth. As such, the large ALBERT model has around 18 times fewer parameters [28]. Lastly, as per the improvements brought to RoBERTa, Google decided to remove the NSP (Next Sentence Prediction) step initially proposed in BERT. Instead, to quantify inter-sentence coherence, they used a sentence-order prediction (SOP) loss, which allowed the model to perform better in multi-sentence encoding tasks [26][28].

6.4. T5 – Text-to-Text Transfer Transformer

T5 [29] is yet another Transformer based neural network developed by Google, which allows for the translation of any language task to a text-to-text format, managing to discover the use-cases of transfer learning. Similar to other models, it contains a vast number of parameters and layers, which is trained on a blend of huge amounts of unlabeled text and data from common NLP tasks, then independently fine-tuning the model for any of the specific goals in mind. It manages to set state of the art for a wide plethora of renowned linguistic tasks in English, such as text classification or summarization [30].

The model introduces a framework for shared Text-to-Text, which proposes that all NLP jobs be reshaped into a specific format that leverages character strings as input and output. This contrasts the approach previously used for BERT,

where the model could only output a sequence found within the input or a class label [30].

6.5. GPT-3 – Generative Pre-trained Transformer

GPT-3 is the latest language model from the OpenAI team [31]. The main novelty of this model is the sheer scale and number of parameters. This comes as a result of being trained on most of Wikipedia and available books corpus, which is in essence more data than any model has ever been trained on. For scale, take a look at Figure 7 [14], which presents the number of parameters for each of the more recent models, in chronological order of appearance.

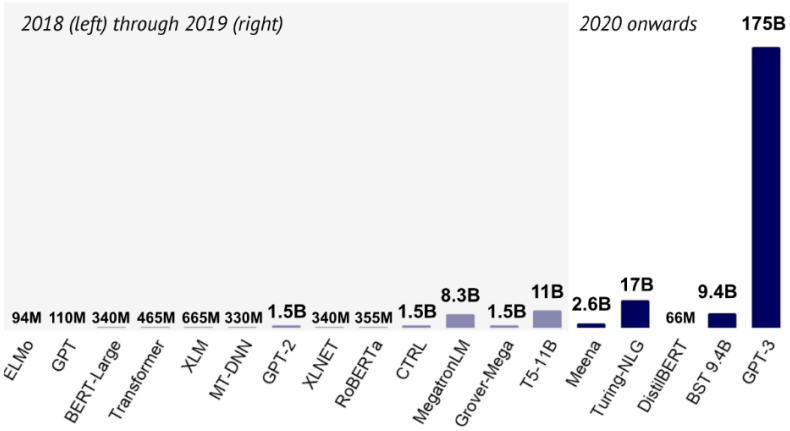

Figure 6: Bar chart presenting the number of parameters in leading language models [14]

The model can generate poetry, write role-playing adventures, or create simple apps. In terms of linguistic activities, it is capable of zero-shot or few-shot learning [31]. The following is an example of one-shot learning [32]:

(1) Alice was friends with Bob. Alice went to visit her friend _____. → Bob

(2) George bought some baseball equipment, a ball, a glove, and a _____. → bat

In essence, the model is given zero/one/more examples, before it is asked to predict an answer. To build upon such exceptional behavior, GPT-3 can also translate between languages, beating previous state-of-the-art models in some language pairs. It can also do a fine job with reading comprehension activities, comparable to the state-of-the-art a few years ago [31].

GPT-3 has memorized many facts about the world as a result of its extensive text training. It outperforms previous state-of-the-art in this domain when answering trivia questions. From a natural language query, it can also generate working JavaScript code [31] [32].

7. Benchmarks, Datasets and Results for Deep Learning Models

General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems [33]. There are nine sentence or sentence pair language comprehension tasks based on pre-existing datasets that span a wide variety of sizes, genres, and degrees of difficulty [34]. It also includes a diagnostic dataset for evaluating and analyzing model output in natural language for a variety of linguistic phenomena [33]. A public leaderboard is also available for monitoring results. Since the GLUE benchmark is model-agnostic, any system that can process sentences and sentence pairs can participate [33].

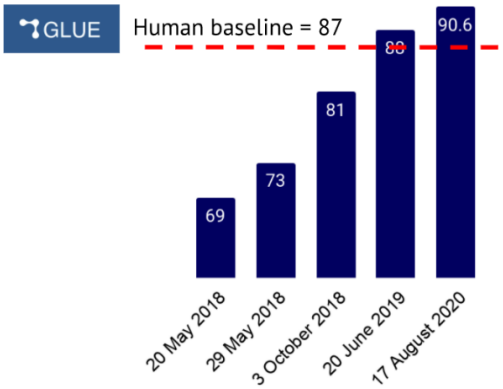

Figure 7: State-of-the-art language model performance evolution [33]

As the GLUE benchmark has recently been surpassed, a new, improved, and more challenging benchmark took its place: **SuperGLUE** [34]. Now, even this new benchmark is close to being overtaken.

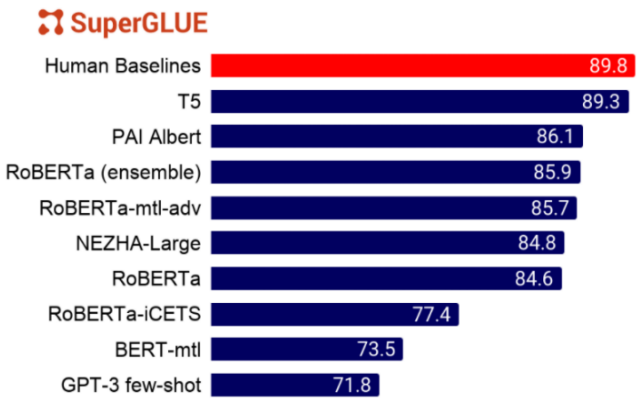

Figure 8: State-of-the-art models performance on SuperGLUE benchmark [35]

Unfortunately, huge computational, economic, and environmental costs are rapidly soaring for incrementally smaller improvements in model efficiency. Without major conceptual breakthroughs (such as Transformers), dropping the error rate of state-of-the-art machine learning models may end up costing hundreds of billions of dollars [14].

		Polynomial			Exponential		
Benchmark	**Error rate**	**Computation Required (Gflops)**	**Environmental Cost (CO_2)**	**Economic Cost ($)**	**Computation Required (Gflops)**	**Environmental Cost (CO_2)**	**Economic Cost ($)**
ImageNet	Today: 11.5%	10^{14}	10^{6}	10^{6}	10^{14}	10^{6}	10^{6}
	Target 1: 5%	10^{19}	10^{10}	10^{11}	10^{27}	10^{19}	10^{19}
	Target 2: 1%	10^{28}	10^{20}	10^{20}	10^{120}	10^{112}	10^{112}
MS COCO	Today: 46.7%	10^{14}	10^{6}	10^{6}	10^{15}	10^{7}	10^{7}
	Target 1: 30%	10^{23}	10^{14}	10^{15}	10^{29}	10^{21}	10^{21}
	Target 2: 10%	10^{44}	10^{36}	10^{36}	10^{107}	10^{99}	10^{99}
SQuAD 1.1	Today: 4.621%	10^{13}	10^{4}	10^{5}	10^{13}	10^{5}	10^{5}
	Target 1: 2%	10^{15}	10^{7}	10^{7}	10^{23}	10^{15}	10^{15}
	Target 2: 1%	10^{18}	10^{10}	10^{10}	10^{40}	10^{32}	10^{32}
CoLLN 2003	Today: 6.5%	10^{13}	10^{5}	10^{5}	10^{13}	10^{5}	10^{5}
	Target 1: 2%	10^{43}	10^{35}	10^{35}	10^{82}	10^{73}	10^{74}
	Target 2: 1%	10^{61}	10^{53}	10^{53}	10^{181}	10^{173}	10^{173}
WMT 2014 (EN-FR)	Today: 54.4%	10^{12}	10^{4}	10^{4}	10^{12}	10^{4}	10^{4}
	Target 1: 30%	10^{23}	10^{15}	10^{15}	10^{30}	10^{22}	10^{22}
	Target 2: 10%	10^{43}	10^{35}	10^{35}	10^{107}	10^{99}	10^{100}

Table 2: Analysis of implications in driving error rates down [35]

ImageNet and MS COCO are machine vision benchmarks. What is of interest to the current discussion are the SQuAD 1.1, CoNLL 2003, and WMT 2014 (EN-FR) benchmarks. The Stanford Question Answering Dataset (SQuAD) is a reading comprehension datasets a reading comprehension dataset made up of questions posed by crowd workers on a collection of Wikipedia articles, with each question's answer being a fragment of text from the relevant reading passage

[36]. The questions can also be unanswerable.

Conference on Computational Natural Language Learning (CoNLL) is a yearly conference [37]. CoNLL-2003 is a shared task that concerns language-independent named entity recognition [37]. t focuses on four sorts of named entities: people, places, organizations, and names of other entities that don't fit into the first three categories [38].

Workshop on Statistical Machine Translation (WMT) 2014, as the name implies, is a dataset that evaluates translations between language pairs. The table above addresses the performance obtained on the English-French dataset.

From the benchmarks mentioned above, SQuAD is exciting because of how a truth detection problem can be formulated - as a question. Although that is not the purpose of the SQuAD dataset, delving deeper into state-of-the-art performance for this particular task is worthy of more attention. In the table above, SQuAD 1.1 was used as a benchmark. Currently, a newer version of the SQuAD benchmark is available, SQuAD 2.0 [36].

Below, the performance of BERT in regard to the SQuAD 2.0 dataset (including multiple variants), applied on the task of question answering is shown [36].

ID	Architecture on Top of BERT	F1	EM
1	BERT-base PyTorch Implementation	76.70	73.85
2	BERT-base Tensorflow Implementation	76.07	72.80
3	GRU Encoder + Self-attention + GRU Decoder + BERT-SQUAD-Out	73.59	69.87
4	BiLSTM Encoder + BiDAF-Out	76.37	73.05
5	CNN Encoder +Self-attention +BERT-SQUAD-Out	76.49	73.23
6	CNN Encoder + BERT-SQUAD-Out	76.56	73.64
7	GRU Encoder + GRU Decoder + BERT-SQUAD-Out	76.85	73.77
8	CNN Encoder + BiLSTM Decoder + Highway + BERT-SQUAD-Out	77.07	73.87
9	BiLSTM Encoder + Highway + BERT-SQUAD-Out	77.41	74.32
10	BiLSTM Encoder + Highway + BiLSTM Decoder + BERT-SQUAD-Out	77.66	74.87
11	BiLSTM Encoder + BiLSTM Decoder + Highway + BERT-SQUAD-Out	**77.96**	**74.98**
12	Ensemble of 11 and 7	**78.35**	**75.60**
13	Ensemble of 11 and 7 and BERT large case model	**79.443**	**76.966**

Table 3: F1 and EM scores for different model architectures [36]

The classic F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall. The formula below is used to calculate the F1 score, where the following notation is used: TP (True Positives), FP (False Positives), FN (False Negatives).

$$F1 = \frac{2}{recall^{-1} + \ precision^{-1}} = \frac{2\,(precision \times recall)}{precision + \ recall} = \frac{TP}{TP + \frac{1}{2}\,(FP + FN)}$$

The percentage of predictions that match every one of the ground truth answers exactly is called exact match (EM).

8. Transfer Learning

Transfer learning gets its name from the fact that it mimics human behavior: if you know how to ride a bike, you can apply some of that experience to learning to ride a motorcycle. In essence, transfer learning is the concept of overcoming independent learning paradigms and applying information gained from one task to solve problems that are connected to it. This is particularly true for modern machine learning solutions, and it is a significant improvement over the First AI Period, when isolated algorithms were only used for specific tasks [39].

Concretely, we can build our model upon existing Deep Learning models – even those such as BERT or RoBERTa. This implies that even from the start, we have a model on our hands that can understand the English language and may only need fine-tuning – a huge improvement from us having to train the model from scratch. This means that even with a "small" dataset, we can train an existing variant of the BERT model on our specific task and find out that it offers great results, despite the limitation of our own, personal task-related data quantity. Most great applications leverage this concept, iterating upon existing models, without having to reinvent the wheel each time there is a wish to perform language-related tasks. Instead, extremely large models such as GPT-3, which are pre-trained on vast language corpora, invalidate the need for task-specific fine-tuning on a specific dataset.

9. Programming Languages and Frameworks for Deep Learning

Insofar as popularity is concerned, there are two main programming languages used in the Data Science community: R and Python. The focus on the R programming language is statistical analysis, whereas Python is more heavily used in machine learning and artificial intelligence. What makes Python a more popular choice is the intuitive way of programming in concordance with the massive open-source community backing it up, which translates to faster adoption throughout the programming community.

Community is important, and Python is the result of a cumulative effort from various entities and individuals. As such, if a programming community massively contributes to a product, chances are the end-product ends up being qualitative and well-developed, having multiple programmers from around the globe with varying levels of experience contribute towards the same end-goal. This is how Python 3 was born.

If a large group has a vision, which aligns with the interests of firms, opportunities for investments arise. Concretely, with the growing interest of researchers, academia, programmers, and corporations towards innovating the domain of machine learning and artificial intelligence, corporations brought their own contributions towards achieving this similar goal, creating various machine learning frameworks. Examples are PyTorch from Facebook and TensorFlow from Google, to name a few.

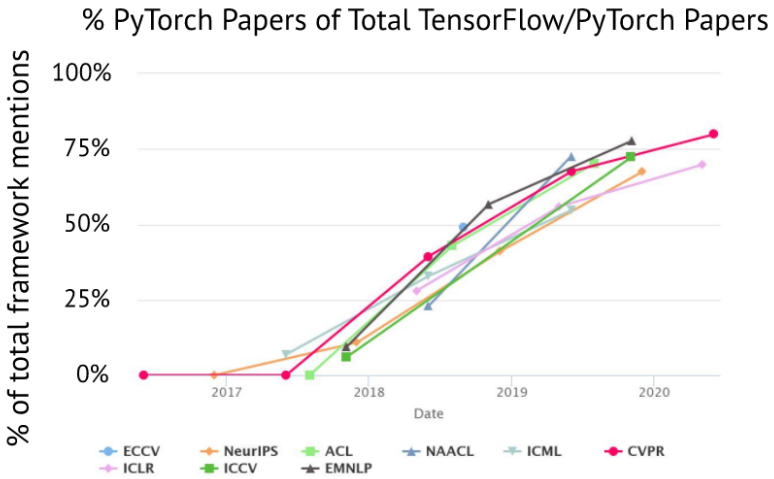

Figure 9: Graph showing the evolution of PyTorch mentions in scientific papers [35]

Both frameworks have a similar purpose: to facilitate the implementation

of powerful machine learning algorithms. Instead, they differ conceptually: TensorFlow operates under the "static graph" definition, whereas PyTorch has an underlying "dynamic graph" approach. In essence, this means that with a static graph, every component of the code must be known beforehand, whereas, in a dynamic graph, nodes can be executed on the go. The latter allows for much faster experimentation, with a detriment in the speed of execution.

75% of conference papers that mention the system they use mention PyTorch, but not TensorFlow [14]. In 2018, 55% of the 161 authors who published more TensorFlow papers than PyTorch papers switched to PyTorch, while 15% switched the other way [14].

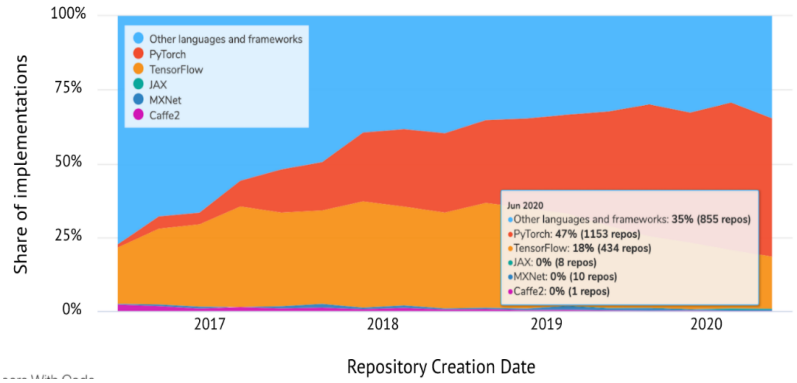

Figure 10: Machine learning framework popularity (usage) on GitHub [35]

On GitHub, PyTorch is also more prominent than TensorFlow in terms of paper implementation. PyTorch is used in 47% of these implementations, while TensorFlow is used in just 17% [14]. JAX is a Google application that is more math-friendly and is preferred for work that does not include convolutional models or transformers [14].

10. Formulating the Scope of Research

The task of Truth Detection is complex and multi-faceted. There are many ways of approaching this particular task on a conceptual level, but also practically. Even so, starting from first principles, we come up with the following questions that should hold up irrespective of outlooks:

- How does deceptive language vary from truthful language?
- How can we use current technologies to automatically classify affirmations or text?

The first question wishes to uncover the human-like aspect of formulating sentences, offering and withholding information, conveying information, and use of certain linguistic aspects. This is what we, as humans, come into contact with most often, but rarely question the different ways a certain piece of information may have been conveyed differently to empower various perspectives. This aspect is what linguists, detectives or other entities usually analyze in correlation to other factors (e.g. body language), to determine whether a certain individual is lying or not.

The second question wishes to test the real-life use-cases of state-of-the-art technologies in the broad field of Natural Language Processing, Machine Learning and implicitly Deep Learning, to gauge various model's ability to leverage certain distinguishing features of the English language with the purpose of offering a trust-worthy verdict upon the veracity of affirmations.

11. Dataset - Politifact

PolitiFact [40] is a non-profit organization that reports on the veracity of comments made by elected officials, lobbyists, candidates and their staff, and various groups of interest which are active in American politics. Their staff consists of internationally renowned reporters, academicians, and individuals with distinct backgrounds and relevant personal realizations, specifically in the field of media and communications, which make them worthy of assessing various claims [41]. Their verdicts are given based on a "Truth-O-Meter" scale, which can be one of the following categories: Pants on Fire!, False, Mostly False, Half True, Mostly True, True [40].

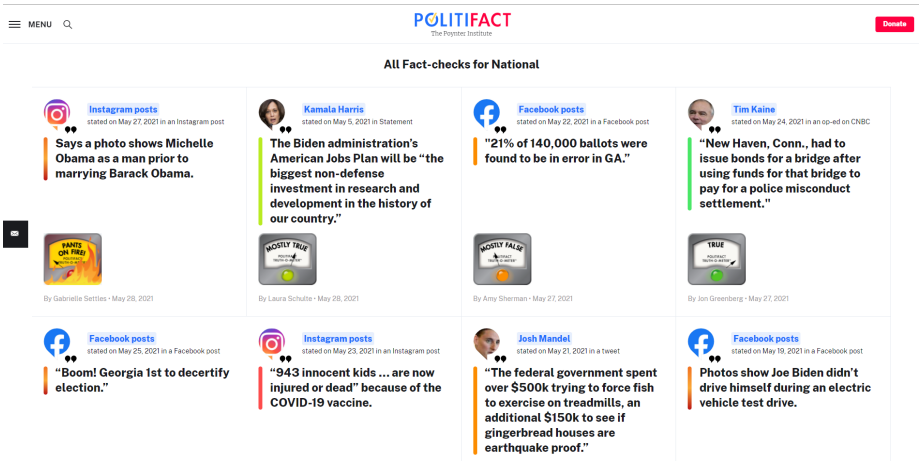

Figure 11: PolitiFact.com Landing Page [40]

With the information present on this website, we can devise a real-life case study based on affirmations made by politicians and various public entities. These affirmations are of varying complexity, with different levels of attributed truthfulness. There are thousands of such entries (samples) which are verified by multiple trust-worthy sources within the PolitiFact staff.

All the above-mentioned reasons make this particular website an extremely viable candidate for building a database with the information prevalent here. In further sections, we will delve into more details as to how we create a database from this website, and how we leverage the extracted information which will be ultimately used for the purpose of truth detection, in an attempt to answer our research questions.

12. Methodology and Approach

The following subsections aim to offer details behind the rationale regarding most of the implementations made. From a practical standpoint, the entire project leverages Python 3 frameworks and packages which shall be mentioned where considered necessary. The full implementation is available at the following GitHub repository: [42].

12.1. Parsing PolitiFact Website

In order to construct potential statistics or train Machine Learning models, we will require for the information to be in a structured format, such as text files or comma-separated values (CSV) files. This implies the extraction of information from the website – an operation commonly called site parsing or web scraping. To complete our task we leverage a highly popular package called Beautiful Soup [43]. The table below illustrates a sample of our newly created dataset which contains a total of more than 14.000 entries. The generated file is of a CSV format.

statement	Sen. Kamala Harris is "supporting the animals of MS-13."
source	Donald Trump
link	https://www.politifact.com/california/statements/2018/jul/03/donald-trump/pants-fire-white-house-claim-sen-harris-supporting/
veracity	Pants on Fire!

Table 4: Dataset entry sample

After parsing, we also obtain a set of entries belonging to classes that bring no value to our research cause. These classes are: Full Flop, Half Flip, No Flip, Half-True. After removing the aforementioned entries, we are left with more than 11.000 entries (samples). The pie-chart below presents the distribution of remaining samples, based on veracity.

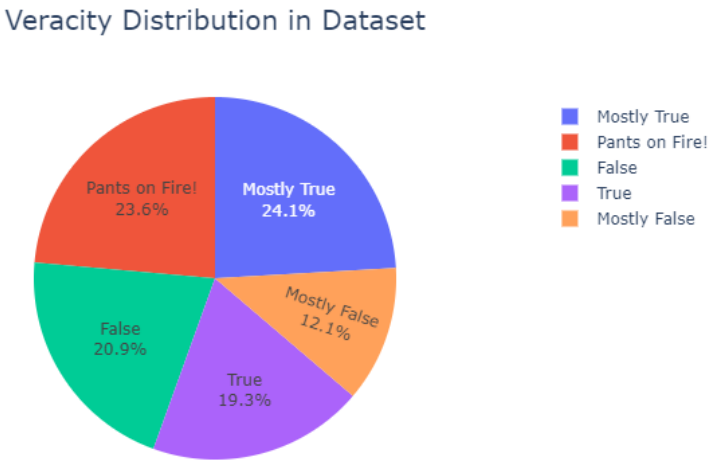

Figure 12: Veracity distribution in the clean dataset

12.2. Creating Multiple Dataset Variants

We intend to simply our analysis by reducing the granularity for veracity from six classes to just two: Truths and Lies. The decision to create multiple variants of the dataset is based upon the strictness of veracity and quality of verdict. With this in mind, we create two variants, which bear the following names, and contain entries from the various classes.

File name (variant)	**Truths Classes**	**Lies Classes**	**No. of samples**
politifact_clean_binarized.csv	True, Mostly True	Mostly Lie, Lie, Pants on Fire!	11188
politifact_strict_binarized.csv	True	Lie, Pants on Fire!	6158

Table 5: Dataset variants

Entries for both variants now bear a similar structure, but with veracity strings having been replaced by Boolean values.

statement	Sen. Kamala Harris is "supporting the animals of MS-13."
source	Donald Trump
link	https://www.politifact.com/california/statements/2018/jul/03/donald-trump/pants-fire-white-house-claim-sen-harris-supporting/
veracity	0

Table 6: Dataset entry in binarized form

The pie-charts below present the veracity distributions after the process of binarization.

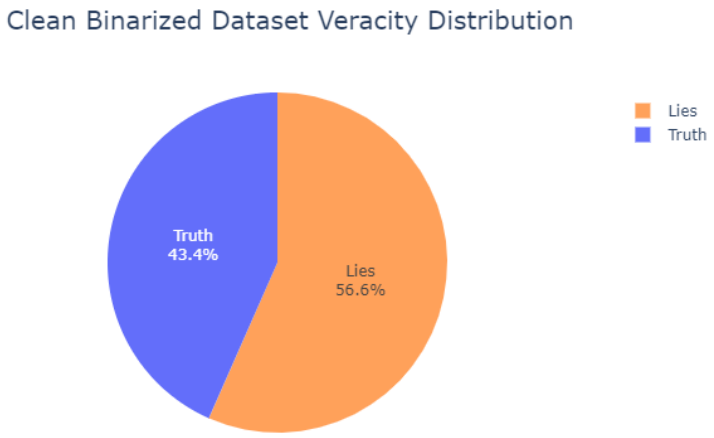

Figure 13: Veracity distribution in the clean dataset

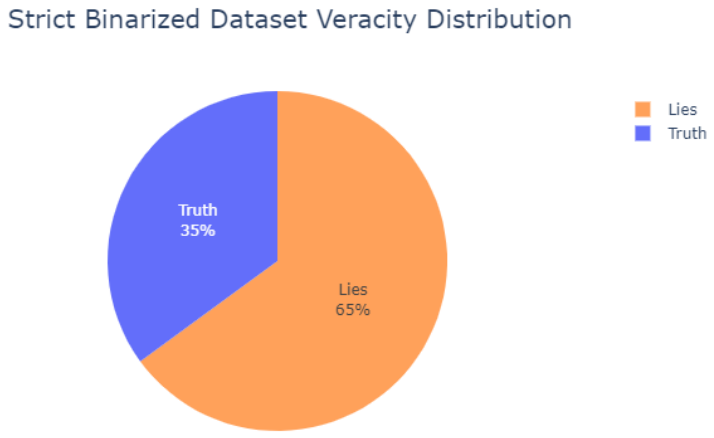

Figure 14: Strict binarized dataset veracity distribution

We notice that upon imposing stricter standards for the veracity attribute, we are left with a more imbalanced dataset. This is worthy of mention, as in later sections, we present results that are affected by the quality of data remaining in the dataset.

12.3. Preprocessing

The pre-processing stage is divided into multiple steps, such as: lowercasing, punctuation removal, tokenization, and lemmatization. These steps are necessary for performing feature engineering and training on machine learning models but may prove to be redundant or even unfavorable when dealing with deep learning models. The operations below are performed using built-in Python functions and the spaCy library [44].

Lowercasing implies that we take all words and replace any capital letters with their lowercase variant. Although it remains good practice, the main goal is to remove redundancy – we wish for "hello", "Hello", "heLLo" to be represented by a single variant, which helps reduce the storage needed to run algorithms.

Punctuation removal is exactly what it sounds like but may vary based on the content we are dealing with. Even though we are facing a rather straight-forward case where most samples are just affirmations, in the context of analyzing tweets, for example, we might want to include emojis expressed with the help of punctuation (e.g. :)). Our scope does not include any sentiment analysis, so we may safely remove punctuation.

Tokenization is a process that aims to split a larger text into smaller sub-units called tokens. Tokens can be words, characters, or parts of words. In our case, we decide to split on the space delimiter, meaning that we essentially split upon words – thus, our tokens are words. Tokens may later be used by various NLP algorithms or models to construct a vocabulary (which is a set of all words in a particular text). As mentioned before, we aim to remove redundancy with all these pre-processing steps, so as to occupy as little memory as possible for the benefit of efficiency.

Lemmatization and stemming both aim to reduce inflectional or derivative forms of a certain word [45]. An example might be:

Loving, loved -> love

Even though they wish to accomplish the same objective, they differ in approach. Stemming is a heuristic procedure that cuts off the ends of words with no fixed methodology, and it frequently includes the removal of derivational affixes [45]. Lemmatization typically uses a vocabulary and a morphological study of words, with the goal of removing only inflectional ends and returning the base or dictionary form of a word, known as the lemma [45].

For the reasons mentioned above, we opt for the lemmatization step in the detriment of stemming. The table below presents a sample that undergoes pre-processing and how it has changed.

Sentence before pre-processing	Sentence after pre-processing
"@username123 I'm angry I couldn't find a sample sentence to lemmatize so I had to write this!! :("	'i angry i find sample sentence lemmatize i write'

Table 7: Pre-processing example

12.4. Feature Engineering

Feature Engineering is a need birthed by the wish to obtain data in a certain form or extract data with a certain characteristic. It is a broad term that refers to preparing information to match a particular shape to maximize the compatibility with machine learning algorithms. Pre-processing and feature engineering play a quintessential role in obtaining good results with classical or statistical machine learning algorithms. This particular step is usually where data scientists spend most of their time.

Given our dataset, we wish to obtain general information about our entries, with NLP requirements in mind. As such, per sample, we would like to obtain the number of letters, number of words, average word length, and readability score. The feature engineering stage, similar to the pre-processing stage, is powered by the spaCy library.

To calculate a readability score, we opt for Gunning's Fog Index (FOG) readability formula, named after American textbook publisher Robert Gunning which observed that most high school graduates had difficulties reading [46]. As such, the FOG index varies from 6 (sixth grade) to 17 (college graduate), based on reading level by grade, an increasing readability score implying a higher text complexity [46].

One operation performed that was initially considered useful but ended up not providing too much value was the extraction of all calendar dates from links and introducing them as a new column.

The table below illustrates a single entry in our database after performing general-purpose feature engineering.

Statement	Says a progressive income tax proposal from Democratic governor candidate J.B. Pritzker "is wholly regressive."
Source	Evelyn Sanguinetti
Veracity	0
Date	02-07-18
Lemmas	progressive income tax proposal democratic governor candidate pritzker wholly regressive
Number of letters	111

Number of words	15
Average word length	6.5
Readability Score	12.7

Table 8: Example of data sample after performing general feature engineering

12.5. Part of Speech Tagging (General Tags)

It is of interest for us to see how various parts of speech are used throughout our samples, based on context and definition. We leverage the full capabilities of spaCy with the purpose of parsing affirmations and "tagging" each word with its corresponding part of speech.

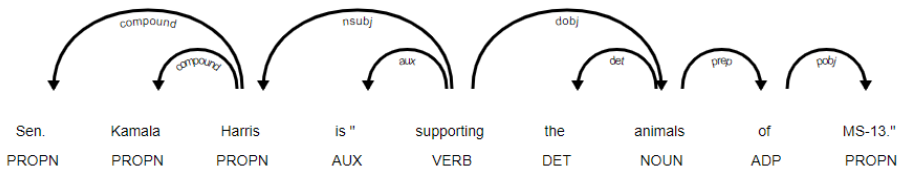

Figure 15: Visualizing Part of Speech (POS) tagging example from the dataset

SpaCy offers the possibility of POS tagging based on two types of tags: general or specific. The table below presents all the corresponding general tags, according to the documentation [47].

POS	DESCRIPTION	EXAMPLES
ADJ	adjective	big, old, green, incomprehensible, first
ADP	adposition	in, to, during
ADV	adverb	very, tomorrow, down, where, there
AUX	auxiliary	is, has (done), will (do), should (do)
CONJ	conjunction	and, or, but
CCONJ	coordinating conjunction	and, or, but
DET	determiner	a, an, the
INTJ	interjection	psst, ouch, bravo, hello
NOUN	noun	girl, cat, tree, air, beauty
NUM	numeral	1, 2017, one, seventy-seven, IV, MMXIV
PART	particle	s, not,

PRON	pronoun	I, you, he, she, myself, themselves, somebody
PROPN	proper noun	Mary, John, London, NATO, HBO
PUNCT	punctuation	., (,), ?
SCONJ	subordinating conjunction	if, while, that
SYM	symbol	$, %, §, ©, +, ?, ×, ÷, =, :), ??
VERB	verb	run, runs, running, eat, ate, eating
X	other	sfpksdpsxmsa
SPACE	space	

Table 9: POS general tags within spaCy [47]

12.6. Advanced Part of Speech Tagging (Specific Tags)

Advanced POS Tagging implies a single distinction – the use of specific tags, as opposed to general tags. There are 16 general tags and 53 specific tags, so we can certainly extract more nuanced information from a more specific approach towards tagging.

TAG	**POS**	**MORPHOLOGY**	**DESCRIPTION**
$	SYM		symbol, currency
``	PUNCT	PunctType=quot PunctSide=ini	opening quotation mark
”	PUNCT	PunctType=quot PunctSide=fin	closing quotation mark
,	PUNCT	PunctType=comm	punctuation mark, comma
-LRB-	PUNCT	PunctType=brck PunctSide=ini	left round bracket
-RRB-	PUNCT	PunctType=brck PunctSide=fin	right round bracket
.	PUNCT	PunctType=peri	punctuation mark, sentence closer
:	PUNCT		punctuation mark, colon or ellipsis
ADD	X		email
AFX	ADJ	Hyph=yes	affix
CC	CCONJ	ConjType=comp	conjunction, coordinating
CD	NUM	NumType=card	cardinal number
DT	DET		determiner
EX	PRON	AdvType=ex	existential there
FW	X	Foreign=yes	foreign word

GW	X		additional word in multi-word expression
HYPH	PUNCT	PunctType=dash	punctuation mark, hyphen
IN	ADP		conjunction, subordinating or preposition
JJ	ADJ	Degree=pos	adjective
JJR	ADJ	Degree=comp	adjective, comparative
JJS	ADJ	Degree=sup	adjective, superlative
LS	X	NumType=ord	list item marker
MD	VERB	VerbType=mod	verb, modal auxiliary
NFP	PUNCT		superfluous punctuation
NIL	X		missing tag
NN	NOUN	Number=sing	noun, singular or mass
NNP	PROPN	NounType=prop Number=sing	noun, proper singular
NNPS	PROPN	NounType=prop Number=plur	noun, proper plural
NNS	NOUN	Number=plur	noun, plural
PDT	DET		predeterminer
POS	PART	Poss=yes	possessive ending
PRP	PRON	PronType=prs	pronoun, personal
PRP$	DET	PronType=prs Poss=yes	pronoun, possessive
RB	ADV	Degree=pos	adverb
RBR	ADV	Degree=comp	adverb, comparative
RBS	ADV	Degree=sup	adverb, superlative
RP	ADP		adverb, particle
SP	SPACE		space
SYM	SYM		symbol
TO	PART	PartType=inf VerbForm=inf	infinitival "to"
UH	INTJ		interjection
VB	VERB	VerbForm=inf	verb, base form
VBD	VERB	VerbForm=fin Tense=past	verb, past tense
VBG	VERB	VerbForm=part Tense=pres Aspect=prog	verb, gerund or present participle
VBN	VERB	VerbForm=part Tense=past Aspect=perf	verb, past participle
VBP	VERB	VerbForm=fin Tense=pres	verb, non-3rd person singular present
VBZ	VERB	VerbForm=fin Tense=pres Number=sing Person=three	verb, 3rd person singular present
WDT	DET		wh-determiner

WP	PRON		wh-pronoun, personal
WP$	DET	Poss=yes	wh-pronoun, possessive
WRB	ADV		wh-adverb
XX	X		unknown
_SP	SPACE		

Table 10: POS specific tags within spaCy [47]

12.7. Syntactic Dependencies

A syntactic dependency (SD) is simply the relation between two words, based on their position within the text or sentence. For each pair of words, one word is a "governor", whilst the other is the "dependent" [48]. This relationship is intuitively displayed in Figure 14, alongside part of speech tagging. Yet again, with the help of spaCy, we parse the affirmations and attribute all syntactic dependencies, identifying the labels as per the table below.

LABEL	**DESCRIPTION**
acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
clf	classifier
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
dep	unspecified dependency
det	determiner
discourse	discourse element
dislocated	dislocated elements
expl	expletive
fixed	fixed multiword expression
flat	flat multiword expression
goeswith	goes with
iobj	indirect object
list	list
mark	marker
nmod	nominal modifier

nsubj	nominal subject
nummod	numeric modifier
obj	object
obl	oblique nominal
orphan	orphan
parataxis	parataxis
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

Table 11: Syntactic Dependency labels within spaCy [47]

12.8. Named Entity Recognition

Named entity recognition is the process of detecting and classifying key information in text (typically referred to as entities). This operation may also bear the name of entity extraction, identification, or chunking. Any word or group of words that consistently refers to the same item is considered an entity [49]. Below is an intuitive visualization of how named entity recognition has been applied.

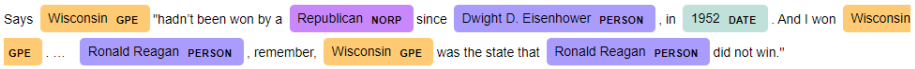

Figure 16: Visualization of named entity recognition being applied on a sample

Based on the spaCy documentation, every discovered entity is assigned to one of the categories (types) below.

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.

TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

Table 12: Entity types according to the spaCy documentation [47]

12.9. Machine Learning

Stages such as text pre-processing or certain aspects of feature engineering were a build-up for the attempt of classification, which will be performed by machine learning algorithms. Part of speech tagging, named entity recognition and identifying syntactic dependencies were for the benefit of better understanding our dataset. They offered a human-like approach towards better understanding what distinguishing features may be encountered based on veracity – which are essential for answering our first research question ("How does deceptive language vary from truthful language?"). The process of better understanding a dataset is often also called exploratory data analysis.

We leverage certain aspects from previous sub-sections in an attempt to answer our second research question regarding our capability to properly classify affirmations based on veracity. This stage is powered by scikit-learn [50].

12.9.1. Bag-of-Words Model

Bag-of-words (BoW) is one of the simplest means of creating feature vectors (also called representations) from the text. It essentially requires a set of words known (the vocabulary) and the ability to somehow identify whether a certain word is present (through document vectorization) [51].

The vocabulary is generally created by creating a set from the text corpus – this is why punctuation removal, lowercasing and lemmatization play an essential role in reducing the memory occupied. The vectors are the length of the vocabulary, and Boolean values will be used to represent which words from the current sentence or sample are present [51]. One drawback of this model (and where the name of bag stems from) is that these types of vectors do not take order into consideration. As such, the following samples would be regarded in the same way:

I hated the apple and loved the movie == I hated the movie and loved the apple

Other than the above-mentioned drawback, BoW may also pose a problem for vast corpora of data, where vocabularies contain thousands of entries. Storing vectors of such lengths for all sentences can quickly prove to be redundant and may require certain memory optimizations.

12.9.2. N-gram Model

N-gram is a probabilistic language model that attempts to predict the upcoming word in a sequence based on what came previously. The name stems from the fact that we usually analyze a sequence of "n" words. If n is 2, we call it a bigram (and so on) [52]. These probabilities are based upon frequency counts of a particular sequence of words, making use of Bayes' probability formulas. This computation implies calculating *P(w|h)*, the probability that the next word will be w, given history h [52].

$$P\left(apples|I\ like\right) = \frac{C\left(I\ like\ apples\right)}{C\left(I\ like\right)}$$

Computing entire sequences of words can be achieved by applying chain rule of probability on the probability decomposition:

$$P\left(X_1 \ldots X_n\right) = P\left(X_1\right) P\left(X_2|X_1\right) \ldots P\left(X_n|X_{1:n-1}\right)$$
$$P\left(w_1 \ldots w_n\right) = P\left(w_1\right) P\left(w_2|w_1\right) \ldots P\left(w_n|w_{1:n-1}\right)$$

Throughout our application, we only leverage bigrams and trigrams, as increasing n beyond 3 or 4 tends to yield worse results and increases computational requirements.

12.9.3. Term Frequency-Inverse Document Frequency Model

Term Frequency-Inverse Document Frequency (TF-IDF) is a mechanism through which words are weighted based on their importance in a corpus or series of documents. Similar to the n-gram model, it relies on the frequency counts of that particular word throughout the text [53].

As the name suggests, there are two components to this particular model: the Term Frequency (TF) and the Inverse Document Frequency (IDF). The former is the count of a particular word throughout the document, divided by the total count of words within that same document. The latter is calculated as a logarithm of the documents count within the corpora divided by the documents count based on where that particular word appears [53].

In a more mathematical expression, we have the following:

$$TF\left(word\right) = \frac{count\left(no.oftimes\ the\ word\ appears\ in\ a\ document\right)}{count\left(number\ of\ words\ in\ the\ document\right)}$$

$$IDF\left(word\right) = ln(\frac{count\left(no.of\ documents\right)}{count\left(no.\ of\ documents\ containing\ word\right)})$$

12.9.4. Multinomial Naïve Bayes

Multinomial Naïve Bayes is a probabilistic classifier, which will attempt to predict that a certain sample belongs to a particular class based on the highest output probability. Similar to the n-gram, it relies on Bayes' probabilistic theorem and assumes that every feature is independent of the others (which is actually a drawback due to the complex nature of real-life features) [54].

As it so happens in our case, MNB is often used on classification tasks involving textual corpora. It is built upon the calculation of probabilities of events, based on another event having already happened. Should we have two classes, A and B, respectively, the probability of B given A is denoted as:

$$P(A|B) = \ P(A) \ * \ \frac{P(B|A)}{P(B)}$$

As all algorithms, MNB has advantages and disadvantages – on the positive side, it is easy to implement, scalable, and can be used on both, discrete and continuous data. On the downside, it is usually not the best-performing prediction algorithm, and is only suitable for classification tasks [55].

12.9.5. Logistic Regression

Logistic Regression, like MNB, is a model used for classification purposes, implying that the predicted variable (also called target or class) is categorical. More specifically, the output of our application is of two types: lies or truths – which implies a binary classification task [56]. Let X be our features matrix, W our weights matrix, and b our biases, sigmoid (σ) as our activation function, and $\hat{y}$ our predicted output, we have the following calculations:

$$Z \ = \ W^T \ \times \ X \ + \ b$$

$$\hat{y} = \ \sigma(Z) \ = \ \frac{1}{1 \ + \ e^{-Z}}$$

If Z is large, σ(Z) will have a value close to 1. Conversely, if Z is small, σ(Z) will be approximately 0. This can be more intuitively visualised in the figure below:

There is an interesting distinction between Logistic Regression and Multinomial Naïve Bayes, which lies within their approaches: the former is discriminative, whereas the latter is generative. Generative models calculate the distribution of probabilities of a feature x in regard to its target y and attempt to predict the probability of y given x, denoted by P(y|x). Their discriminative counterparts directly model the probability P(y|x) by learning the relationship between the input and output through the leverage of an error function (or cost function, if we calculate on batches) [57].

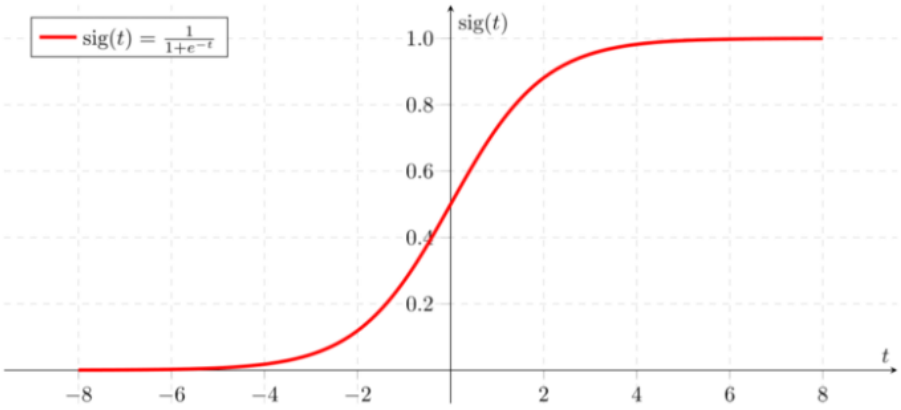

Figure 17: Sigmoid function graph [56]

12.9.6. Measuring Performance

The first measure of performance utilized is very trivial – accuracy. Naturally, we would like to see how many of the samples we correctly predicted out of the total number of samples within the test set. This can help us gauge how well the model is able to generalize.

Compute Area under the Receiver Characteristic Curve (ROC AUC) is yet another measure of performance for classification tasks [58]. It depicts the discriminating threshold generated in a binary classifier system by showing the true positive rate against the false positive rate [59].

We will leverage scikit-learn [50] to generate confusion matrices containing the results in more detail. They will be partly available within the Statistics and Results section or the Appendix.

12.10. Automated Machine Learning (Neural Architecture Search)

Automated Machine Learning (AutoML) aims to democratize the approach towards machine learning by automating the process of creating neural network architectures.

The difficulty of identifying a high-performing deep neural network design is addressed by the study of architecture search. This may involve experimenting with the number of layers, neurons, activation functions, and many other design considerations. Due to not having to analyze multiple designs meticulously, automated architecture search can significantly speed up the creation of new deep learning applications [60].

For our application, we leverage AutoKeras [61], an open-source framework, to build models capable of classifying text in an unsupervised manner. This is done with the help of the Text Classification class, where we can set a series of trials (experiments), and the framework will simply try out various combinations of layers and hyperparameters, optimizing for performance and results.

12.11. Fine-Tuning Pre-trained Language Models

In previous sections, we discussed how powerful state of the art language models are and that their inherent power stems from a large number of parameters and the vast amounts of data they have been trained on. Even though they perform general tasks well, we may wish to fine-tune their behavior to better fit another particular task – as is the case with our application.

We wish to experiment with various language models and attempt to correctly classify affirmations based on veracity. This implies the use of transfer learning, where we build upon the existing knowledge of models, with the hopes of attaining better results for our particular task.

To do this, we will leverage the optimized implementations of BERT (and its variants such as ALBERT, RoBERTa, etc.) offered by Hugging Face [62], a powerful library that contains implementations of a multitude of pre-trained models in an open-source environment.

There are multiple approaches towards transfer learning which will produce different results. The first possibility is that of retraining the entire model using another dataset. This implies that the entire model and all of its layers will be modified accordingly. To some extent, this is a limited approach, due to the fact that this implies we are attempting to accomplish a similar task as what the model was already pre-trained on. To actually make a difference, we would have to provide a lot of data.

Another option might be to "freeze" certain layers (keep them unmodified), and train on other layers (and modify these instead). This is an interesting approach, as we may want to keep certain layers that detect more general features, but may wish to change the latter layers, which delve into greater detail, or more generally, identify specific features.

For our application, we opt for a third approach – freezing the entire network, and attaching another layer at the end, along a SoftMax activation function. We will introduce a dense layer (also named fully-connected) to help us reduce the output to just two classes: truths and lies. We will then activate the final layer with the SoftMax function so as to offer the probabilities that a particular input belongs to one of the two classes. This means that only the last layer will undergo training, and the rest of the model can do what it does best.

13. Statistics, Results and Discussions

This section aims to present the insights gained and the results of experiments performed. The subsections each aim to address one of the two research questions, in a distinct manner. While the previous section's aim was to offer an overview of the technical approach, this section will help us draw conclusions based on what we have uncovered.

13.1. General Linguistic Statistics

General Linguistic statistics refers to the ability to scale our general feature engineering process over the entire dataset. We would like to see how these basic features contrast in regard to veracity. The table below presents the means for both truths and lies, respectively. The following statistics are built upon the larger and more permissive variant of our dataset.

	Truthful Statements Means	**False Statements Means**
Number of letters	109.7	106.6
Number of words	18.3	17.5
Avg. Word Length	5.1	5.2
Readability Score (Gun)	9.8	9.6

Table 13: Means (averages) of features based on veracity

We further plot the histogram of some general linguistic features by placing sentences into buckets according to the number of words in a sentence, and the count of such sentences.

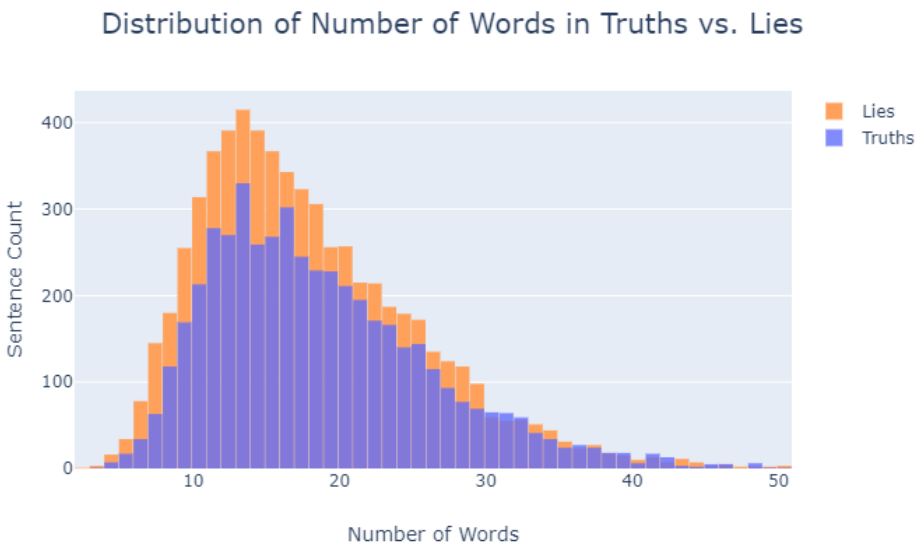

Figure 18: Distribution of the number of words based on the veracity

Similar operations will be performed for adjectives, nouns, verbs, and readability score.

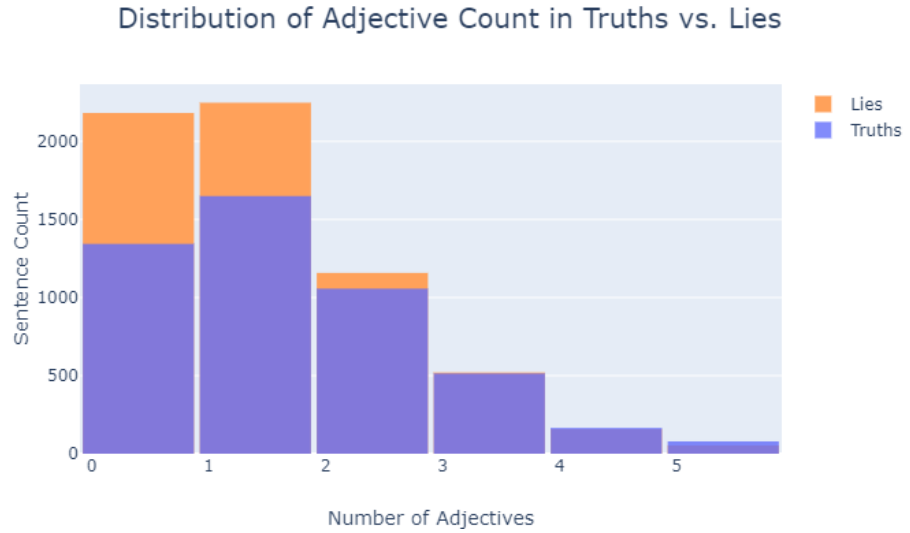

Figure 19: Distribution of the number of adjectives based on the veracity

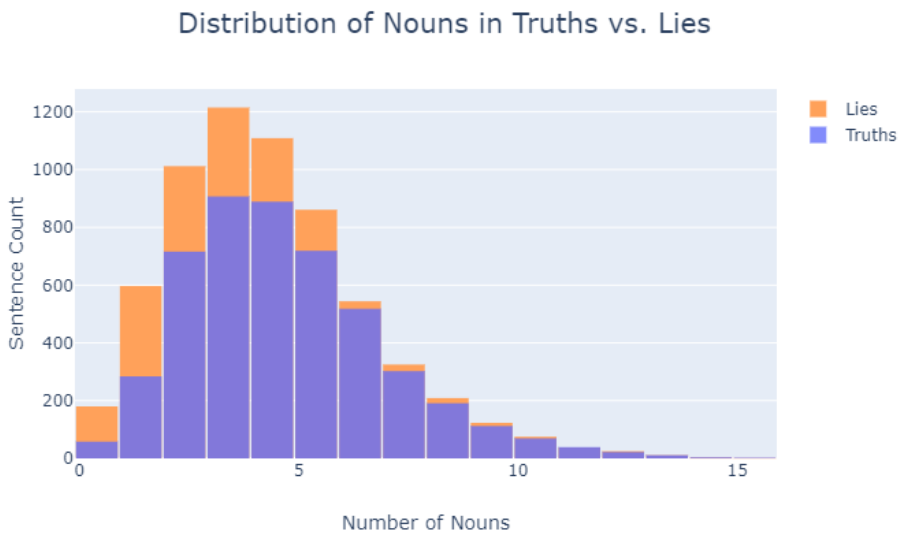

Figure 20: Distribution of nouns based on the veracity

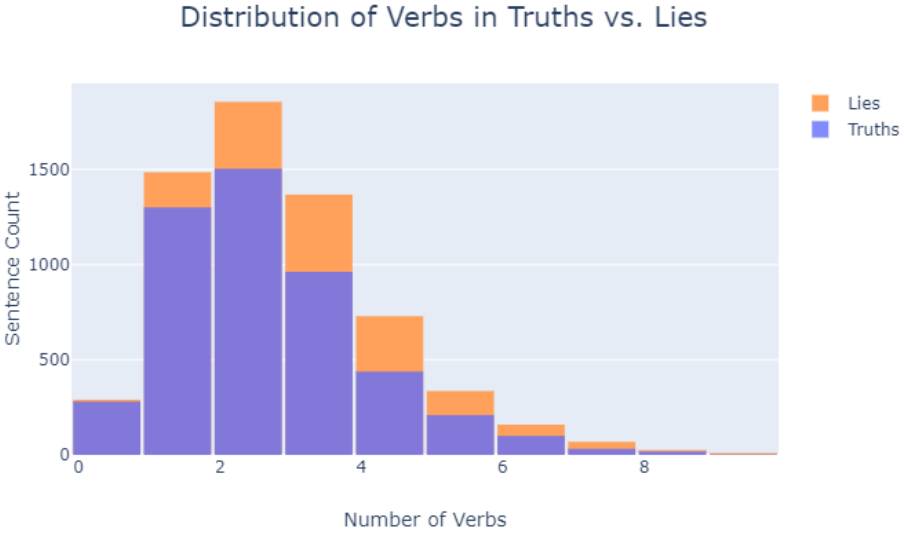

Figure 21: Distribution of verbs based on the veracity

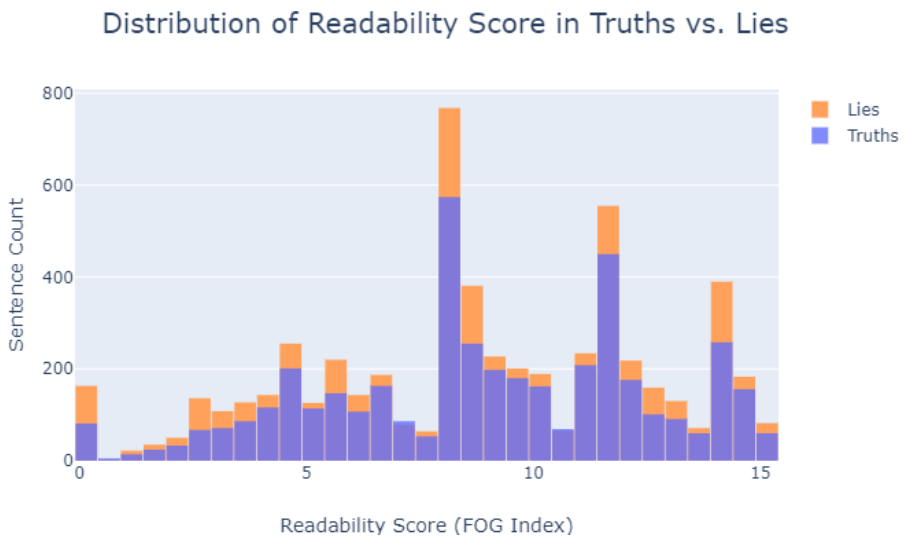

Figure 22: Distribution of Fog Readability Score based on the veracity

The explanation for why there are always more lies than truths in a most buckets is directly correlated to the imbalance in data, which was mentioned at the beginning of the previous section. This is why the first table of this particular subsection was crucial – to present that even though the number of samples is larger for deceptive affirmations, the means are still very close. This should help imply that truths and lies bear similar general linguistic characteristics, with slight variations. The readability score poses no exception, as can be seen in the figure above.

13.2. Part of Speech Tagging (General Tags)

The tables presented in the Methodology and Approach based on the spaCy documentation [47] play a crucial role in understanding how certain features, or in this particular case, parts of speech, vary based on veracity. The following table helps us uncover distinguishing features in truths and lies, respectively.

We also include the P-Value, which is a metric that expresses the likelihood that a particular difference may have occurred by chance, where the statistical significance decreases as the P-Value increases. This essentially means that the lower the P-Value, the more significant a particular entry [63]. We also impose a threshold for the P-Value with the purpose of selecting entries with high significance.

Part of Speech (General Tags)	Truthful Statements Means	False Statements Means	P-Value
adjective	1.37	1.13	0.000

conjunction or particle or adverb	2.27	2.04	0.000
adverb	0.70	0.59	0.000
coordinating conjuction	0.37	0.33	0.004
determiner	1.83	1.64	0.000
interjection	0.00	0.01	0.007
noun	4.35	3.96	0.000
number	0.96	0.64	0.000
possessive	0.49	0.60	0.000
proper noun	2.02	2.51	0.000
symbol	0.14	0.12	0.048
verb	2.28	2.48	0.000

Table 14: Parts of speech general tag statistics

Due to the difficulty in properly reading the table above, visualizations are created based on General POS tags. Entries are sorted based on largest the discrepancy.

The figure below makes it much easier to see distinguishing features as follows: lies tend to contain more proper nouns, verbs, and possessive pronouns, whereas truthful statements contain more nouns, numbers, adjectives, conjunctions, determiners and adverbs. The values are sorted in a descending manner with respect to largest feature discrepancy.

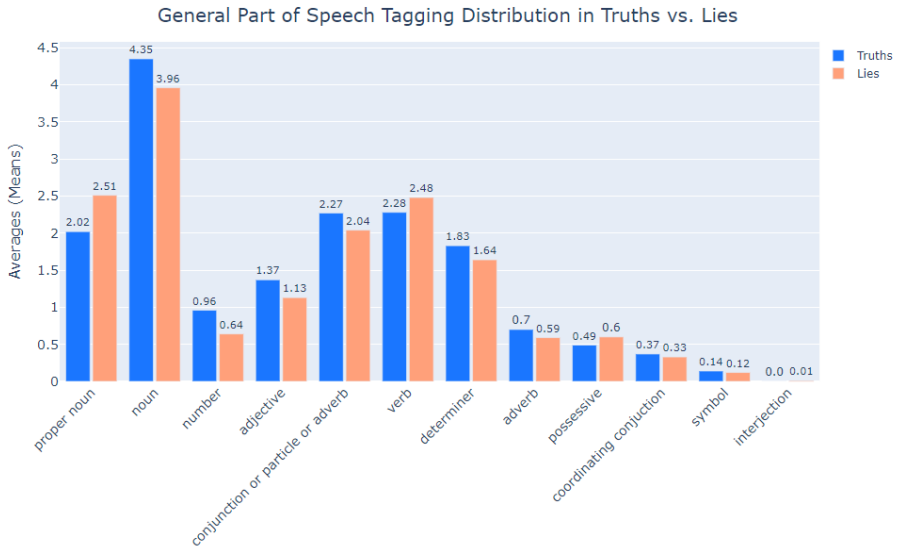

Figure 23: Plotting part of speech (POS) general tag statistics

13.3. Advanced Part of Speech Tagging (Specific Tags)

We apply the same type of operations for a more nuanced overview of the parts of speech usage throughout our dataset. Similarly, due to the difficulty in readability, we will create a visualization that aims to assist the process of understanding our results. The entries shall be sorted based on the largest discrepancy.

Part of Speech (Specific Tags)	**Truthful Statements Means**	**False Statements Means**	**P-Value**
symbol, currency	0.13	0.11	0.039
opening quotation mark	0.85	0.88	0.004
closing quotation mark	1.04	1.12	0.000
punctuation mark, comma	0.54	0.49	0.000
punctuation mark, sentence closer	1.12	1.09	0.009
conjunction, coordinating	0.37	0.33	0.004
cardinal number	0.96	0.64	0.000
determiner	1.72	1.53	0.000
punctuation mark, hyphen	0.15	0.14	0.049

conjunction, subordinating or preposition	2.59	2.24	0.000
adjective	1.10	0.98	0.000
adjective, comparative	0.18	0.09	0.000
adjective, superlative	0.10	0.05	0.000
verb, modal auxiliary	0.15	0.19	0.000
noun, singular or mass	3.09	2.83	0.000
noun, proper singular	1.92	2.41	0.000
noun, plural	1.29	1.16	0.000
predeterminer	0.02	0.01	0.003
possessive ending	0.12	0.15	0.000
adverb	0.72	0.64	0.000
adverb, particle	0.06	0.08	0.001
infinitival "to"	0.24	0.31	0.000
interjection	0.00	0.01	0.007
verb, base form	0.47	0.61	0.000
verb, past tense	0.57	0.62	0.015
verb, gerund or present participle	0.28	0.34	0.000
verb, past participle	0.48	0.45	0.017
verb, non-3rd person singular present	0.48	0.38	0.000
verb, 3rd person singular present	0.72	0.79	0.000

Table 15: Part of speech (POS) specific tag statistics

Building upon the previous subsection, we identify distinguishing features to a more nuanced extent. Falsehoods contain more proper singular nouns, and verbs addressing 3rd person. In contrast, truthful statements pertain to more conjunctions and prepositions, cardinal numbers, singular or mass nouns, determiners, plural nouns, adjectives, and verbs addressing 1st or 2nd person singularly.

The previous two subsections offer rather interesting and somewhat intuitive distinguishing features based on the part of speech tagging, where truthful language contains more potential numerical data (perhaps facts or statistics), alongside more affirmations based on ownership (1st person). Conversely, lies are attributed to certain people (proper nouns), which can be backed up by the use of verbs addressing 3rd person.

The figure below presents a more intuitive visualization of the table above, sorted based on largest difference in means.

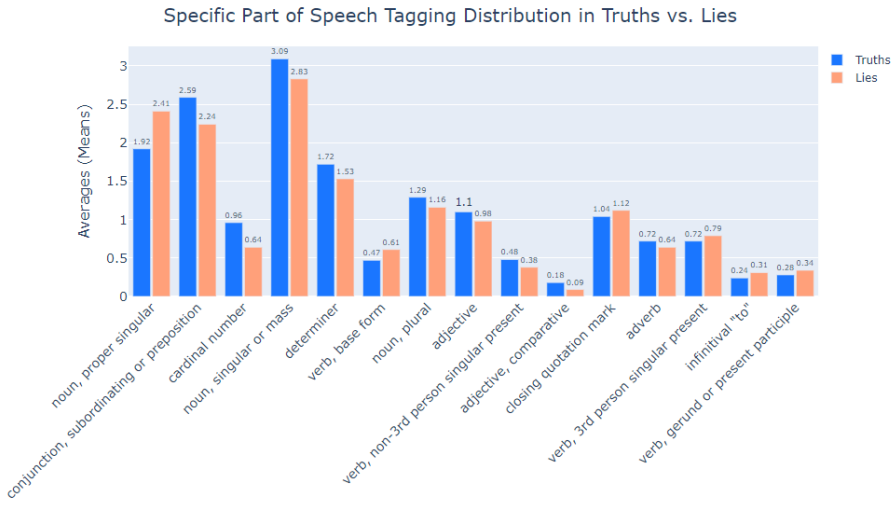

Figure 24: Plotting part of speech (POS) specific tag statistics

13.4. Syntactic Dependencies

Syntactic Dependency Label	Truthful Statements Means	False Statements Means	P-Value
adverbial clause modifier	0.23	0.25	0.011
adverbial modifier	0.69	0.57	0.000
adjectival modifier	1.23	1.03	0.000
auxiliary	0.71	0.82	0.000
case marking	0.12	0.15	0.000
coordinating conjunction	0.36	0.33	0.002
clausal complement	0.37	0.49	0.000
compound	1.76	2.01	0.000
conjunct	0.39	0.36	0.034
determiner	1.64	1.46	0.000
numeric modifier	0.58	0.37	0.000
open clausal complement	0.16	0.21	0.000

Table 16: Syntactic Dependency statistics

Once more, we plot the means of truths and lies based on veracity, but this time in regard to syntactic dependencies. As before, the entries are sorted based on the largest mean discrepancy.

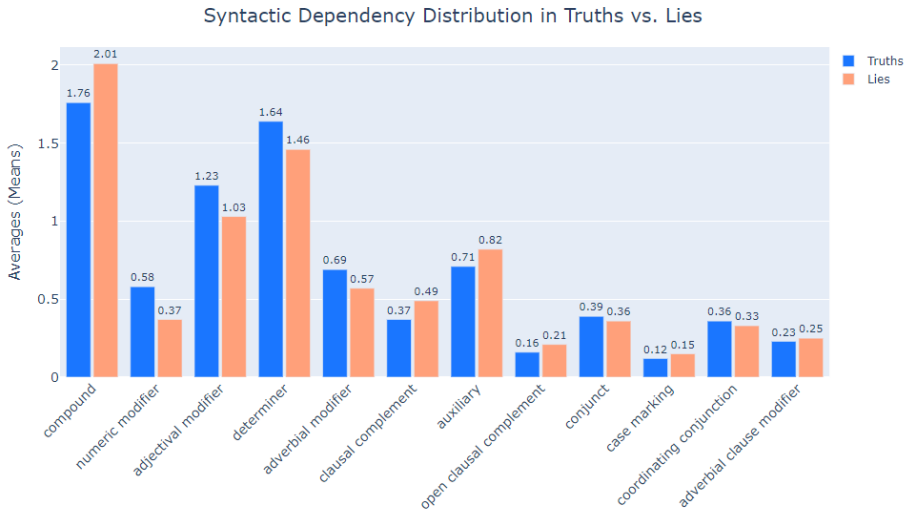

Figure 25: Plotting syntactic dependency (SD) label statistics

As per the figure above, we notice that truthful affirmations contain, on average, more numeric modifiers, adjectival modifiers, determiners, adverbial modifiers, and conjunctions. Conversely, falsehoods contain more compound dependencies, alongside auxiliaries. Even though the differences might be hard to gauge as a non-expert in linguistics, it is rather clear that there is syntactical difference as to how sentences are formulated, and how words tie in together to form phrases or larger pieces of text. In the nature of it all, one might even say we may have the possibility of possibly uncovering intent, simply based on the structure of a particular sentence, but there may be more convolved facets in regard to such an affirmation.

13.5. Named Entity Recognition

Named Entity Recognition is the last statistical operation performed with the help of spaCy, but certainly not the least. As in the previous subsections, we identify distinguishing features based on veracity, and plot them in a bar chart according to largest discrepancy.

Entity Tag	Truthful Statements Means	False Statements Means	P-Value

people	0.35	0.51	0.000
organizations	0.27	0.34	0.000
regions	0.44	0.39	0.000
art	0.01	0.02	0.000
dates	0.32	0.19	0.000
percentages	0.16	0.09	0.000
money	0.15	0.12	0.004
ordinality	0.05	0.03	0.000
numerals	0.31	0.20	0.000

Table 17: Named Entity Recognition statistics

Once again, for practical purposes, we create a plot to help us better visualize distinguishing features.

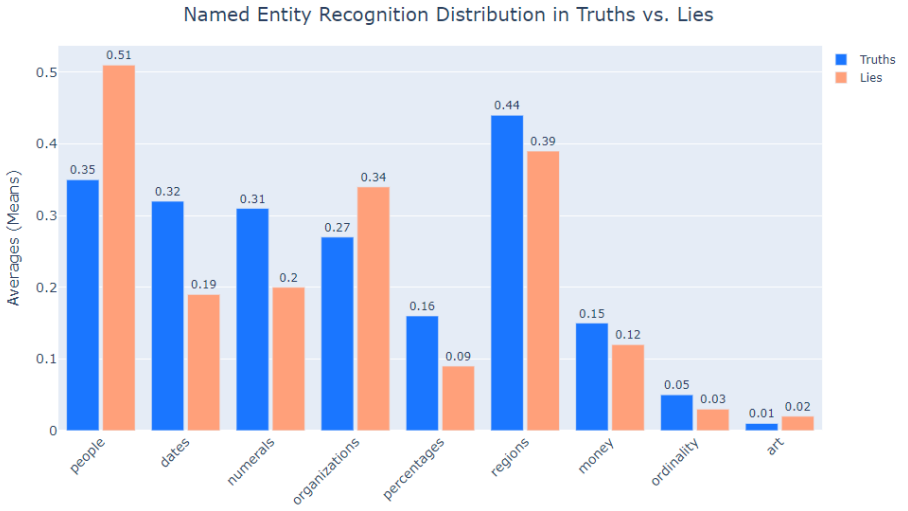

Figure 26: Plotting named entity recognition (NER) tags statistics

By looking at the figure above, we can see that lies contain, on average, more people's and organization names, whereas truthful affirmations tend to contain more dates, numerals, percentages, region names, and generally more numerical-based data. This a very interesting insight as it might suggest that truths may contain more details and concrete statistics, which to some extent resembles the idea of expressing facts. Although this may sound intuitive, we can now concretely gauge the extent to which these aspects vary throughout sentences. The results may also suggest that false affirmations are more heavily concentrated on other people, possibly accusations of some kind. Although we are able to

distinguish certain preeminent features, they only tip the scale towards a certain verdict and should not be taken for granted.

13.6. Common Words (Word Clouds)

As a preamble before delving into the task of classification, we would like to see the most common words throughout our dataset, based on veracity. To do this, we create Word Clouds [64]. We will apply the same operation for lemmas.

Figure 27: Truthful Statements Word Cloud (words)

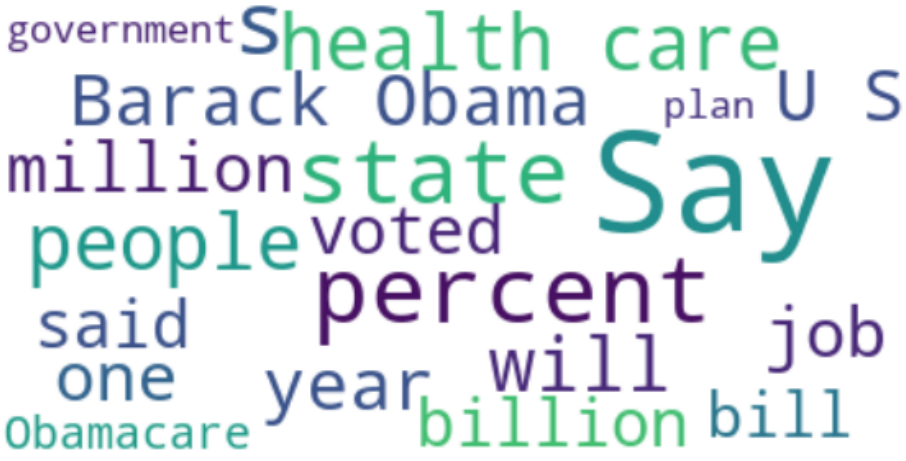

Figure 28: False Statements Word Cloud (words)

Figure 29: Truthful Statements Word Cloud (lemmas)

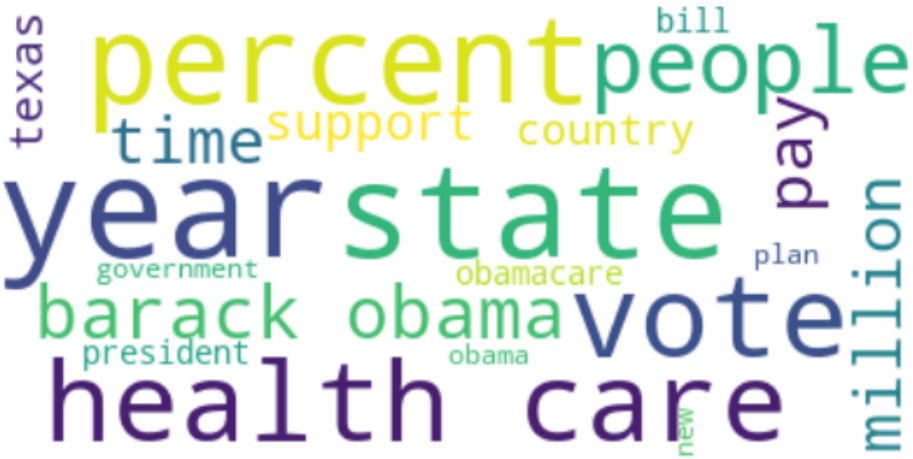

Figure 30: False Statements Word Cloud (lemmas)

Interestingly enough, we see that many words and lemmas are common, regardless of veracity. This is a subtle clue that suggests the difficulty which lies within the task of classification, as will be seen in the upcoming subsections.

13.7. Machine Learning Classification Experiments

If the previous subsections aimed to address the research question regarding differences in language based on veracity, we will now explore the task of correctly

classifying statements based on machine learning models. Our experiments will leverage mechanisms mentioned in the previous section and are performed on both variants of our dataset. The table below aims to provide a summary of the results obtained upon training on the more permissive variant of the dataset. More detailed results regarding this stage can be found in the Appendix section.

The following notations will be used: LR for Logistic Regression, MNB for Multinomial Naïve Bayes, TF-IDF for Term Frequency-Inverse Document Frequency and BoW for Bag-of-Words.

Model	**N-gram Range**	**N_folds (classifiers)**	**Normalization Type**	**ROC_AUC (test data)**	**Accuracy (test data)**
LR + BoW	-	3	Light	0.69	0.65
MNB + BoW	-	3	Light	0.68	0.65
LR + BoW	-	3	Full	0.67	0.64
MNB + BoW	-	3	Full	0.68	0.64
LR + N-gram	2	3	Light	0.7	0.65
MNB + N-gram	2	3	Light	0.69	0.65
LR + N-gram	3	3	Light	0.7	0.65
MNB + N-gram	3	3	Light	0.69	0.65
LR + N-gram	2	3	Full	0.68	0.64
MNB + N-gram	2	3	Full	0.67	0.63
LR + N-gram	3	3	Full	0.68	0.65
MNB + N-gram	3	3	Full	0.67	0.63
LR + TF-IDF Vectorizer (N-gram)	2	3	Light	0.7	0.66
MNB + TF-IDF Vectorizer (N-gram)	2	3	Light	0.69	0.63
LR + TF-IDF Vectorizer (N-gram)	3	3	Light	0.71	0.66
MNB + TF-IDF Vectorizer (N-gram)	3	3	Light	0.69	0.62

Table 18: Machine learning experiments results in larger data variant

We also include a graph with the purpose of visualizing the best-performing models per approach based upon their results.

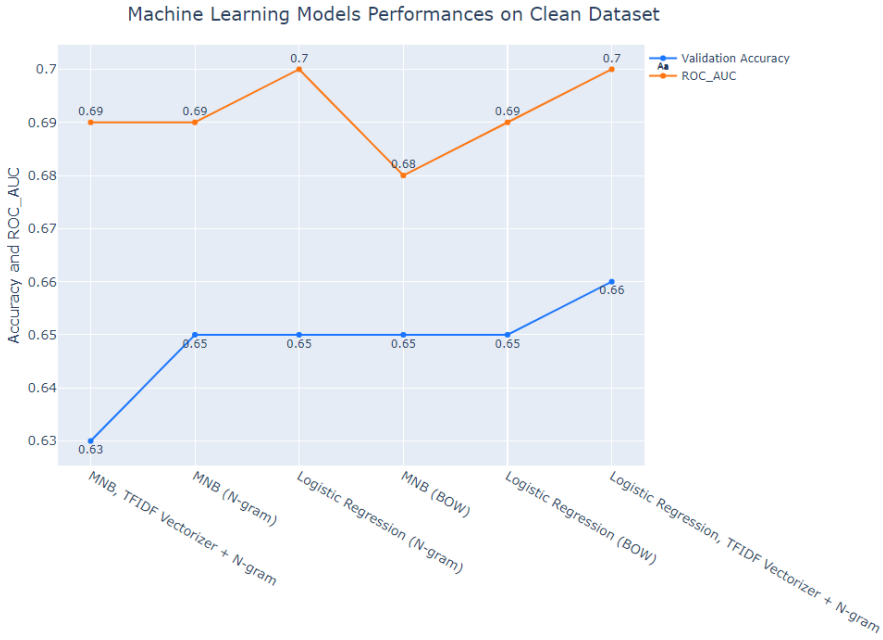

Figure 31: Best-performing Machine Learning models per approach for clean dataset

We notice that our best performing model leverages logistic regression as a classification algorithm, and a TF-IDF vectorizer based on n-grams. In this case, both bigrams and trigrams achieve the same accuracy, but have a slight difference in ROC AUC score, which is higher than average, implying that the model's predictions are more stable.

To gauge how data quality can impact stability in predictions, we perform the same experiments on our strict dataset variant, which contains fewer samples and clearer distinctions between the classes, as presented in the table below.

Model	**N-gram Range**	**N_folds (classifiers)**	**Normalization Type**	**ROC_AUC (test data)**	**Accuracy (test data)**
LR + BoW	-	3	Light	0.69	0.68
MNB + BoW	-	3	Light	0.69	0.68
LR + BoW	-	3	Full	0.68	0.67

MNB + BoW	-	3	Full	0.66	0.67
LR + N-gram	2	3	Light	0.7	0.69
MNB + N-gram	2	3	Light	0.68	0.69
LR + N-gram	3	3	Light	0.7	0.69
MNB + N-gram	3	3	Light	0.68	0.68
LR + N-gram	2	3	Full	0.68	0.67
MNB + N-gram	2	3	Full	0.68	0.67
LR + N-gram	3	3	Full	0.69	0.67
MNB + N-gram	3	3	Full	0.68	0.67
LR + TF-IDF Vectorizer (N-gram)	2	3	Light	0.7	0.69
MNB + TF-IDF Vectorizer (N-gram)	2	3	Light	0.69	0.69
LR + TF-IDF Vectorizer (N-gram)	3	3	Light	0.71	0.69
MNB + TF-IDF Vectorizer (N-gram)	3	3	Light	0.69	0.68

Table 19: Machine learning experiments results on strict data variant

An interesting insight is that even though we have less data, we obtain overall higher performances regarding ROC AUC score and especially accuracy. There is a 3% increase in accuracy by simply providing the model with more qualitative data. As a consequence, the ROC AUC score also increases, to enforce prediction stability or confidence in the predictions made. The low variation in accuracy between models might suggest that this is the general range of results we can hope to obtain using statistical approaches.

Taking into consideration that the test data is properly split, and the training data is somewhat imbalanced, correctly classifying more than two-thirds of the affirmations based solely on textual features is impressive. The task of political classification can be difficult even for a human when there is no possibility of fact-checking the information received – more so for a statistical model which lacks previous knowledge about politics, or even the English language. Below is a graph presenting the results from the table above in a more intuitive manner.

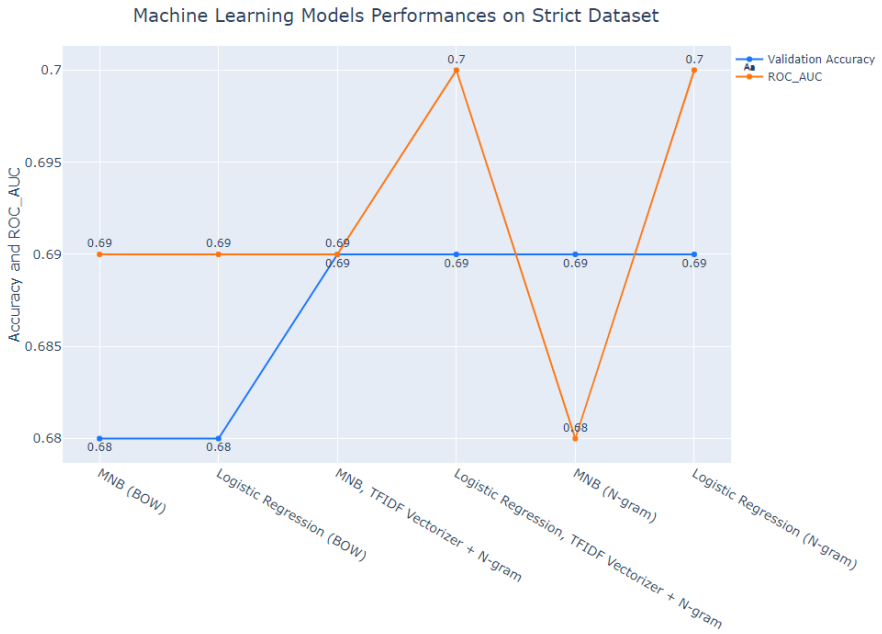

Figure 32: Best-performing machine learning models per approach for the strict dataset

13.8. Automated Machine Learning Classification Experiments

AutoKeras, which is an Automated Machine Learning tool, aims to obtain machine learning models with as little human involvement as possible. This experiment serves as a sanity check, more than an attempt to obtain the best possible results and are performed solely on the permissive variant of the dataset. Should we have made a mistake in the previous subsection regarding our approach towards machine learning classification, this method would certainly make this obvious through the difference in results obtained, which are presented below.

Model	**Max Trials**	**Max Epochs**	**Validation Loss**	**Validation Accuracy**
Text Classifier	1	2	0.99	0.63
Text Classifier	2	5	1.23	0.55
Text Classifier	15	3	0.65	0.6463
Text Classifier + Light Normalization	2	2	0.68	0.584

Text Classifier + Full Normalization	2	2	0.863	0.56

Table 20: Automated machine learning (AutoML) results

The following figure presents the best-performing models per approach in a more intuitive fashion.

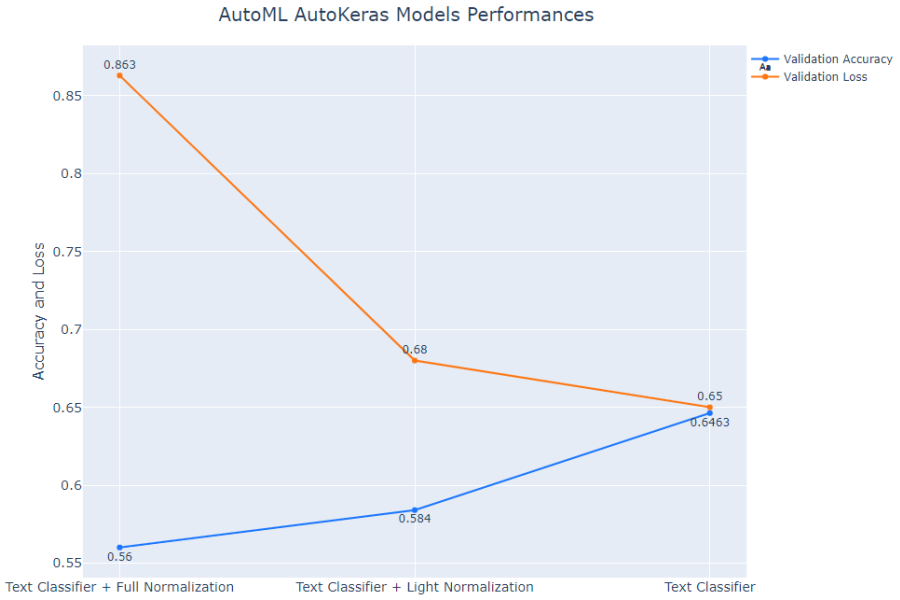

Figure 33: Best-performing automated machine learning (AutoML) models per approach

What we are truly interested in is the validation accuracy, which is really close to the best machine learning model performance from the previous subsection, superseding it. This performance was obtained by a model that underwent 15 trials, with a maximum of 3 epochs. The trials express how many attempts will be conducted for building architectures for the specified task. The training time for such a model ended up being north of an hour, as opposed to less than a few seconds for the models obtained in the previous subsection, and all for less than a 2% boost in accuracy. The important gain from this experiment is that we confirm our approach being correct in both subsections, as they both present closely related results. We also notice that full normalization provides a negative impact in machine learning and automated machine learning classification experiments, respectively.

13.9. Fine-Tuning Pre-trained Language Models Classification Experiments

In an attempt to push our results further, we fine-tune pre-trained state-of-the-art language models. The purpose is to leverage the process of transfer learning and potentially push our classification results further, as per the methodology described in the previous section. We will also attempt the classification task on both dataset variants so as to gauge how data quality may create a difference in performance, and implicitly accuracy.

The table below contains the results obtained for BERT and its variants for the larger variant of the dataset.

Model	**Max sequence length**	**Batch size**	**Learning rate**	**Epochs**	**Validation Loss**	**Accuracy**
bert-base-uncased	50	64	1e-5	50	0.657	0.62
bert-base-uncased	50	48	4e-5	30	0.648	0.63
bert-base-cased	50	48	4e-5	30	0.664	0.62
bert-large-uncased	50	48	4e-5	30	0.667	0.59
bert-large-cased	50	48	4e-5	30	0.670	0.58
roberta-base	50	48	4e-5	30	0.677	0.56
roberta-large	50	48	4e-5	30	0.675	0.58
distilroberta-base	50	48	4e-5	30	0.660	0.63
albert-base-v2	50	48	4e-5	30	0.651	0.61

Table 21: Deep learning experiments results for large dataset variant

We notice in the image below that RoBERTa reaches the best overall performance of 63% accuracy, slightly inferior to the best statistical model. We notice that the validation accuracy is not too low, meaning that the model is not learning too much from the data.

This is an interesting insight and can be explained as follows: complex models pre-trained on vast amounts of data really show their true value in complex texts, and implicitly longer sequences of text. These large models are good on so

many language tasks due to their knowledge of languages – English in our case. This means that our dataset containing rather short affirmations and sentences does not present itself as much of a learning source for our fine-tuned models. Through these results we find that statistical models are still very good options for shorter, somewhat simpler texts.

As per the latest subsections, we create a visualization to more easily grasp the information from the table above.

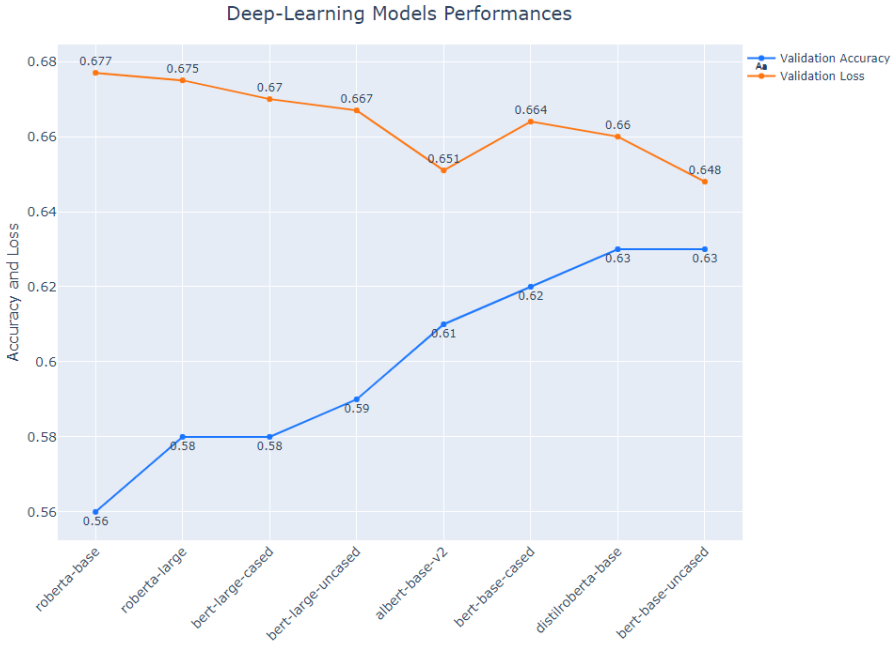

Figure 34: Deep learning results visualization for large dataset

We fine-tune BERT and its variants for the strict variant of our dataset, which contains fewer entries (samples) to see if we obtain different results. These can be seen in the table below.

Model	**Max sequence length**	**Batch size**	**Learning rate**	**Epochs**	**Validation Loss**	**Accuracy**
bert-base-uncased	50	64	1e-5	50	0.650	0.60
bert-base-uncased	50	48	4e-5	30	0.636	0.60

bert-base-uncased	50	48	4e-5	100	0.615	0.65
bert-base-cased	50	48	4e-5	30	0.657	0.60
bert-large-uncased	50	48	4e-5	30	0.667	0.61
bert-large-cased	50	48	4e-5	30	0.683	0.59
roberta-base	50	48	4e-5	30	0.685	0.68
roberta-base	50	48	5e-6	50	0.691	0.65
roberta-base	50	48	1e-6	100	0.692	0.63
roberta-large	50	48	4e-5	30	0.679	0.67
distilroberta-base	50	48	4e-5	30	0.673	0.66
distilroberta-base	50	48	1e-5	50	0.684	0.64
albert-base-v2	50	48	4e-5	30	0.647	0.60

Table 22: Deep learning experiments results for strict dataset variant

Below is the graphical representation of the table above.

We notice that the best performing fine-tuned pre-trained language model performs slightly worse than the best statistical model, by 1% to be exact. The difference in quality directly affects both models – the clearer the distinction, the better the predictions, despite fewer data.

If for the more permissive data, the gap was of 3% in accuracy obtained, the results obtained for strict data reduce that gap to a 1% accuracy difference, which some might argue is negligible. We confirm yet again confirm that statistical models are still highly viable options when dealing with shorter text samples and apparently overall superior to more sophisticated state-of-the-art models.

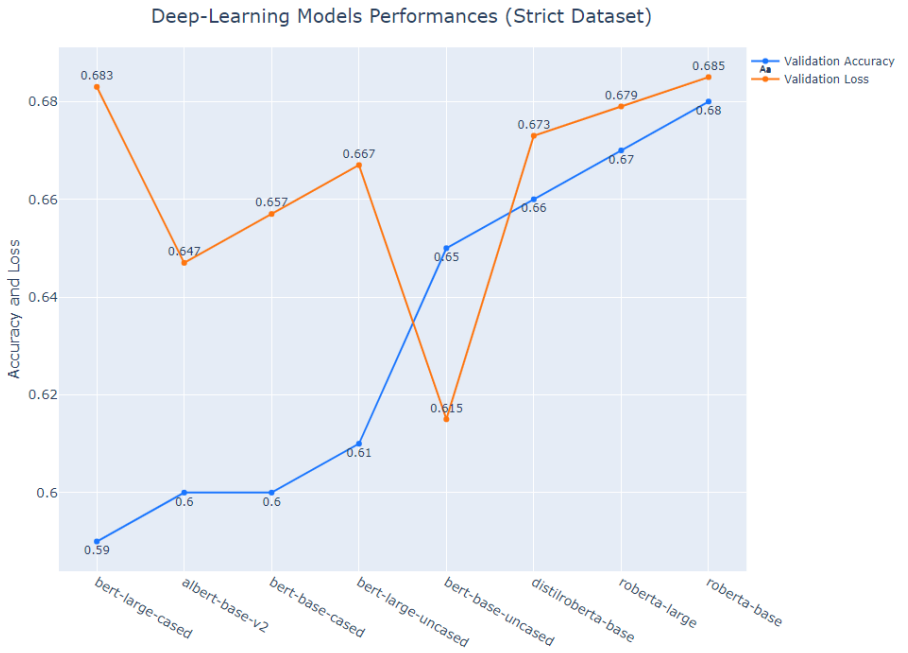

Figure 35: Deep learning experiments visualization for strict dataset

As before, the validation loss is still rather high for the process of model training, which might imply that the model is not learning many new features from the current data. One interesting note is that whilst the RoBERTa model was the worst-performing model for the permissive (clean) data experiments, it became the best-performing for the stricter variant of the dataset. Even though we do not manage to push the boundaries of classical and statistical machine learning using state-of-the-art models in the task of veracity classification, we most certainly better understand the behavior of such models, and inherently where they do (or do not) thrive. Had the task of classification been formulated differently, we may have obtained completely different results.

In an attempt to enrich the results obtained, another series of experiments containing augmented data was attempted. After finalizing training, it was observed that this particular attempt provided no benefits whatsoever. On the contrary, it offered worse results than those performed on normal data. As such, this round of experiments was not included in this paper.

14. Conclusions

Natural Language Processing is a mature domain of AI but has recently benefited from innovations brought along by deep learning methods. Deep learning has opened the way for various applications, new levels of performance, which hold the potential to impact and change society – hopefully in a positive manner.

Truth Detection, together with closely related tasks have been discussed to offer a grasp upon the tightly knit nature of sub-domains within deep learning, but especially NLP. A significant improvement in one area may determine better results in another, particularly within the new era of AI.

The latest technical approaches and neural network architectures are also discussed and explained to a certain degree of detail, as so to offer a better understanding of the underlying mechanisms that drive modern applications. Conceptual innovations, such as Transformers, are the ones that truly drive a domain further, shifting our perspective and ability to create applications to a whole new level. A technical improvement at the most primal level will inevitably improve results in most applications NLP related, setting new baselines for performance and results.

Research questions rooted in first principles are formulated with the aim of better understanding the complexity of our task, and potentially finding complete or partial answer for these queries. To start, a case study based on real-life data is devised. This implies parsing and creating our own dataset containing political affirmations in the USA and their inherent level of truthfulness. Two variants are created so as to more selectively gauge potential differences in data quality, and how these are reflected in classification tasks.

The first part of the implementation is dedicated to uncovering linguistic distinguishing linguistic features based on veracity within the newly obtained dataset. The aim is to perform exploratory data analysis and understand how the data is perceived from a human-like perspective. We notice recurring themes based on parts of speech, entities, or syntactic dependencies, which help us better understand how statements may vary in regard to veracity. Even though certain features increase the probability of a sample belonging to a certain class, it does not guarantee it, and can therefore be misleading. Textual features can imply veracity only to a certain extent and may require further study.

The second part of the implementation aims to automatically classify statements, based on veracity, by leveraging machine learning and fine-tuned pre-trained deep-learning language models. Two-thirds of validation statements are correctly predicted using binary classification, but more importantly, we understand how data quality, models, data quantity, and problem formulation can play an essential role in attempting to approach such an involved task. In a real-life scenario, truth detection is a much more fine-grained task than what binary classification can account for. It is especially difficult to classify affirmations in classes such as “half-true” with no fact-checking mechanism – which bears the

question: can multiple models, each with a separate purpose, address such a complex problem? What if tasks such as truth detection, fact-checking, stance detection, click-bait detection or even fake news detection had their own fine-tuned model, and were part of a more complex system which could offer weights to each model output to offer a final verdict?

Contrarily, perhaps a model sufficiently large enough, multiple orders of magnitude larger than GPT-3, could account for all the above-mentioned tasks, and would only require minimal fine-tuning to address such a complicated issue as deception detection. Such a model might have the ability to query itself and provide a verdict of veracity by simply analyzing pieces of information. Currently, such a model would be too large to create and would involve investments so huge that only a few state actors or companies could ever think to afford it. It is entirely possible that a system much closer to Artificial General Intelligence is required to address the full complexity of truth detection and all of its intricacies.

As has been shown, practical applications have yet to catch up with the rapid evolvement of language models, as to leverage the newly found discoveries in order to obtain more sophisticated and performant applications. Whilst a lot of practical applications of AI, and implicitly NLP, are intellectual property that may not be available on the public domain, we can nonetheless conclude that there are a plethora of improvements and innovations simply waiting to be discovered, which will hopefully result in the advancement of our society.

References

[1] Li Deng and Yang Liu. "A joint introduction to natural language processing and to deep learning". In: *Deep Learning in Natural Language Processing*. Springer International Publishing, 1st Jan. 2018, pp. 1–22. ISBN: 9789811052095. DOI: 10.1007/978-981-10-5209-5_1.

[2] *Study: On Twitter, false news travels faster than true stories | MIT News | Massachusetts Institute of Technology*. URL: https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308.

[3] *Information Overload Helps Fake News Spread, and Social Media Knows It - Scientific American*. URL: https://www.scientificamerican.com/article/information-overload-helps-fake-news-spread-and-social-media-knows-it/.

[4] Ray Oshikawa, Jing Qian and William Yang Wang. *A Survey on Natural Language Processing for Fake News Detection*. Tech. rep. 2020. URL: https://www.snopes.com/fact-check/.

[5] Arkaitz Zubiaga et al. "Detection and Resolution of Rumours in Social Media: A Survey". In: *ACM Computing Surveys* 51.2 (Apr. 2017), p. 32. DOI: 10.1145/3161603.

[6] James Thorne and Andreas Vlachos. *Automated Fact Checking: Task formulations, methods and future directions Title and Abstract in Greek*. Tech. rep. URL: http://www.fullfact.org/.

[7] William Ferreira and Andreas Vlachos. *Emergent: a novel data-set for stance classification*. Tech. rep.

[8] Guixian Xu et al. "Sentiment analysis of comment texts based on BiLSTM". In: *IEEE Access* 7 (2019), pp. 51522–51532. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2909919.

[9] Chris Potts. *Deceptive language*. Tech. rep. URL: https://web.stanford.edu/class/cs424p/materials/ling287-handout-11-02-deception.pdf.

[10] Nicholas D Duran et al. "The linguistic correlates of conversational deception: Comparing natural language processing technologies". In: *Applied Psycholinguistics* 31 (2010), pp. 439–462. DOI: 10.1017/S0142716410000068.

[11] Jian Liu, Yubo Chen and Kang Liu. "Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection". In: vol. 33. 01. AAAI Press, July 2019, pp. 6754–6761. ISBN: 9781577358091. DOI: 10.1609/aaai.v33i01.33016754.

[12] Ian J Goodfellow et al. *Generative Adversarial Nets*. Tech. rep. URL: https://arxiv.org/pdf/1406.2661.pdf.

[13] Estela Saquete et al. "Fighting post-truth using natural language processing: A review and open challenges". In: *Expert Systems With Applications* 141 (2020), p. 112943. DOI: 10.1016/j.eswa.2019.112943.

[14] *State of AI Report 2020*. [Online; accessed 2020-11-13]. 0. URL: https://www.stateof.ai/?s=04&fbclid=IwAR04MG62-d0iXS-h11PqAyGdNzF_6w73KKfsAY0vWLJWn7-ugeBGffCOupo.

[15] *The Intuition Behind Transformers — Attention is All You Need | by Sam Palani | Towards Data Science*. 0. URL: https://towardsdatascience.

com / the - intuition - behind - transformers - attention - is - all - you - need - 393b5cfb4ada.

[16] Dzmitry Bahdanau, Kyung Hyun Cho and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: International Conference on Learning Representations, ICLR, 1st Sept. 2015.

[17] Ashish Vaswani et al. "Transformer: Attention is all you need". In: *Advances in Neural Information Processing Systems 30* (2017), pp. 5998–6008. ISSN: 10495258.

[18] *Google AI Blog: Reformer: The Efficient Transformer*. URL: https://ai.googleblog.com/2020/01/reformer-efficient-transformer.html.

[19] Nikita Kitaev, Łukasz Kaiser and Anselm Levskaya. "Reformer: The Efficient Transformer". In: (13th Jan. 2020). URL: http://arxiv.org/abs/2001.04451.

[20] *Google AI Blog: Rethinking Attention with Performers*. URL: https://ai.googleblog.com/2020/10/rethinking-attention-with-performers.html.

[21] Krzysztof Choromanski et al. "Rethinking Attention with Performers". In: (30th Sept. 2020). URL: http://arxiv.org/abs/2009.14794.

[22] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1 (10th Oct. 2018), pp. 4171–4186. URL: http://arxiv.org/abs/1810.04805.

[23] *BERT: State of the Art NLP Model, Explained - KDnuggets*. URL: https://www.kdnuggets.com/2018/12/bert-sota-nlp-model-explained.html.

[24] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv* (2019). URL: http://arxiv.org/abs/1907.11692.

[25] Zhenzhong Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: (26th Sept. 2019). URL: http://arxiv.org/abs/1909.11942.

[26] *Google AI Blog: ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations*. URL: https://ai.googleblog.com/2019/12/albert-lite-bert-for-self-supervised.html.

[27] *Google's ALBERT Is a Leaner BERT; Achieves SOTA on 3 NLP Benchmarks | by Synced | SyncedReview | Medium*. URL: https://medium.com/syncedreview/googles-albert-is-a-leaner-bert-achieves-sota-on-3-nlp-benchmarks-f64466dd583.

[28] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21 (23rd Oct. 2019), pp. 1–67. URL: http://arxiv.org/abs/1910.10683.

[29] *Google AI Blog: Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer*. 0. URL: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html.

[30] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv* (28th May 2020). URL: http://arxiv.org/abs/2005.14165.

[31] *GPT-3: The First Artificial General Intelligence? | by Julien Lauret | Towards Data Science*. URL: https://towardsdatascience.com/gpt-3-the-first-artificial-general-intelligence-b8d9b38557a1.
[32] *GLUE Benchmark*. URL: https://gluebenchmark.com/.
[33] *SuperGLUE Benchmark*. URL: https://super.gluebenchmark.com/leaderboard/.
[34] *State of AI Report 2020*. URL: https://www.stateof.ai/.
[35] *The Stanford Question Answering Dataset*. URL: https://rajpurkar.github.io/SQuAD-explorer/.
[36] Erik F Tjong, Kim Sang and Fien De Meulder. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. Tech. rep. URL: http://lcg-www.uia.ac.be/conll2003/ner/.
[37] *CoNLL 2020 | CoNLL*. URL: https://www.conll.org/2020.
[38] *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning | by Dipanjan (DJ) Sarkar | Towards Data Science*. URL: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a.
[39] *PolitiFact | National*. URL: https://www.politifact.com/truth-o-meter/.
[40] *PolitiFact*. URL: https://www.politifact.com/staff/.
[41] *the-sergiu/TruthDetection: My BSc Thesis implementation which dwells upon the complex task of Truth Detection (often also referred to as Deception Detection or Lie Detection)*. URL: https://github.com/the-sergiu/TruthDetection.
[42] *Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.
[43] *spaCy · Industrial-strength Natural Language Processing in Python*. URL: https://spacy.io/.
[44] *Stemming and lemmatization*. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.
[45] *The Gunning Fog Readability Formula*. URL: https://readabilityformulas.com/gunning-fog-readability-formula.php.
[46] *Annotation Specifications · spaCy API Documentation*. URL: https://web.archive.org/web/20201207014953/https://spacy.io/api/annotation.
[47] *Syntactic dependencies*. URL: https://webanno.github.io/webanno/use-case-gallery/dependency-parsing/.
[48] *What is named entity recognition (NER) and how can I use it? | by Christopher Marshall | super.AI | Medium*. URL: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d.
[49] *scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/.
[50] *A Gentle Introduction to the Bag-of-Words Model*. URL: https://machinelearningmastery.com/gentle-introduction-bag-words-model/.
[51] Daniel Jurafsky. *N-Gram Language Models*. URL: https://web.stanford.edu/%7B%5Ctextasciitilde%7Djurafsky/slp3/3.pdf.

[52] *Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining*. URL: http://www.tfidf.com/.

[53] *Applying Multinomial Naive Bayes to NLP Problems: A Practical Explanation | by Synced | SyncedReview | Medium*. URL: https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf.

[54] *Multinomial Naive Bayes Explained: Function, Advantages Disadvantages, Applications in 2021 | upGrad blog*. URL: https://www.upgrad.com/blog/multinomial-naive-bayes-explained/.

[55] *Logistic Regression — Detailed Overview | by Saishruthi Swaminathan | Towards Data Science*. URL: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[56] *Comparison between Naïve Bayes and Logistic Regression | DataEspresso*. URL: https://dataespresso.com/en/2017/10/24/comparison-between-naive-bayes-and-logistic-regression/.

[57] David J. Hand and Robert J. Till. "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems". In: *Machine Learning* 45.2 (2001), pp. 171–186. ISSN: 08856125. DOI: 10.1023/A:1010920819831.

[58] *Confusion Matrix, $ROC_AUC and Imbalanced Classes in Logistic Regression | by Lily Su | Medium$*. URL: https://medium.com/@lily_su/confusion-matrix-roc-auc-and-imbalanced-classes-in-logistic-regression-5c7ead3deefc.

[59] *AutoML*. 0. URL: https://www.automl.org/automl/.

[60] *Text Classification - AutoKeras*. URL: https://autokeras.com/tutorial/text_classification/.

[61] *Hugging Face – The AI community building the future*. URL: https://huggingface.co/.

[62] *P-Value Definition*. URL: https://www.investopedia.com/terms/p/p-value.asp.

[63] *WordCloud for Python documentation — wordcloud 1.8.1 documentation*. URL: https://amueller.github.io/word_cloud/.

Appendix

This section contains the full results obtained after performing the classification experiments. They are included for the benefit of a more detailed comparison of performance.

Machine Learning Results

Below are all the results obtained for the machine learning experiments on both dataset variants. Confusion matrices were constructed on the test data. As a reminder, the following notation is used: LR (Logistic Regression), BoW (Bag-of-Words), MNB (Multinomial Naïve Bayes), TF-IDF (Term Frequency-Inverse Document Frequency).

Results for Clean Dataset

1. **LR + BoW, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1474	424
Positive Class	751	708

2. **NB + BoW, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1490	408
Positive Class	766	693

3. **LR + BoW, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1474	424
Positive Class	751	708

4. **NB + BoW, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1475	423
Positive Class	780	679

5. **LR + bigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1463	435
Positive Class	744	715

6. **MNB + bigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1337	561
Positive Class	620	839

7. **LR + trigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1493	405
Positive Class	757	702

8. **MNB + trigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1375	523
Positive Class	662	797

9. **LR + bigram, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1490	408
Positive Class	788	671

10. **MNB + bigram, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1608	290
Positive Class	954	505

11. **LR + trigram, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1500	398
Positive Class	783	676

12. **MNB + trigram, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1624	274
Positive Class	981	478

13. **LR + TF-IDF Vectorizer (bigram)**

	Negative Prediction	**Positive Prediction**
Negative Class	1479	419
Positive Class	720	739

14. **MNB + TF-IDF Vectorizer (bigram)**

	Negative Prediction	Positive Prediction
Negative Class	1686	212
Positive Class	1040	419

15. **LR + TF-IDF Vectorizer (trigram)**

	Negative Prediction	Positive Prediction
Negative Class	1445	453
Positive Class	699	760

16. **MNB + TF-IDF Vectorizer (trigram)**

	Negative Prediction	Positive Prediction
Negative Class	1723	175
Positive Class	1086	373

Results for Strict Dataset

1. **LR + BoW, Light Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1052	156
Positive Class	432	208

2. **NB + BoW, Light Normalization**

	Negative Prediction	Positive Prediction
Negative Class	952	256
Positive Class	340	300

3. **LR + BoW, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1088	120
Positive Class	490	150

4. **NB + BoW, Full Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	947	261
Positive Class	363	277

5. **LR + bigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1053	155
Positive Class	421	219

6. **MNB + bigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1029	179
Positive Class	400	240

7. **LR + trigram, Light Normalization**

	Negative Prediction	**Positive Prediction**
Negative Class	1066	142
Positive Class	431	209

8. **MNB + trigram, Light Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1043	165
Positive Class	428	212

9. **LR + bigram, Full Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1490	408
Positive Class	788	671

10. **MNB + bigram, Full Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1030	178
Positive Class	433	207

11. **LR + trigram, Full Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1098	110
Positive Class	493	147

12. **MNB + trigram, Full Normalization**

	Negative Prediction	Positive Prediction
Negative Class	1059	149
Positive Class	456	184

13. **LR + TF-IDF Vectorizer (bigram)**

	Negative Prediction	**Positive Prediction**
Negative Class	1093	115
Positive Class	460	180

14. **MNB + TF-IDF Vectorizer (bigram)**

	Negative Prediction	**Positive Prediction**
Negative Class	1031	177
Positive Class	405	235

15. **LR + TF-IDF Vectorizer (trigram)**

	Negative Prediction	**Positive Prediction**
Negative Class	968	240
Positive Class	336	304

16. **MNB + TF-IDF Vectorizer (trigram)**

	Negative Prediction	**Positive Prediction**
Negative Class	1723	175
Positive Class	1086	373

Deep Learning Results

Below are all the results obtained for the deep learning experiments on both dataset variants.

Results for Clean Dataset

1. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	551	400
Positive Class	236	492

	precision	recall	f1-score	support
0	0.70	0.58	0.63	951
1	0.55	0.68	0.61	728
accuracy			0.62	1679
macro avg	0.63	0.63	0.62	1679
weighted avg	0.64	0.62	0.62	1679

2. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	580	371
Positive Class	243	485

	precision	recall	f1-score	support
0	0.70	0.61	0.65	951
1	0.57	0.67	0.61	728
accuracy			0.63	1679
macro avg	0.64	0.64	0.63	1679
weighted avg	0.64	0.63	0.64	1679

3. **bert-base-cased**

	Negative Prediction	Positive Prediction
Negative Class	580	371
Positive Class	243	485

	precision	recall	f1-score	support
0	0.68	0.60	0.64	951
1	0.55	0.63	0.59	728
accuracy			0.62	1679
macro avg	0.62	0.62	0.62	1679
weighted avg	0.63	0.62	0.62	1679

4. **bert-large-uncased**

	Negative Prediction	Positive Prediction
Negative Class	541	410
Positive Class	280	448

	precision	recall	f1-score	support
0	0.66	0.57	0.61	951
1	0.52	0.62	0.56	728
accuracy			0.59	1679
macro avg	0.59	0.59	0.59	1679
weighted avg	0.60	0.59	0.59	1679

5. **bert-large-cased**

	Negative Prediction	Positive Prediction
Negative Class	473	478
Positive Class	221	507

	precision	recall	f1-score	support
0	0.68	0.50	0.58	951
1	0.51	0.70	0.59	728
accuracy			0.58	1679
macro avg	0.60	0.60	0.58	1679
weighted avg	0.61	0.58	0.58	1679

6. **roberta-base**

	Negative Prediction	Positive Prediction
Negative Class	315	636
Positive Class	95	633

	precision	recall	f1-score	support
0	0.77	0.33	0.46	951
1	0.50	0.87	0.63	728
accuracy			0.56	1679
macro avg	0.63	0.60	0.55	1679
weighted avg	0.65	0.56	0.54	1679

7. **roberta-large**

	Negative Prediction	Positive Prediction
Negative Class	483	468
Positive Class	236	492

	precision	recall	f1-score	support
0	0.67	0.51	0.58	951
1	0.51	0.68	0.58	728
accuracy			0.58	1679
macro avg	0.59	0.59	0.58	1679
weighted avg	0.60	0.58	0.58	1679

8. **distilroberta-base**

	Negative Prediction	Positive Prediction
Negative Class	585	366
Positive Class	260	468

	precision	recall	f1-score	support
0	0.69	0.62	0.65	951
1	0.56	0.64	0.60	728
accuracy			0.63	1679
macro avg	0.63	0.63	0.63	1679
weighted avg	0.64	0.63	0.63	1679

9. **albert-base-v2**

	Negative Prediction	Positive Prediction
Negative Class	575	376
Positive Class	274	454

	precision	recall	f1-score	support
0	0.68	0.60	0.64	951
1	0.55	0.62	0.58	728
accuracy			0.61	1679
macro avg	0.61	0.61	0.61	1679
weighted avg	0.62	0.61	0.61	1679

Results for Strict Dataset

1. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	352	248
Positive Class	122	202

	precision	recall	f1-score	support
0	0.74	0.59	0.66	600
1	0.45	0.62	0.52	324
accuracy			0.60	924
macro avg	0.60	0.61	0.59	924
weighted avg	0.64	0.60	0.61	924

2. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	316	284
Positive Class	90	234

	precision	recall	f1-score	support
0	0.78	0.53	0.63	600
1	0.45	0.72	0.56	324
accuracy			0.60	924
macro avg	0.62	0.62	0.59	924
weighted avg	0.66	0.60	0.60	924

3. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	416	184
Positive Class	136	188

	precision	recall	f1-score	support
0	0.75	0.69	0.72	600
1	0.51	0.58	0.54	324
accuracy			0.65	924
macro avg	0.63	0.64	0.63	924
weighted avg	0.67	0.65	0.66	924

4. **bert-base-cased**

	Negative Prediction	Positive Prediction
Negative Class	322	278
Positive Class	94	230

	precision	recall	f1-score	support
0	0.77	0.54	0.63	600
1	0.45	0.71	0.55	324
accuracy			0.60	924
macro avg	0.61	0.62	0.59	924
weighted avg	0.66	0.60	0.61	924

5. **bert-base-uncased**

	Negative Prediction	Positive Prediction
Negative Class	391	209
Positive Class	149	175

	precision	recall	f1-score	support
0	0.72	0.65	0.69	600
1	0.46	0.54	0.49	324
accuracy			0.61	924
macro avg	0.59	0.60	0.59	924
weighted avg	0.63	0.61	0.62	924

6. **bert-base-cased**

	Negative Prediction	Positive Prediction
Negative Class	333	267
Positive Class	112	212

	precision	recall	f1-score	support
0	0.75	0.56	0.64	600
1	0.44	0.65	0.53	324
accuracy			0.59	924
macro avg	0.60	0.60	0.58	924
weighted avg	0.64	0.59	0.60	924

7. **roberta-base**

	Negative Prediction	Positive Prediction
Negative Class	589	11
Positive Class	285	39

	precision	recall	f1-score	support
0	0.67	0.98	0.80	600
1	0.78	0.12	0.21	324
accuracy			0.68	924
macro avg	0.73	0.55	0.50	924
weighted avg	0.71	0.68	0.59	924

8. **roberta-base**

	Negative Prediction	Positive Prediction
Negative Class	497	103
Positive Class	214	110

	precision	recall	f1-score	support
0	0.70	0.83	0.76	600
1	0.52	0.34	0.41	324
accuracy			0.66	924
macro avg	0.61	0.58	0.58	924
weighted avg	0.63	0.66	0.64	924

9. **roberta-base**

	Negative Prediction	Positive Prediction
Negative Class	479	121
Positive Class	218	106

	precision	recall	f1-score	support
0	0.69	0.80	0.74	600
1	0.47	0.33	0.38	324
accuracy			0.63	924
macro avg	0.58	0.56	0.56	924
weighted avg	0.61	0.63	0.61	924

10. **roberta-large**

	Negative Prediction	Positive Prediction
Negative Class	555	45
Positive Class	262	62

	precision	recall	f1-score	support
0	0.68	0.93	0.78	600
1	0.58	0.19	0.29	324
accuracy			0.67	924
macro avg	0.63	0.56	0.54	924
weighted avg	0.64	0.67	0.61	924

11. **distilroberta-base**

	Negative Prediction	Positive Prediction
Negative Class	420	180
Positive Class	134	190

	precision	recall	f1-score	support
0	0.76	0.70	0.73	600
1	0.51	0.59	0.55	324
accuracy			0.66	924
macro avg	0.64	0.64	0.64	924
weighted avg	0.67	0.66	0.66	924

12. **distilroberta-base**

	Negative Prediction	Positive Prediction
Negative Class	400	200
Positive Class	133	191

	precision	recall	f1-score	support
0	0.75	0.67	0.71	600
1	0.49	0.59	0.53	324
accuracy			0.64	924
macro avg	0.62	0.63	0.62	924
weighted avg	0.66	0.64	0.65	924

13. **albert-base-v2**

	Negative Prediction	Positive Prediction
Negative Class	329	271
Positive Class	99	225

	precision	recall	f1-score	support
0	0.77	0.55	0.64	600
1	0.45	0.69	0.55	324
accuracy			0.60	924
macro avg	0.61	0.62	0.59	924
weighted avg	0.66	0.60	0.61	924