UNIVERSITY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Cybersecurity and Data Science»

ΠΜΣ «Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Fact checking with large language models** Έλεγχος γεγονότων με την χρήση μεγάλων γλωσσικών μοντέλων
Student's name-surname: Ονοματεπώνυμο Φοιτητή:	**Koufopoulos Ioannis-Aris** Κουφόπουλος Ιωάννης-Άρης
Father's name: Πατρώνυμο:	**Emmanouil** Εμμανουήλ
Student's ID No: Αριθμός Μητρώου:	**ΜΠΚΕΔ2214**
Supervisor: Επιβλέπων:	**Papastefanatos George, MSc Instructor** Παπαστεφανάτος Γεώργιος, Διδάσκων ΠΜΣ

June 2024 - Ιούλιος 2024

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dr. George Papastefanatos **MSc Professor**	**Dimitrios Apostolou** **Professor**	**Dr. Stavros Maroulis** **MSc Professor**
Δρ. Γεώργιος Παπαστεφανάτος Διδάσκων ΠΜΣ	Δημήτριος Αποστόλου Καθηγητής	Δρ. Σταύρος Μαρούλης Διδάσκων ΠΜΣ

Abstract

The recently increased focus on misinformation has stimulated research in fact checking, the task of assessing the truthfulness of a claim. While automated fact checking pipelines have seen a growing interest over the past few years, the rapid development of technologies that can be utilized to aid the automated pipeline, have paved the way for the revision of the current methods and approaches. The swift advancements in the field of deep learning, along with the emergence of the large language models (LLMs) have produced various techniques that can successfully replace traditional natural language processing (NLP) approaches. Thus, for the purposes of this thesis, we will employ an already established fact checking pipeline, and replace most of its methods with modern architectures, such as text embedding models, that are commonly found in LLMs. Furthermore, we will expand the dataset that was constructed for the purpose of storing Greek language statements and replace the classification models that were utilized to produce a verdict, with cutting-edge neural networks. The results of this dissertation proved that our approach produces very promising results in many cases, often reaching and surpassing the accuracy results produced by traditional NLP techniques. Furthermore, our methodology allows for considerable opportunities for further enchantment, since the final pipeline that we established is highly customisable and provides a breeding ground for additional experimentation—either by further enlarging the dataset or by introducing more effective text embedding models that may be produced in the future, and classification models with different architectures that are more reliable in capturing patterns.

Keywords: Large language models, Automated Fact-Checking, Text embedding models, Greek fact-checking dataset

Περίληψη

Η πρόσφατη αυξημένη εξάπλωση της παραπληροφόρησης έχει διεγείρει την έρευνα γύρω από τον τομέα της επαλήθευσης γεγονότων, ο οποίος τομέας ορίζεται ως η διαδικασία αξιολόγησης της αλήθειας ενός ισχυρισμού. Ενώ οι αυτοματοποιημένες διαδικασίες επαλήθευσης γεγονότων έχουν δει αυξανόμενο ενδιαφέρον τα τελευταία χρόνια, η ταχεία ανάπτυξη τεχνολογιών που μπορούν να χρησιμοποιηθούν για την υποβοήθηση της αυτόματης διαδικασίας, έχουν ανοίξει τον δρόμο για την αναθεώρηση των τρεχουσών μεθόδων και προσεγγίσεων. Οι ταχείες εξελίξεις στον τομέα της βαθιάς μάθησης, μαζί με την εμφάνιση των μεγάλων γλωσσικών μοντέλων (LLMs) έχουν παραγάγει διάφορες τεχνικές που μπορούν να αντικαταστήσουν επιτυχώς τις παραδοσιακές προσεγγίσεις επεξεργασίας φυσικής γλώσσας (NLP). Επομένως, για τους σκοπούς αυτής της διατριβής, θα χρησιμοποιήσουμε μια ήδη υπάρχουσα διαδικασία επαλήθευσης γεγονότων και θα αντικαταστήσουμε τις περισσότερες από τις μεθόδους της με σύγχρονες αρχιτεκτονικές όπως μοντέλων ενσωμάτωσης κειμένου, που βρίσκονται συνήθως στα LLMs. Επιπλέον, θα επεκτείνουμε το σύνολο δεδομένων που δημιουργήθηκε για την αποθήκευση ισχυρισμών στην ελληνική γλώσσα και θα αντικαταστήσουμε τα μοντέλα ταξινόμησης που χρησιμοποιήθηκαν για την παραγωγή μιας απόφασης, με μοντέρνα νευρωνικά δίκτυα. Τα αποτελέσματα αυτής της διατριβής απέδειξαν ότι η προσέγγισή μας παράγει πολύ υποσχόμενα αποτελέσματα σε πολλές περιπτώσεις, συχνά φτάνοντας, και ξεπερνώντας τα αποτελέσματα που αφορούν την ακρίβεια και που παράγονται από παραδοσιακές τεχνικές NLP. Επιπλέον, η μεθοδολογία μας παρέχει σημαντικές ευκαιρίες για περαιτέρω βελτίωση, καθώς η τελική διαδικασία που καθιερώσαμε είναι ιδιαίτερα προσαρμόσιμη και παρέχει ένα γόνιμο έδαφος για επιπλέον πειραματισμό — είτε με την περαιτέρω διεύρυνση του συνόλου δεδομένων είτε με την εισαγωγή πιο αποτελεσματικών μοντέλων ενσωμάτωσης κειμένου που μπορεί να παραχθούν στο μέλλον, και μοντέλων ταξινόμησης με διαφορετικές αρχιτεκτονικές που είναι πιο αξιόπιστες στην ανίχνευση προτύπων.

Λέξεις-κλειδιά: Μεγάλα γλωσσικά μοντέλα, Αυτόματη επαλήθευση γεγονότων, Μοντέλα ενσωμάτωσης κειμένου, Ελληνικό σύνολο δεδομένων επαλήθευσης ισχυρισμών

Acknowledgements

I would like to the University of Piraeus for providing the foundational knowledge and providing me with the opportunity to attain my master’s degree in data science and cybersecurity. Additionally, I want to express my sincere gratitude to the Athena research and innovation center for offering the essential resources and assistance that allowed me to complete my thesis.

My sincere appreciation goes out to my supervisors, Dr. George Papastefanatos, Principal Researcher and Dimitris Katsiros, Research Associate, both from the ATHENA Research Center for their assistance, direction, and inspiration during the course of this study. Their helpful criticism and continuous availability enabled the completion of this thesis. Their perseverance and commitment to supporting me in reaching my objectives are greatly appreciated.

Lastly, I want to express my gratitude to my friends and family for their compassion and support throughout this difficult path. Their love and support have been a source of courage for me during this process.

Table of Contents

List of figures

List of tables

1 Introduction

Nowadays, the use of the web has become the main mean to obtain and share valuable information throughout the globe. However, in the last years, there has been an increasing number of false claims in news, social media, and various web sources. For this reason, automated fact-checking processes have seen a growing interest throughout various scientific communities, in domains such as public health, education, politics, business and finance etc. The use of AI and data driven techniques have emerged and complement the manual task of fact-checking. Thus, fact-checking organizations embrace those techniques in different ways, with the promise of accuracy and speed.

In the context of this thesis, we will make use of the word embedding techniques underlying in large language models (LLMs), for text classification, and thus replacing the already known NLP approaches. LLMs in particular, have shown remarkable abilities in comprehending and processing natural language tasks. They also exhibit fluency and adaptability far beyond natural language processing systems. Therefore, our goal is to make use of those technologies that are embedded into a large language model's architecture, to successfully classify Greek language claims in the form of text, which were harvested from Greek speaking web sources.

More specifically, the main idea of this study is to utilize an already established pipeline, which is integrated in the Check4Facts platform. Check4Facts is a fact-checking initiative based in Greece, launched in June 2022. It was developed through the collaboration of several institutions, including the National Center for Social Research, the National and Kapodistrian University of Athens, and the Athena Research Institute, with funding from the Hellenic Foundation of Research and Innovation. The site initially focuses on fact-checking claims related to immigration and crime, health, and climate change. This platform's machine learning (ML) algorithm lies in gathering web resources (web articles and documents) on an examined statement and extracting features that quantify the relation between the statement and its resources, as well as their characteristics. These features serve as indicators upon which the ML algorithm is trained for deciding whether a statement concerns a valid fact or not [1]. The current algorithm that was implemented on those grounds and will be analyzed in the next sections, differs by employing innovative state-of-the-art methods for feature extraction, model training and classification tasks, as a mean of achieving overall better performance results. The word embedding models that will be employed, are able to capture semantic relationships and meanings of text input, by representing them as a vector in a high-dimensional space. Word embedding models can also successfully surpass traditional NLP limitations such as the inability to capture complex linguistic relationships and the relatively large quantity of features needed for simple classification tasks.

Additionally, neural networks and deep learning processes will succeed the machine learning models and algorithms that were used for classification and fell short in achieving adequate results in terms of data variance and accuracy. Due to this, a new dataset is also introduced in this thesis that consists of the claims already used in the Check4Facts pipeline, with the addition of new claims gathered from Greek language websites. An extensive analysis of this dataset will be made in the next sections.

The first section of this thesis will present relating research made in various fields of the automated fact-checking domain, such the standard pipeline system and traditional text representation methods that translate text into a format that can be later processed by machine learning algorithms. We also present methods that extract linguistic features from textual information. Moreover, we present text similarity metrics that measure the degree of relevance between two pieces of text. Deep learning approaches for text representation are also demonstrated, along with some benchmark datasets that are utilized to evaluate the performance of the machine learning algorithms in fact-checking tasks. Finally, we present research relevant to the integration of LLMs to the fact checking pipeline.

Section three illustrates the methodology that was employed to collect and assemble a fact-checking dataset that will be utilized to assess the performance of the machine learning models for text classification. Section four displays technical details and analysis about our fact checking pipeline and the basis on which it was developed. It includes details about the exploration and harvesting of web pages, the construction of textual embedding vectors, and the utilization of neural networks, to classify the veracity of a claim.

Section five is dedicated to the analysis of the classification results. Different classification tasks were constructed, aiming to compare the performance of distinct train-test-validation splits of the final dataset, and with the use of varying feature combinations. Section six constitutes the conclusion of our thesis, where we summarize our findings and present the finding as well as the drawbacks that were met during this dissertation.

The primary objective of this thesis is to remodel an already assembled automated fact checking pipeline, like the one that the Check4Facts platforms offers.

- Enrich the already existing dataset of claims with new ones, that encompass a variety of subjects such as public health, immigration policies, politics, entertainment etc.

- The collection of relevant evidence that support or refute a claim will utilize modern methods and metrics that are usually found in LLMs

- Reorganize the feature generation process. Instead of utilizing traditional natural language processing tools, we will employ LLM architectures that show very promising result in capturing linguistic relationships and context.

- Introduce new classification models that are well-established in the field of deep learning and have demonstrated significant success across various domains. These models are widely recognized their effectiveness and are frequently utilized in numerous applications due to their promising results and robust performance.

- Execute various experimentation based on the above. We introduce different classification task and additionally employ varying sets of features in order to get a clear understanding of the performance of our models in terms of feature sets, number of external sources utilized, and model architecture.

2 Related Work

This section presents related work on automated fact checking approaches with an emphasis on large language models and machine learning techniques. Additionally, we present various pipeline implementations on automated fact checking that have emerged in the recent years. We also emphasize on text similarity metrics and traditional text representation methods, or text representation with the utilization of deep learning models. The text similarity metrics undertake the task of comparing pieces of text between them. The text representation methods are responsible for translating text into a machine-readable format, to be utilized by machine learning algorithms. Some popular linguistic features are also presented, which were usually employed by natural language processing tasks in order to extract semantic meaning and interpret textual information. Finally, we present some of the most common benchmark dataset in the automated fact-checking domain.

2.1 Automated fact checking pipeline

In its essence, an automated fact checking pipeline includes a series of steps that are taken to methodically confirm or reject the veracity of a claim that is stated in a form of speech or text. On of the characteristics that make those pipeline so promising, is the variety of machine learning and deep learning methods and techniques that are implemented, along with natural language processing methods. The significant advancements in machine learning and artificial intelligence, has given us the opportunity to re-evaluate the pipeline approaches on fact-checking, in a manner that no human intervention would be necessary, and the results regarding the authenticity of a statement would be satisfactory. In this section, we present the most important parts in the fact checking pipeline, which are common in all different types of pipelines.

2.1.1 Claim detection

The first stage in automated fact checking is the detection of arbitrary statements, which are selected for verification. Detection relies on the concept of check-worthiness. Check-worthy claims are defined as the claims for which the general public would be interested in knowing the truth. For example, “over six million Americans had COVID-19 in January” would be check-worthy, as opposed to “water is wet”. Claim detection can involve a binary decision for each potential claim, or an importance-ranking of claims. The latter considered a standard practice in journalistic fact-checking.

Another installation of claim detection is based on rumor detection. A rumor is defined as an unverified theory or statement that circulated the web, typically on social media. Rumor detection considers language subjectivity and growth of interest and readership through a social network. Those rumors are usually found in a form of social media posts, and in the final stages of this pipeline, a binary classifier determines the authenticity of each of those statements. Metadata, such as the number of likes and re-posts, is often used as features.

Check worthiness can also be subjective in many cases. For example, the importance of debunking information relating the COVID-19 pandemic, is not equally important to every social group. The check worthiness also varies over time, as countering misinformation relating current events are considered to be more important and is often prioritized over older arbitrary claims. For example, misinformation about COVID-19 has a greater social impact back in 2021 over misinformation about the Spanish flu. In addition, older rumors may have already been fact-checked by journalists, reducing their impact. Misinformation that is harmful to marginalized communities may also considered to be less worthy of a check by the general public than misinformation that targets the majority of the society. Claims originating from marginalized groups may be subject to greater scrutiny than claims originating from the vast majority. These kinds of biases might be reproduced in datasets that document the choices journalists make regarding which claims to rank highest.

To counter this, [30] introduced a stricter definition of whether a claim is worth checking or not. If a claim makes an assertion about the world that is verifiable with available evidence, then the statement is check-worthy. Claims on the other hand, that are based on opinions or experiences, are not.

In [35], the research addresses the fact checking sub task of claim check worthiness detection as a , a text classification problem where, given a statement, one must predict if the content of that statement makes “an assertion about the world that is checkable. Multiple variations of this task, as stated above, are rumour detection, check-worthiness ranking in political debates and speeches, and ‘citation needed’ detection on Wikipedia. Each of those sub-categories is focused on discovering claims that require additional verification, which is a common underlying issue. Figure 1 presents examples of check-worthy and non-check-worthy statements from those three different domains.

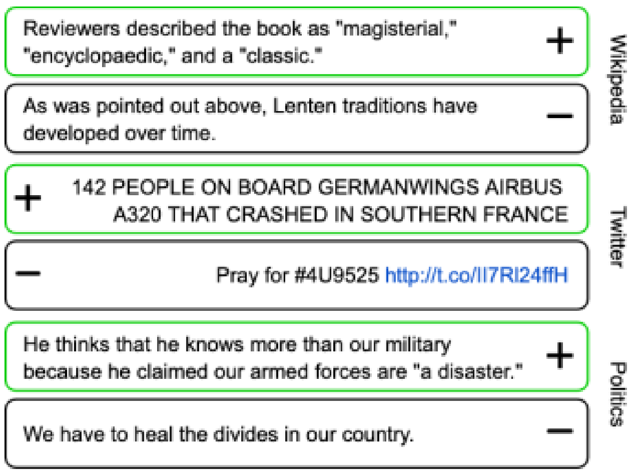

Reviewers described the book as "magisterial," "encyclopaedic," and a "classic."
+
As was pointed out above, Lenten traditions have developed over time.
−
Wikipedia
+
142 PEOPLE ON BOARD GERMANWINGS AIRBUS A320 THAT CRASHED IN SOUTHERN FRANCE
−
Pray for #4U9525 http://t.co/Il7RI24ffH
Twitter
He thinks that he knows more than our military because he claimed our armed forces are "a disaster."
+
We have to heal the divides in our country.
−
Politics

Figure 1. Examples of check-worthy and non-check-worthy claims

2.1.2 Evidence Retrieval

Evidence retrieval’s main goal is to find information regarding claim, such as various texts, tables, knowledge bases, images, relevant metadata etc. in order to aid the main fact-checking objective, to determine a claim’s veracity. Earlier efforts did not make use of extra evidence beyond the claim itself. Relying solely on the superficial claim patterns, without making any extra consideration and without the utilization of any kind of features, those approaches failed to identify well-stated misinformative statements including even machine generated claims. Recent developments on the domain of natural language processing have aggravated this issue. Machine generated text is sometimes considered to be more trustworthy than human written text. In conclusion, evidence is necessary not only to allow for verification but also to produce verdict explanations that persuade users of fact-checks applications.

Stance detection can be viewed as an instantiation of evidence retrieval, which typically assumes a more limited amount of potential evidence and predicts its stance towards a claim. For example, some approaches like [31], used news articles stance as evidence to predict whether the articles are supported, refuted, or merely reported a claim. Other approaches used whole documents as evidence in the form of sentences. This, however, introduces a lot of noise and redundant text data in the classification task, and thus, it is better to filter out sentences from evidence documents that do not participate in the veracity of a claim.

Another issue is that not all evidence information can be considered trustworthy. Most fact-checking pipelines collect evidence from trusted sources such as encyclopaedias like Wikipedia, or results that are harvested from search engines. With this method, veracity is thus defined as coherence with the evidence, and evidence as information that may be obtained from this source. For real-world applications evidence must be passed through a human, or a combination of a human checker, an automated mean, or a combination of both.

2.1.3 Verdict prediction

Given an unidentified claim as input, along with relating evidence retrieved, the stage of verdict prediction attempts to determine the veracity of a claim. Binary classifications consist one of the simplest and most used approaches, that labels a claim as true or false. When evidence is employed to help in the assessment of a statement, it is preferable to use the labels supported/refuted by evidence instead of the classic true/false label, since in many cased it has been observed that the evidence itself are not evaluated by the systems.

Finer-grained classification techniques are used in several iterations of the task. A straightforward modification would be to add a label indicating that there is insufficient data to determine whether the claim is true. Moreover, some datasets and pipelines follow an approach that was implemented for the first time by journalistic agencies, employing multi-class labels that represent the degree of truthfulness a claim may have (e.g. True, Mostly-True, Partly-True etc.).

2.1.4 Justification Production

The production of justification on the veracity of a claim has always been an important stage in the fact-checking pipeline. Fact-checkers and fact-checking automated systems need to convince the reader of their interpretation of the evidence and the steps that were made in order to reach a logical conclusion. Debunking claims by purely assigning a label to each claim, can introduce a “backfire” effect where the belief in the false claim is reinforced. This need is most important in automated fact-checking, where developers generate black-box components along the pipeline, whose decision-making methods are difficult to comprehend at first glance.

Justification production for a claim verification typically relies on four main strategies. First, attention weights can be utilized to highlight the noteworthy parts of the evidence. In cases like this, justifications typically consist of scores for each evidence token. Secondly, decision making processes can be designed to be comprehendible by human fact-checkers by relying on logic-based systems like in [9] for example. In this instance, the argument for the claim's truthfulness usually serves as the justification for its validity. The third strategy is to model the task as a form of summarization, where systems generate textual explanations of their decisions [32]. The fourth type consists of situations where certain the model's decision-making process is self-explanatory. This can be found in large language models for instance.

The basic idea of the justification stage is to show which pieces of text participated in the verdict of a statement’s validity. However, a justification must also examine how the retrieved relevant evidence was used and explain any assumptions or commonsense facts utilized. Finally, the logical process taken to reach a verdict, must be clear and easy to explain. However, merely presenting the evidence returned by a retrieval system can be seen as a very basic baseline for justification as it fails to explain the process used to reach a verdict.
Finally, the justification stage differs slightly from evaluation criteria. Robust evidence makes it easier to produce an accurate verdict. On the other hand, a good justification system accurately reflects the reasoning of the model through a readable and plausible explanation, regardless of correctness of the verdict.

2.1.5 Examples of automated fact checking pipelines

For the purposes of this section, we are going to present some pipeline systems that are used in automated fact-checking, to get a better grasp of how a claim is passed through various stages of the pipeline, for the final classification model to assess its validity.

In figure 2 [33], we present an example of an NLP pipeline consisting of three stages. The first one is claim detection, where claims are identified as appropriate for validation. The second stage is the evidence retrieval stage, where the web is searched for relevant sources supporting or refuting the claim. The third step is where a claim is assessed in terms of validity, based on the evidence retrieved. Evidence retrieval and claim verification can sometimes be considered as the same task. Claim verification can also be divided in two sub-tasks that can be dealt with separately or jointly: verdict prediction where statements are given authenticity labels;

and justification production, which calls for the presentation of arguments that support the model's decision to assign the corresponding label.

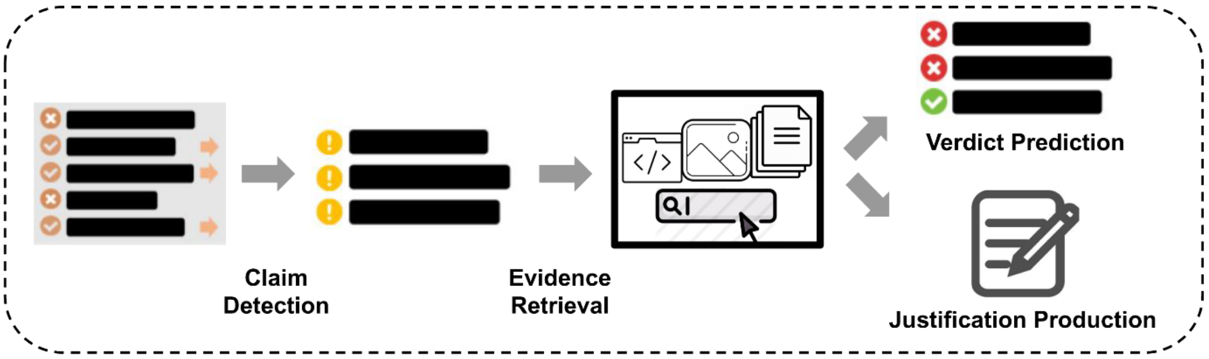


Figure 2. Example of fact checking pipeline

Figure 3 [34] illustrates a comprehensive fact-checking pipeline with the components mentioned above. It mainly consists of two parts: a claim detection component that searches for claims that require verification and looks for matches between claims when they are related to the same fact-check, and a claim validation component that retrieves documents and justifications that can be used as proof to verify a claim and completes the verification task, resulting in a verdict. The difference with traditional fact checking pipelines is firstly, the claim-matching component in the pipeline, and secondly, the rationale detection system. A short analysis for both is following below.

Claim matching, also referred as identifying previously fact-checked claims. When a claim is spotted in the claim detection component, claim matching aids in determining whether this claim exists in a database and secondly, whether it has been resolved in the past by a previous fact-check. The task is formulated as follows: Firstly, a check-worthy claim is used as input. Additionally, having a database that consist of already fact-checked claims, it consists in determining if any of the claims in the database is related to the input. If this is the case, the new claim would not need fact-checking again, as its validity has been already verified in the past.

Claim matching is normally considered to be a ranking task, where claims that belong to the database are ranked based on their similarity to the input claim. This task comes right after the claim detection step, to determine if a claim has not been seen in the past and can help avoid the need for a new pipeline implementation regarding a claim that has already been authenticated.

Rationale selection is a selection method which is utilized in order to select relevant sentences from the relevant documents retrieved, to get supporting evidence that participate in the claim's verification. Rationale selection consists of common natural language processing techniques, such as keyword matching, sentence similarity scoring and supervised ranking. Similar to document retrieval, this component of the pipeline attempts to use one of those methods, or a combination of them, to get a ranking score and select the top-k sentences as rationale. The parameter k is inserted manually by the user.

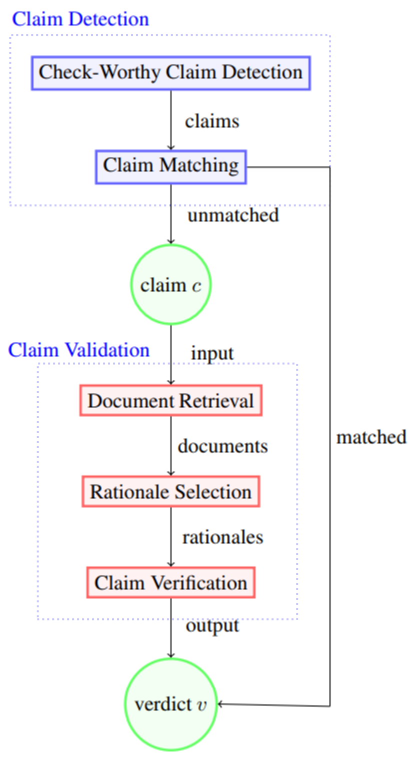


Figure 3. Fact checking pipeline with differentiations.

2.1.6 Check4Facts pipeline

In this section we intent to present the pipeline used in the Check4Facts[1] platform, along with a brief analysis of various features and components used for the classification task. The development of this framework established the foundations for the current thesis and thus it is important to include the previous approach along with its methods in order to get a better understanding of the methodology presented in the next sections of this study.

The solution to the fact-checking challenge, using Greek statements, include the basic components presented below. The final goal is to come up with a trained classification model for the fact-checking task.

- Search Engine: For each statement, Google is queried for relating sources reporting the statement.
- Harvester: For each resource, the title, the body, the most similar paragraph, and sentence (w.r.t. the claim) are harvested
- Feature extraction: Extract training features by encoding statements and relevant harvested data as feature vectors.
- Model selection: Evaluate a series of classifiers in order to find the most accurate one, as well as its best configuration.
- Model training: Train the best performing classifier using its optimal parameter configuration on all available training data.

The models are then deployed following the training process, with the goal of successfully classifying new unseen data. The subsequent part examines the deployment phase, as:

[1] https://check4facts.gr/

- Search Engine: Google is queried for resources related to the test statement.
- Harvester: Again. The title, body and the most similar paragraph and sentence of each resource are harvested (w.r.t. the claim)
- Feature extraction: Extract test features with the same manner as in the training phase.
- Classification: Utilize the trained model to get a prediction about the test statements validity. Input its vector to the model and get a binary verdict.

In the check4facts pipeline, a sentence can be represented with the use of a series of various features which are useful for the classification task. Those features are generated using NLP techniques, such as sentiment analysis and pre-trained word vectors. The text inputs that undergo this procedure are the main statement that will be refuted or validated, the titles of the web resources, the bodies of the web resources and the most similar paragraph and sentence found on the web. A web resource consists of the content that was harvested from the web, when given a claim, that holds relevant information in order to determine a claim’s veracity.

Some of the NLP methods used to translate the text inputs into a format that is comprehensible for the machine learning models, are presented below. We will not elaborate extensively on this topic as this is not the purpose of this section.

- Word embeddings: A model available on spaCy which is pretrained on the Greek corpus offers embeddings of high quality. The acquired embeddings are of size three hundred.
- Similarity feature: A function is implemented that computes the semantic similarity between an input sentence and a statement utilizing the corresponding spaCy embeddings. This feature ranges from 0.0 to 1.0 denoting entirely dissimilar or equivalent input pairs respectively.
- Subjectivity score: This feature represents an overall subjectivity score for input text, expressing how objective or subjective it is.
- Objective terms: Represents the number of objective terms in the input text.
- Subjective terms: This is symmetrical to the previous feature but w.r.t the subjective terms.
- Polarity score: This feature assigns an overall polarity score to input text, expressing how positive or negative its sentiment is.
- Negative terms: Represents the number of negative terms in the input text.
- Positive terms: This is symmetrical to the previous feature but w.r.t the positive terms.
- Emotion scores: Information about a specific emotion as well as its intensity is extracted with the use of a Greek lexicon. There is an intensity scale for a series of basic emotions. These emotions include ‘Anger’, ‘Disgust’, ‘Fear’, ‘Happiness’, ‘Sadness’ and ‘Surprise’. Each term of the lexicon is annotated for each of those emotions by four annotators, where annotations are integer values between 1 and 5, indicating the intensity of the emotion the term expresses. Emotion scores are a series of minimum, average and maximum values, regarding each emotion, for a specific text.
- Emotion terms: This feature produces an integer value, which responds to the number of input terms annotated of expressing an emotion e. Terms that count towards this feature are matched inputs terms that have a mean emotion e score greater than a defined threshold t, constituting a hyperparameter in our setting.

The classification algorithms are demonstrated below:

- **Naïve Bayes**: The Naïve Bayes classifier models the probabilistic relationship between the training characteristics and each class by operating on top of the posterior probabilities as outlined in the Bayes theorem. More specifically, the algorithm determines which class has the maximum posterior among them by computing the posterior probability for each accessible class given the attributes of a new test sample.
- **K-Nearest Neighbors**: KNN is a non-parametric classifier that works by firstly computing the distances between data point that is to be classified and all available training data and identifying a specified number of data points (K) closest to it. The term "nearest neighbours" refers to these data points. Next, among the class labels of its neighbours, it assigns the sample to the most common class label.

- **Logistic Regression (LR)**: A logistic function is used in the logistic regression (LR) statistical model to represent a dependent variable that is binary. This is accomplished by utilising maximum-likelihood estimation and learning the model's coefficients from training examples. In terms of input, the LR model merely models the probability of output. On the other hand, by selecting a threshold value and categorising samples according to their matching probability value, it can be utilised as a binary classifier.

- **Support Vector Machines (SVM)**: The SVM classification technique maximises the distance between examples that correspond to various class labels by mapping training examples to data points in space. This is accomplished by identifying an optimal discriminative surface during the algorithm's training phase. New test examples are then mapped into the same space and their class label is predicted this correspondence to this surface.

- **Multi-layer Perceptron (MLP)**: MLP is considered the most common type of neural network. It seeks to minimise the classification error by determining the weights between its connected neurons so that the classification error is minimized. The process of classifying a new test sample involves feeding its feature representation into the perceptron's input, which is tasked with predicting its class label.

- **Decision Trees (DT)**: DT is a classification technique that makes use of a decision tree structure to predict a sample's label based on observations of it. Class labels are represented by leaves in these tree structures, and feature conjunctions that result in those class labels are represented by branches.

- **Random Forests (RF)**: RF is a classification ensemble learning technique that works by building a large number of decision trees during training. These decision trees are applied to new test cases, each of which predicts a class label. New samples are classified by the algorithm according to the majority class label among the predictions of each individual tree. It is also known that the RF technique can mitigate the overfitting predisposition of simpler decision tree algorithms.

- **Extra trees (ET)**: The ET algorithm is a classification technique related to Random Forests (RF). It operates by fitting multiple randomised decision trees (also known as extra-trees) on different subsamples of the training instances. Averaging is used to increase prediction accuracy and manage overfitting.

2.2 Language model-based fact verification

In the digital age, swiftly spreading misinformation necessitates not only the verification of claims but also the provision of clear explanations for these verifications. Such explanations are crucial for building trust within the audience, as lack of them often leads to distrust in fact-checking results. For this reason, large language models have drawn significant attention due to their outstanding reasoning capabilities and extensive knowledge repository, positioning them as superior in handling various natural language processing tasks compared to other language models. In previous sections, we covered the basis of fact-checking pipeline components. Here, we aim to present related work in terms of integrating the LLM model into various stages of the fact-check pipeline. This section presents research made in that direction.

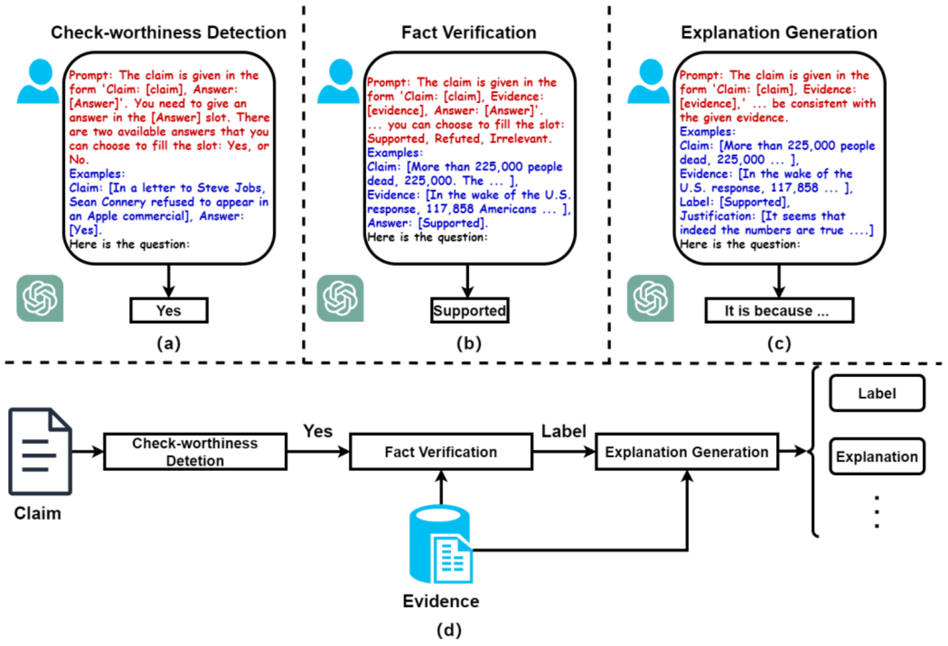
2.2.1 Language models for general fact-checking subtasks

The study conducted in [44], aims to evaluate various large language models in tackling some basic fact-checking subtasks. In general, experiments demonstrated that LLMs achieve superior performance against pre-trained and state-of-the-art low-parameter models in most cases. However, they encounter challenges in handling Chinse fact verification tasks and the corresponding fact-checking pipeline due to language inconsistencies and hallucinations.

In order to examine the performance of LLMs on fact-checking tasks, LLMs were utilized to solve the task of each component in the pipeline, and ultimately the whole pipeline. The first step was the design and selection of proper prompts for each task both manually and automatically. Data was then preprocessed to fit the prompts and to conduct the experiments. The framework for the evaluation process is displayed in figure 4. Figure (a), (b), and (c) are frameworks of our experiments in Check-worthiness Detection, Fact Verification, and Explanation Generation tasks, respectively. Figure (d) denotes the pipeline Fact-Checking task in detail.

LLMs like ChatGPT were utilized to generate several prompts and templates for each task. Figure 5 shows the some of the prompts designed for each task. To evaluate the performance of LLMs on each subtask of fact-checking, three validation sets of fact-checking benchmarks were employed.

The result of this research was that the LLMs are capable of effectively dealing with the fact-checking subtasks effectively. They can comprehend the requirements of each task and leverage their knowledge to solve them to some extent. Additionally, LLMs are able to retrieve relevant and meaningful evidence by themselves to verify the truthfulness of claims to some extent, and although there are still some challenges that are presented, such as hallucination. The outputs of LLMs are not always in the expected format, which is a big challenge when it comes to evaluating the results, especially for classification tasks. It has also been observed that LLMs are better at dealing with English data, but the face challenged when it comes to handling data in other languages, such as Chinese.

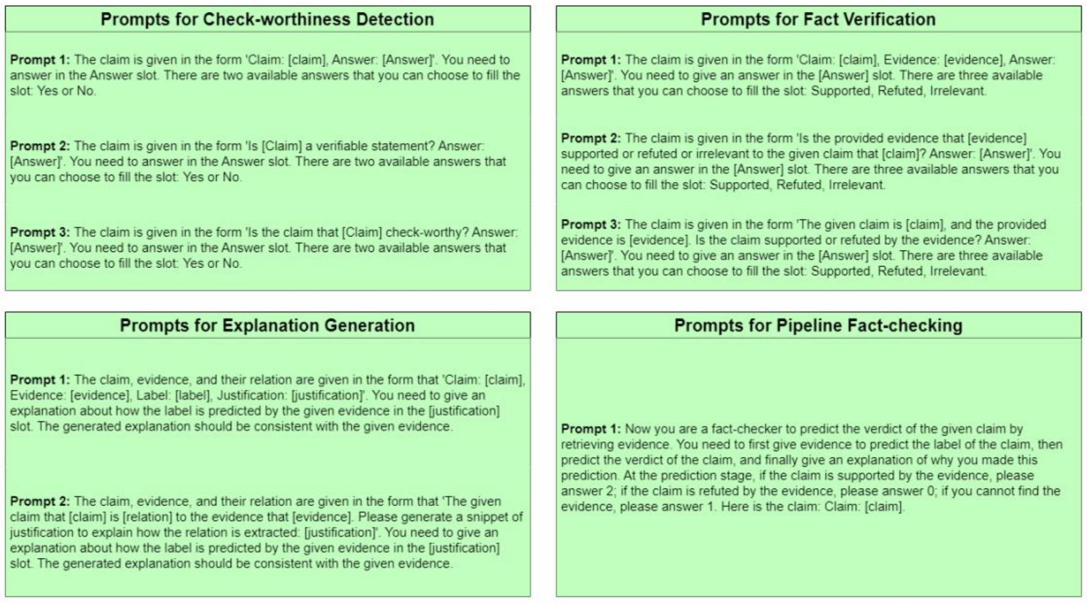


Prompt: The claim is given in the form 'Claim: [claim], Answer: [Answer]'. You need to give an answer in the [Answer] slot. There are two available answers that you can choose to fill the slot: Yes, or No.
Examples:
Claim: [In a letter to Steve Jobs, Sean Connery refused to appear in an Apple commercial], Answer: [Yes].
Here is the question:

Prompt: The claim is given in the form 'Claim: [claim], Evidence: [evidence], Answer: [Answer]'. ... you can choose to fill the slot: Supported, Refuted, Irrelevant.
Examples:
Claim: [More than 225,000 people dead, 225,000. The ...],
Evidence: [In the wake of the U.S. response, 117,858 Americans ...],
Answer: [Supported].
Here is the question:

Prompt: The claim is given in the form 'Claim: [claim], Evidence: [evidence],' ... be consistent with the given evidence.
Examples:
Claim: [More than 225,000 people dead, 225,000 ...],
Evidence: [In the wake of the U.S. response, 117,858 ...],
Label: [Supported],
Justification: [It seems that indeed the numbers are true ....]
Here is the question:

Figure 4. Fact checking pipeline with an intergated LLM



Prompts for Check-worthiness Detection

Prompt 1: The claim is given in the form 'Claim: [claim], Answer: [Answer]'. You need to answer in the Answer slot. There are two available answers that you can choose to fill the slot: Yes or No.

Prompt 2: The claim is given in the form 'Is [Claim] a verifiable statement? Answer: [Answer]'. You need to answer in the Answer slot. There are two available answers that you can choose to fill the slot: Yes or No.

Prompt 3: The claim is given in the form 'Is the claim that [Claim] check-worthy? Answer: [Answer]'. You need to answer in the Answer slot. There are two available answers that you can choose to fill the slot: Yes or No.

Prompts for Fact Verification

Prompt 1: The claim is given in the form 'Claim: [claim], Evidence: [evidence], Answer: [Answer]'. You need to give an answer in the [Answer] slot. There are three available answers that you can choose to fill the slot: Supported, Refuted, Irrelevant.

Prompt 2: The claim is given in the form 'Is the provided evidence that [evidence] supported or refuted or irrelevant to the given claim that [claim]? Answer: [Answer]'. You need to give an answer in the [Answer] slot. There are three available answers that you can choose to fill the slot: Supported, Refuted, Irrelevant.

Prompt 3: The claim is given in the form 'The given claim is [claim], and the provided evidence is [evidence]. Is the claim supported or refuted by the evidence? Answer: [Answer]'. You need to give an answer in the [Answer] slot. There are three available answers that you can choose to fill the slot: Supported, Refuted, Irrelevant.

Prompts for Explanation Generation

Prompt 1: The claim, evidence, and their relation are given in the form that 'Claim: [claim], Evidence: [evidence], Label: [label], Justification: [justification]'. You need to give an explanation about how the label is predicted by the given evidence in the [justification] slot. The generated explanation should be consistent with the given evidence.

Prompt 2: The claim, evidence, and their relation are given in the form that 'The given claim that [claim] is [relation] to the evidence that [evidence]. Please generate a snippet of justification to explain how the relation is extracted: [justification]'. You need to give an explanation about how the label is predicted by the given evidence in the [justification] slot. The generated explanation should be consistent with the given evidence.

Prompts for Pipeline Fact-checking

Prompt 1: Now you are a fact-checker to predict the verdict of the given claim by retrieving evidence. You need to first give evidence to predict the label of the claim, then predict the verdict of the claim, and finally give an explanation of why you made this prediction. At the prediction stage, if the claim is supported by the evidence, please answer 2; if the claim is refuted by the evidence, please answer 0; if you cannot find the evidence, please answer 1. Here is the claim: Claim: [claim].

Figure 5. LLM made prompts for each fact checking stage.

2.2.2 LLMs for the claim-matching stage of fact-checking

In [47], the study explores the potential utilization of large language models to support the claim matching stage in the fact-checking procedure. It further supports that when fine-tuned appropriately, LLMs can effectively match claims. The framework that they proposed, called FACT-GPT, could benefit fact-checkers by minimizing redundant verification, support online platforms in content moderation, and assist researchers in the extensive analysis of misinformation from a large corpus. In figure 6, we present an overview of the framework, aimed at assisting the claim matching stage in the fact-checking pipeline.

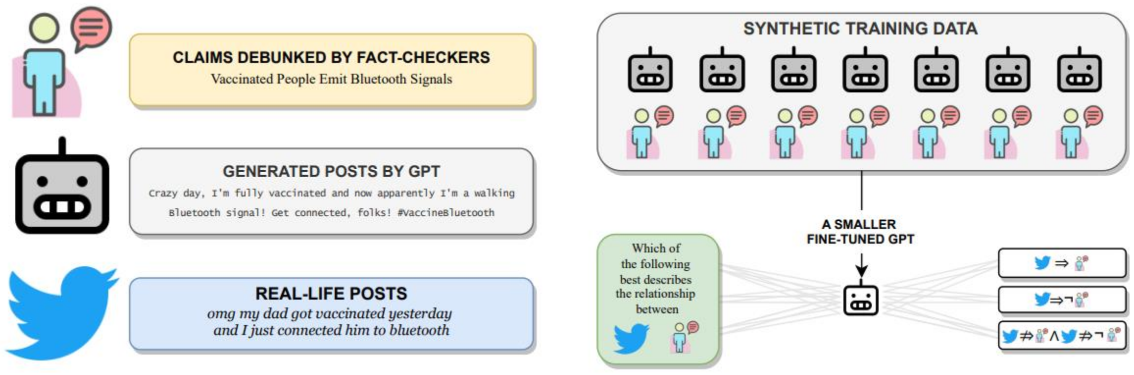

CLAIMS DEBUNKED BY FACT-CHECKERS
Vaccinated People Emit Bluetooth Signals
GENERATED POSTS BY GPT
Crazy day, I'm fully vaccinated and now apparently I'm a walking Bluetooth signal! Get connected, folks! #VaccineBluetooth
REAL-LIFE POSTS
omg my dad got vaccinated yesterday and I just connected him to bluetooth
SYNTHETIC TRAINING DATA
A SMALLER FINE-TUNED GPT
Which of the following best describes the relationship between

Figure 6. Illustration of the usage of a LLM during claim matching

As the above figure suggests, a small LLM was fined-tuned on synthetic training data that are that are generated from LLMs, with the help of a dataset made of false debunked claims by professional fact checkers, regarding the COVID-19 pandemic. The fine-tuned model's main purpose was to be evaluated in the textual entailment task, in a real-life dataset consisting of twitter posts. Textual entailment involved categorizing relationships between pairs of statements into three unique classes: Entailment, Neutral, and Contradiction. A pair is classified as Entailment if the veracity of Statement A inherently implies the truth of Statement B. The pair is labeled as Neutral if the truthfulness of statement A doesn't affirm or deny statement B's truth. It's identified as Contradiction if Statement A's truth infers that Statement B is false.

In conclusion, the research demonstrated that LLMs have the capacity to comprehend entailment relationships between social media posts and debunked claims. Small, fine-tuned LLMs can also perform in the same levels as larger models can, thereby offering a more accessible and cost-effective solution without decreasing the degree of quality. On the other hand, the drawbacks reported in categorizing posts that contradict already debunked claims.

2.2.3 LLMs in the justification stage of fact-checking

While large language models excel in text generation, their capability for producing faithful explanations in fact-checking remains underexamined. [46] investigates LLMs' ability to generate such explanations. They propose a method framework called Multi-Agent Debate Refinement (MADR) which employs multiple LLMs as agents with diverse roles in a process aimed at enhancing faithfulness in generated explanations.

MADR utilizes the debate for generating feedback to be employed in subsequent refinement stages (debate methodology is defined as the process where multiple language model instances or agents individually propose and jointly debate their responses and reasoning processes to arrive at a single common answer).

The advantages of this method are the valid identification of errors, the generation of accurate feedback, and lastly, the thorough analysis of explanations both with and without knowledge of predefined error types, that produces an explicit rationale.

2.2.4 LLMs as knowledge bases stage of fact-checking

In this section, we are going to analyze the use of language models as knowledge bases in the already established fact checking pipeline. Unlike previous knowledge, we describe language models as a component that can be used in earlier pipeline steps. Until recently, traditional fact-checking pipelines accessed additional knowledge needed for the determination of a claim validity, via external knowledge bases (i.e., Wikipedia). They also made use of additional modules such as document retrievers and evidence selectors for retrieving the appropriate evidence and verification data that are then summarized in {claim, [evidences]} pairs and predict a final verification label. In [7] , The proposed pipeline replaces external knowledge bases as well as the use of additional modules with a pre-trained language model, named BERT.

The main idea of this study is as follows:

- Mask a token from the claim depending on the component of the claim we wish to verify. For example, the claim “Thomas Jefferson founded the University of Virginia after retiring” becomes “Thomas Jefferson founded the University of [MASK] after retiring.”
- Get the predicted token from the language model. In this case, the predicted token was “Virginia.”
- If the predicted token matches the masked token, then the claim is supported. Otherwise, it is not.

The pipeline steps of this method, comes across two limitations. Manual masking of claims and naïve validation of the predicted token that fails to deal with synonyms and other semantic variants of the answer. To solve those, two pipeline steps were introduced:

- Automated masking: The last-named entity in the claim is masked, which is identified using a Named-Entity-Recognition model (NER) from spaCy. The decision to mask named entities is to ensure that the token masked utilizes the knowledge embedded in language models.
- Verification Using Entailment: To advance through naively matching predicted and masked tokens, a textual entailment model is employed to validate the LMs predictions as the third step above suggests. Textual entailment models predict the directional truth relation between a text pair (i.e., “sentence an entails b” if a human reading a would infer that b most likely true).

Detailed steps for the end-to-end model are as follows:

- The last-named entity of the claim (which was retrieved by the NER model) is masked.
- The language model produces the predicted token and fills the masked claim accordingly to produce the “evidence” sentence.
- Using outputs from the final layer of the pretrained entailment model, entailment features are extracted using the generated "evidence" phrase and the claim.
- Input the entailment features into a multi-layered perceptron (MLP) for final fact-verification prediction.

The results reported in this study, suggest that masking techniques as the one previously examined, provide better results for extracted encoded knowledge stored in LMs. Another insight is that the way the claims should be masked is significantly determined by the LMs' pre-training. Additionally, language models used as sources of information for fact-checking, are as effective as standard baseline models. Although this experiment is not enough to beat the state-of the-art traditional pipelines, this approach shows strong potential for improvement and future work can explore stranger models for generating evidence or improving the masking methods.

2.2.5 Self-Checker

In [45], a new plug-and-play framework was introduced named Self-Checker, which has the ability to harness LLMs for efficient and rapid fact-checking in a few-shot manner. The results of this study revealed promising result for the use of large language models in fact-checking. When this framework is compared to state-of-the-art fine-tuned deep learning models, there is still room for improvement. Nevertheless, adopting an LLM approach could be a step to the right direction for the future of fact-checking research.

Self-Checker is a training-free framework and contains a set of plug-and play modules. A claim processor, query generator, evidence seeker, and verdict counselor. The illustration of Self-Checker is depicted in Figure 7. It is designed to assess the factuality of textual inputs and employs a policy agent that makes decisions about future actions based on a set of choices. Each of his modules is implemented by prompting an LLM through prompts that have been constructed in detail for this purpose.

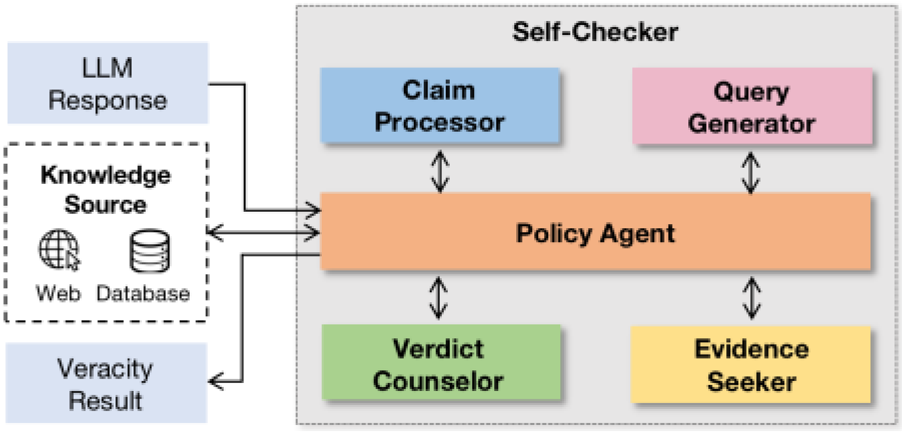


Figure 7. Illustration of the Self-Checker pipeline

The frameworks components are shortly explained below:

- **Policy agent**: This module determines the future actions of the system, given a set of predefined actions. These actions include: (1) calling the claim processor to process the complex input, (2) requesting search queries from the query generator, (3) retrieving relevant passages from a knowledge source based on the generated search queries, (4) utilizing the evidence seeker to extract evidence sentences for a claim from the retrieved passages, (5) requesting the verdict counsellor to provide a verdict prediction based on the gathered evidence, and (6) sending the final conclusion to the users.

- **Claim Processor**: This component is responsible for identifying claims for verification from the input text. Making use of the text generation capabilities of LLMs, we define the task of obtaining a set of claims to verify as a generation task. Given a text t as in put, the claim processor generates a set of claims {c1, c2, ..., cm} that are included in t and need to be verified. If a specific claim for verification has been provided, the claim processor can also break it down into a set of simpler claims. Every generated claim should represent the same information as the original input, which needs to be confirmed.

- **Query Generator**: In order to verify a claim, it is essential to retrieve relevant information from an external knowledge source. Given a claim c, the query generator prompts an LLM to generate a set of search queries q = {q1, q2, ..., qk} for the purpose of information retrieval. These generated queries are then used to obtain relevant passages {p1, p2, ..., pk} from a knowledge source.

- **Evidence Seeker**: The evidence seeker aims to identify evidence sentences for a given claim from the retrieved passages. Given a claim c and the set of retrieved passages {p1, p2, ..., pk}, the evidence seeker returns a set of selected sentences {s1, s2, ..., sn} that indicate the veracity of the claim. Again, for this task, an LLM is carefully prompted with task instructions, the claim, in-context examples and retrieved passages.

- **Verdict Counselor**: The primary objective of the verdict counselor is to analyse the set of claims that are under examination, along with the corresponding evidence sentences for each claim. This module is responsible for predicting the veracity of the entire set of claims. By examining the relevant evidence per claim, the verdict counselor determines the authenticity of each claim and assigns a label, such as supported, partially supported, or refuted. The labels are then aggregated to obtain the final result of the entire set. The veracity labels used by the verdict counselor are predefined (supported/partially supported/not supported/refuted). To accomplish this process, an LLM is prompted with specific instructions.

2.2.6 Hiss Prompting method

Previous approaches regarding fact verification with the use of language models (PLMs), have made use of pretrained language models in fake news related tasks. For example, in [2] there is a direct use of the PLM's internal knowledge for fact verification. The knowledge base is stored as a parameter inside the model. In [3], the authors implement a retrieval-augmented support to endow language models with document retrieval capability, which applies for selecting relevant evidence in fact extraction and verification. Other approaches as presented in [4] utilize LLMs and their few-shot capability to assess the claim's factuality based on the perplexity of evidence-conditioned claim generation. The main limitations of claim verification with the use of LLMs, are the omission of necessary thoughts and fact hallucination. For these reasons, the authors in [5] propose a Hierarchical Step-by-Step prompting method. The main idea behind this is to direct the LLM to separate a claim into several subclaims and then verify each of them via multiple questions answering steps progressively.

In order for the limitations mentioned above to be exceeded, this method prompts LLMs to thoroughly generate all explicit and implicit points that are check-worthy given a certain claim. Moreover, hallucinations are also addressed that way, by providing relevant and up-to-date contextual information to LLM as external knowledge, assuming that hallucinations are most likely a result from the lack of knowledge on the necessary context.

The first stage if HiSS, is to focus on instructing the model to capture all the explicit points in the original claim and decompose them into subclaims. Specifically, LLM is prompted with various verification examples, to illustrate the entire verification process, followed by the test claim to be checked. The demonstration examples exhibit to LLM hot to break down a claim into a series of subclaims, that cover all check-worthy points expressed. The demonstration examples vary in their complexity, with some claims undergoing deep decomposition and more complex claims being decomposed into a few more subclaims. The LLM follows the demonstrated decomposition approach in accordance with the complexity of the input claim. Therefore, the total number of subclaims is determined by the LLM automatically.

In the second stage, the model verifies each subclaim that was generated from the previous step. In particular, the model produces a series of probing questions corresponding to an implicit point in a subclaim. The number of the probing questions is also determined by the LLM in accordance with the demonstration example. To generate the questions, a progressive approach is implemented, in order to adjust the subsequent question generation based on the answers to previous questions and the newly acquired information. As these questions get more focused and in-depth, the probing inquiries make it easier to analyse the subclaims thoroughly.

The next stage after creating a question is to get the relevant response from LLM. To mitigate fact hallucination, LLM is asked whether or not it possesses the required confidence to answer a question. The answer can be positive or negative. In case of a negative answer, relevant external information is gathered from the web. The model will cease to further generate an answer for a question and wait for the user to input the necessary information that was acquired from external sources. If the answer is positive, the model does not pause and proceeds to generate an answer to a question.

Following the verification of each subclaim, the LLM is able to provide a final estimate. At this point, it outputs a claim label, classifying the latter with a corresponding label from a corresponding dataset.

2.3 Language model-based claim detection on non-English language

This section investigates to what extent pre-trained language models can be used for automated claim detection for fact checking in a low resource setting, which is a very common scenario for non-English fact-checking domains. Although [6] does not cite fact checking classification methods and tools, to assess the veracity of a claim, it is important to delve deeper into foreign language domains, to get a clearer understanding of a language model's comprehensive ability to classify claims in languages other than English.

Other attempts of claim detection in languages other than English have also been conducted. ClaimRank for example, is a claim detection system that supports both Arabic and English. In previous sections, we concluded that LLMs often struggle in managing data written in languages other than English, like Chinese.

2.3.1 A case study using Norwegian Pre-Trained language models

The purpose of this study is to conclude that large language models could be successfully employed to solve the automated claim detection problem. The choice of the model is dependent on the desired end-goal, but the overall analysis shows that the language models are less sensitive to changes in claim length than the SVM model.

The first stage of the pipeline is to fine-tune a series of models on a small dataset from a Norwegian non-profit fact-checking organization. The claims of the dataset consist of claims manually annotated with labels reflecting their check-worthiness. Following a preprocessing stage, the final dataset includes 4116 claims, across six different labels. These labels are produced during a fact-checking procedure. For a claim to be considered for fact-checking, it must be supported by corresponding information; it should not contain any kind of normative statement or prediction about the future. A claim should also be relevant to the majority of people and controversial to some extent.

Moving forward from this point, a binary classification is defined, with the labels "Discarded" and "Upheld". "Discarded" class refers to claims that have been rejected and there is no further need to investigate its validity. The "Upheld" class on the other hand, refers to claims that have been deemed suitable for a fact-checking procedure. The first class has 2810 claims, whereas the second class contains 1306.
As mentioned above, for the classification purposes, a series of models were fine-tuned using a TensorFlow-based model for sequence classification from the HuggingFace transformers library. The baseline model is a SVM classifier with tf-idf features implemented using Scikit-learn library.

Results from this study suggest that all the language models were superior to the baseline system in terms of F1 score. For the case of precision, the baseline system outperforms the LMs, but recall and F1 are extremely poor. Another significant reflection of the result the performance of the NorBERT2 model in terms of recall. Overall, the LMs surpass the SVM model in terms of performance. The study also suggests that choosing the right language model depends on the specified task. For example, having different priorities like high recall score or overall performance, may lead one to select an alternative model.

Lastly, this research comes with certain limitations. The behaviour of the models for instance, might be affected by the skewed distribution of classes in the dataset, where the majority class exceeds the minority class by more than twice its number. Future work is needed in terms of reproducing the results and conducting further analysis, but nonetheless, the findings of this experiment were encouraging and showed promising results for the future endeavours.

2.4 Traditional text representation methods

Previous work regarding text representation methods have been implemented with the use of external sources from the web. In [13], the text representation is built from a claim and the corresponding relevant snippets and web pages. After calculating three similarities between the claim and the snippet or claim and the web page. The embeddings of the text are then calculated based on the average of the embeddings of its words, with the use of pre-trained embeddings.

Other, more limited approaches include subject-predicate-object triples obtained by information extraction [10]. The most common example is (Obama, born in, Kenya) where “Kenya” is the critical value. The assumption of such a structure is particularly important to identify alternative values for the slot in question (e.g., “Hawaii”, “America”). However, these approaches cannot manage many sorts of claims, which are often in the form of long sentences or entire paragraphs.

Another approach is the use of knowledge graphs, which provide a collection of structured canonical information about the world, in a machine-readable format. The main idea is to identify and retrieve the element in the graph that provides the information supporting or refuting the claim at question. Once the relevant triplet is found, the classification label is computed through a rule-based approach that considers the error between the claimed values and the retrieved values from the graph. However, the assumption that the true facts relevant to the claim are present in those graphs, is not always the case [16]. One alternative approach is to consider the graph topology in order to predict how likely a claim that is expressed as an edge to a graph is to be true [11].

Recent studies, like [1] made use of word embeddings with the intention of representing text in an n-dimensional space that modifies human language into a machine comprehensible vector. This numerical valued vector that is generated, is able to capture the text’s morphology and syntax and semantic information about the text.

In this section, we will examine some of the most common and traditional methods that translate text into a series of numerical features that can be calculated directly.

2.4.1 Bag-of-words models

The basic idea of the bag-of-words model that were implemented in relating research, is to represent text as a series of words without considering the order in which those words appear on the text. Those method include BOW (bag of words), TF-IDF (term frequency-inverse document frequency), LSA (latent semantic analysis) and more. The most popular model among them, are presented below:

BOW: In simple terms, bag of words utilizes a technique known as count vectorization. In its essence, with this method we are able to represent pieces of text by counting the number of words that appear on this text, regarding a vocabulary, which encompasses a vast collection of unique words. Despite its limitations, such as the model’s negligence for the word ordering, high dimensionality issues and the sparsity of the vectors produced, bag of words is a foundational technique that serves as a baseline for more advanced methods in text analysis.

TF-IDF: TF-IDF is able to resolve the limitations that are immersed from the bag of words model. It is an extension of the previous model that weights the terms based on their importance across the entire corpus. Calculation of the TF-IDF score (or weight) for each word in the document is as follows:

$$tf\_idf(w,d,D) = tf(w,d) \times idf(w,D)$$

where:

$$tf\ (w,d) = Freq(w,d)$$

$$idf(w,D) = \log \frac{|D|}{N(w)}$$

$Freq(w, d)$ is a function that counts the frequency of a word in a document, $|D|$ indicates the total number of documents, and $N(w)$ represents the total number of documents that the word w appears in. There is a TF-IDF representation for all the words in the document, and the number of words is equal to the number of words represented by TF-IDF.

2.4.2 Shallow Window-Based Methods

The shallow window-based methods differ from the bag of word model approaches in an important way. The bag of word models fails to capture the semantic distance between words. By generating word vectors via shallow window-based methods, we initialize a fixed-size "window" of words around each target word, capturing local context. Its features are based on co-occurrences of words within these windows, capturing more semantic information about how words are used in relation to one another. Shallow window-based approaches can solve the issue of dimensionality limitations and lack of semantics caused by the bag of words techniques because of the independence that is introduced with each word. In this section we are going to present the most popular implementations of this approach.

Word2Vec: Word2Vec consists of a combination of two related model architectures that are used to produce word embeddings. The first one is named CBOW architecture, and predicts the current word based on the context. The second one is named the Skip-gram architecture, and it predicts surrounding words given a current word. These embeddings generated by this method are dense vector representations of words in a continuous vector space, where semantically similar words are mapped to nearby points.

The first architecture is called CBOW (Continuous bag of words model). With the use of CBOW, all words get projected into the same position (their vectors are averaged). This architecture is called bag of words, as the order of the words does not influence the projection. In addition, this model also utilizes words from the future. This model is also named continuous, because unlike standard bag-of-words models, it uses continuous distributed representation of the context [27]. In simple terms, the CBOW takes as input (when given a window size) a collection of words surrounding the target word. This collection is then one-hot encoded and averaged in the hidden layer, to form a single vector representation.

In terms of output, there is a layer that consists of a softmax function, that predicts the probability distribution over the entire vocabulary. The word with the highest probability is the model's prediction for the target word. The model is trained to minimize the cross-entropy loss between the predicted probability distribution and the actual one-hot encoded target word.

The Skip-gram model is similar to CBOW, but instead of predicting the current word based on its context, it tries to maximize the classification of a word based on another word in the same sentence. Specifically, this model takes advantage of each current word as an input to a log-linear classifier with a continuous projection layer and predicts words within a certain range (equal to the window size) before and after the current word. As the window size increments, it leads to an improvement in the quality of the resulting word vectors, but it also increases the computational complexity. Since the more distant words are usually less related to the current word that the ones closer to it, this model assigns a smaller amount of weight to distant words by sampling less of those words during training.

In figure 8, we present the two model’s architectures. As already stated, the continuous bag of words (CBOW) architecture predicts the current word based on the context, and the skip-gram predicts surrounding words given the current word.

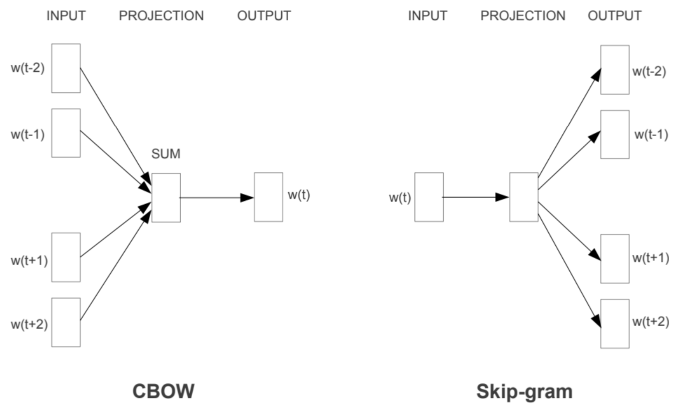


Figure 8. Word2Vec architecture

GloVe: All unsupervised methods for learning word representations rely on the statistics of word occurrences in a corpus as their primary source of information. While many relevant methods are currently in use, it is still unclear how meaning is derived from these statistics and how the resulting word vectors might represent that meaning. That was the main reason that a different model for word representation is introduced, which is called GloVe. GloVe stands for Global Vectors, because the global corpus statistics are captured directly by the model [28].

The basis of GloVe stems from the domain of global word frequency statistics, which explains the semantic information of words by modeling the contextual relationship of words. Its core idea is that words with similar meanings often appear in similar contexts. It takes advantage of co-occurrence matrices A ,where each element A_{ij} in this matrix represents how often a word j appears in the context of a word i.

In simple words, the main goal of GloVe is to learn word vectors in a manner that their dot product will eventually equal the logarithm of the words' probability of co-occurrence. This is achieved by minimizing the following cost function:

$$\sum_{i,j=1}^{V} f(X_{ij})\left(w_i^T \widetilde{w}_j + b_i + \widetilde{b_j} - log\, X_{ij}\right)^2$$

Where w_i and $\widetilde{w}_j$ are the word vectors for the target and context words, b_i and $\widetilde{b_j}$ are the bias terms, and $f(X_{ij})$ is a weighing function that mitigates the influence of rare and frequent co-occurrences. The weighing function is presented below, where $X_{\max}$ and a are hyperparameters.

$$f(X_{ij}) \begin{cases} \left(\frac{X_{ij}}{X_{\max}}\right)^{\alpha}, & if\ X_{ij} < X_{\max} \\ 1, & otherwise \end{cases}$$

BERT: Bidirectional encoder representation from transformers or BERT, is a state-of-the-art natural language processing model. One of its primary uses is to generate rich, contextualized representations of text that can be used for a variety of natural language processing tasks. Its main innovative technique is based on the pre-training approach which includes masked language modeling (MLM) and next sentence prediction (NSP) which can capture a text’s expression and predict the next sentence of a text, respectively. With MLM, the goal is to mask some words in a sentence and let the model to learn how to predict these masked words. With NSP on the other hand, the model learns to predict if a given sentence pair is consecutive.

Unlike traditional models, that read text sequentially, BERT is bidirectional in nature. This means that BERT has the ability to read the entire sequence of words simultaneously from both directions, allowing it to capture a richer context for each word, improving the modules comprehension abilities of language and relationships between words.

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks— whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs.

BERT comes in different sizes, primarily BERT-Base and BERT-Large. BERT-Base has 12 layers, 768 hidden units, and 110 million parameters. BERT-Large has 24 layers, 1024 hidden units, and 340 million parameters.

In figure 9 we present the model’s architecture. BERT representations are jointly conditioned on both left and right context in all layers [29].

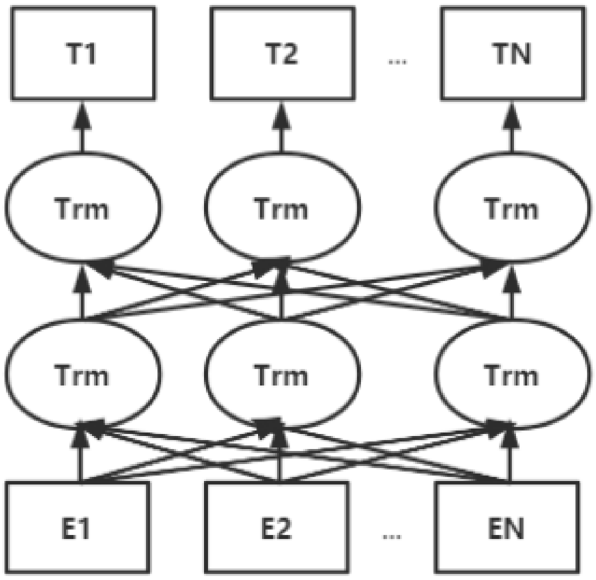


Figure *9. Berts model architecture*

2.4.3 Knowledge graphs

Knowledge graphs representation has one main function. To project entities and relationships in the knowledge graph into continuous low-dimensional vector spaces through machine learning technology, while maintaining the basic structure and properties of the original knowledge graph. Moreover, the learning process takes under consideration the knowledge graph’s global and local properties. In simple terms, the resulting entity and relation vector is a representation that carries semantic features, which can express semantic information efficiently.

A knowledge graph consists of entities, relationships, and triples. Entities are the nodes of the graph, which compose the network. They represent real-world objects, concepts, or things (e.g., "Barack Obama," "USA President"). Relationships are the edges of the graph which demonstrate the connections between entities (e.g., "Barack Obama" is the "President of" the "USA"). Triples are (subject, predicate, object) collections, also known as SPO triples. For example, ("Barack Obama", "was born in", "Hawaii"). Finally, an ontology specifies the kinds of interactions that can exist in a graph, defining the schema or structure of the knowledge network.

Knowledge graphs were very popular when they first emerged. They made the task of extracting structured data from unstructured data simpler, they provided a better comprehension of the links and context found in the data and had scalability potentials. Nowadays, its high maintenance and scalability costs, along with entity disambiguation shortcomings, resulted in the employment of alternative approaches other than knowledge graphs.

2.5 Text similarity metrics and techniques

In this section, we are going to present the most popular similarity techniques that are utilized to measure the relevancy between two pieces of text. It is crucial to choose a metric that is robust to high dimensional text embedding vectors, and also cost-effective, since this metric is going to be used multiple times, considering the fact that we have to deal with hundreds of claims.

In general, similarity is defined as the commonness between two text snippets. The greater the commonness, the higher the similarity and vice versa. Text similarity is an essential tool in many natural language processing tasks. Various measures of text similarity techniques have been proposed over the years. Most researchers divide text similarity measurement methods on the basis of statistics or corpus and knowledge bases, such as Wikipedia [25]. This categorization ignores the text distance calculation method, and only considers the representation of the text. In the recent years, the development of neural networks, have paved the way for other approaches such as semantic matching methods and graph methods. Text similarity not only accounts for the semantic similarity between texts but also captures the shared semantic properties of two words. For example, the words ‘King’ and ‘man’ may be related closely to one another, but they do not share semantic similarities. The words ‘King’ and ‘Queen’ on the other hand do. Thus, when we observe sematic similarity between words, we can safely assume that those words are related semantically. The semantic relationship including similarity is measured in terms of semantic distance, which is inversely proportional to the relationship.

2.5.1 Cosine similarity metric

In the field of vector space modeling, Cosine is widely used to measure the similarity between two vectors. Its calculation is very efficient, especially for sparce vectors, as only the non-zero dimensions need to be considered. As a fundamental component, cosine similarity has been applied in solving different text mining problems, such as text classification, text summarization, information retrieval, question answering and so on. In search engine cases, similarity between the user’s query and relating documents are sorted from the highest to the lowest one. The higher the similarity score between the document’s vector and the query’s vector, the higher the relevance between those two.

More specifically, cosine similarity (or dot product in relating literature), models a text document as a vector of terms. By this model, the similarity between two documents can be derived by calculating the cosine value between to document’s term vectors. This metric can be used with any form of text such a simple sentence, a paragraph, or even a whole document. Although it is considered one of the most popular techniques of measurement it does not come without drawbacks.

Cosine similarity, regardless of its advantages and the popularity that it meets, is still unable to fully capture a text’s semantic meaning. In many recorded cases, implementing the cosine similarity measurement between two vectors containing text embeddings, produces unreliable results. Syntax matching and semantic meaning are not always correlated, and that differences is often disregarded by this metric. In other domains, such as information retrieval systems, it may produce false results and cause a decline in performance.

Despite its flaws, cosine similarity remains highly effective measurement technique, mostly due to its normalization properties, its suitability for high-dimensional and sparse vectors, and adaptability in many applications. Ultimately, its effectiveness in capturing the similarity of content while also being computational efficient makes it one of the most widely used metrics in the natural language processing domain.

In document-query cases, a document can be represented as a term vector that the vector’s dimensions refer to the terms available in the document [23]. A document can be described as a vector form:

$$\vec{d} = (w_{d0}, w_{d1}, \dots, w_{dk})$$

Likewise, the query term can be described as a vector form as:

$$\vec{q} = (\, w_{q0}, w_{q1}, \dots, w_{qk})$$

where w_{di} and w_{qi} $(0 \leq i \leq k)$ are float numbers indicating the frequency of each term inside a document, while each vector's dimension corresponds to a term that is found in the document. Finally, given two arrays that contain text embeddings, the relevance between those two, can be defined by the formula below:

$$Similarity(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{k=1}^{t} w_{qk} \times w_{dk}}{\sqrt{\sum_{k=1}^{t} (w_{qk})^2} \cdot \sqrt{\sum_{k=1}^{t} (w_{qk})^2}}$$

2.5.2 Euclidean distance metric

Euclidean distance is a standard metric mainly employed for geometrical problems. It is the ordinary distance between two points and can be easily calculated in two and three-dimensional space. Euclidian distance is widely known for its clustering problem, including text clustering, but is also used as a text similarity metric. In computer science, it is utilized for the implementation of the k-means algorithm.

Euclidean distance is measured by calculating the square root of the sum of squared differences between the two vectors elements.

$$d_{euc}(d, q) = \sqrt{\sum_{i=1}^{n} (d_i - q_i)^2}$$

Where d and d are the two text embeddings vectors and d_i, q_i are the i-th element of the vectors.

2.5.3 Jaccard coefficient metric

The Jaccard coefficient, which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects, and in our case the embedding vectors. Jaccard is to solve the similarity through the set; when the text is relatively long, the similarity will be smaller. Therefore, when Jaccard is used to calculate similarity between two pieces of text, it is usually normalized at first.

The formula calculating the Jaccard coefficient is listed below:

$$J(d, q) = \frac{|d \cap q|}{|d \cup q|}$$

Where $|d \cap q|$ is the number of elements in the intersection of sets d and q and $|d \cup q|$ is the number of elements in the union of sets d and q.

2.5.4 Manhattan distance metric

The Manhattan distance can also be used to measure the distance between two real-valued vectors. Manhattan distance is calculated as the sum of the absolute differences between the two vectors which generally works only if the points are arranged in the form of a grid and the problem that we are working on gives priority to the distance between the points only along the grids, not the geometric distance. The similarity of two documents is obtained through Manhattan distance after one-hot encoding. The Manhattan formula is as follows:

$$D_{Manhattan(d,q)} = \sum_{i=1}^{n} |d_i - q_i|$$

2.5.5 Hamming distance metric

Hamming distance is a metric for comparing the two binary data strings. While comparing two strings of such kind and equal length, Hamming distance is equal to the number of bit positions in which the two bits are different. The Hamming distance between two strings, a and b, is denoted as d(a, b). It is used for error detection or error correction when data is transmitted over computer networks. It is also used in coding theory for comparing equal length data words. For binary strings A and B, this is equal to the number of times "1" occurs in binary strings.

2.5.6 Distribution distance metrics

Making use of a vector multidimensional position to calculate its length with another vector is a very efficient technique as we have seen in the above examples. However, those approaches do not come without limitations. On an overall basis, when we utilize length distance to calculate similarity, we come across two fundamental issues. Firstly, is suitable for symmetrical problems such as Sim(A,B) = Sim(B,A), but if we are operating in a question-answer basis, the corresponding similarity is not symmetrical. Secondly, there is a risk in using length and distance to judge similarity without knowing the statistical characteristics of the data.

In order to assess how similar two documents are to one another based on distribution; the distribution distance is employed to compare whether the documents are from the same distribution. JS and KL divergence are the most often used techniques to look at distribution distance.

Jensen–Shannon divergence is a method to measure the similarity between two probability distributions. It is also known as the information radius (IRAD) or the total divergence to the average. This metric is often used in combination with LDA (latent Dirichlet allocation) to compare the topic distribution of the new documents, with the already existing documents in the corpus and to determine which of them present similarities regarding their distribution. This is achieved through comparing the differences in distribution between the new and the old documents. Jensen-Shannon's distance gets smaller, the more similar the distribution is between two documents. Assuming that the two documents are following different distributions, the Jensen-Shannon divergence formula is as follows:

$$JS(P_1||P_2) = \frac{1}{2}\,KL(P_1|\left|\frac{P_1+P_2}{2}\right) + \frac{1}{2}\,KL(P_2||\frac{P_1+P_2}{2})$$

The KL() function in above, refers to the Kullback-Leibler divergence which is also known as relative entropy. It is a measure of different degrees of a probability distribution and a second reference probability distribution. Let $p(x)$ and $q(x)$ be two probability distributions with values of X, then the relative entropy of p to q is:

$$d(p||q) = \sum_{i=1}^{n} p(x)\log\frac{p(x)}{q(x)}$$

Finally, the word mover's distance, based on the concept of Optimal Transport, utilizes word embeddings to quantify the distance between words in the documents. It measures the minimum distance required for a word in one text to reach a word in another text in the semantic space, to minimize the cost of transporting text 1 to text 2. The main goal of this algorithm is to find the minimum cost required to transform one document into another by moving word distributions, where the cost is the distance between word embeddings.

Given two documents, D_1 and D_2, we define $\vec{w_i}$ as the embedding of the i-th word in the document. The distance $d(\vec{w}_i, \vec{w}_j)$ between words i and j is usually the Euclidean distance between their embeddings.

The first step is to compute the cost $C_{ij} = d(\vec{w}_i, \vec{w}_j)$ for all word pairs (i, j) where $\vec{w}_i$ belongs to D_1 and $\vec{w}_j$ to D_2 accordingly.

After the costs are calculated, the next step is to calculate the optimal transport problem, that deals with finding the most efficient way to minimizing the transportation cost from one text to another.

- Let T be the transport matrix where T_{ij} denotes how much of word i in D_1 should be transported to word j in D_2
- The main goal is to minimize the total transportation cost:

$$WMD(D_1, D_2) = \min_{T \geq 0} \sum_{i,j} T_{ij}\, C_{ij}$$

Which is subject to:

$$\sum_j T_{ij} = p_i, \forall i$$

$$\sum_i T_{ij} = q_j, \forall j$$

Where p_i and q_j are the normalized word frequencies of $D_1$and D_2 respectively.

2.6 Linguistic features in text

Extensive work has been made with the goal of creating features that are capable of providing enough evidence to support or refute a claim, in the fact-checking pipeline. Some of the most standard methodologies include linguistic features that try to capture the linguistic style of relevant to the claim articles. This is achieved with the use of language features, which is essentially a set of assertive and factive verbs, hedges, report verbs, subjective and biased words etc. [14]. The stance of the relating article is also considered as a feature in many cases. Whether the article reporting the claim is supporting it or not can make a claim more credible or less trustworthy. The reliability of a web source can also be considered a principal factor in assessing the credibility of a claim [15].

The latest feature extraction technique in terms of NLP related methodology is sentiment analysis of a text. Also known as opinion mining, it is a subdivision of data mining, and it refers to the practice of applying text analysis and natural language processing for the purpose of identifying, extracting, and analysing subjective information from textual sources. SA focuses on the challenge of classifying a given input text based on the polarity of its sentiment (positive, negative, or neutral). More advanced SA techniques look at whether the textual sources have any connections with emotional states such as fear, anger, happiness, and sadness. Another metric associated with sentiment analysis techniques is the subjectivity of a text. Instead of classifying text as being either positive, negative, or neutral, the text could be associated with a number on a pre-defined scale (e.g.,0 to 1) [12].

2.7 Deep learning approaches for text representation and modeling

Deep learning based neural network models have known considerable success in many NLP tasks, including learning distributed word, sentence and document representation, parsing, statistical machine translation sentiment analysis etc. The process of encoding text into continuous vectors in a high-dimensional space, also known as distributed text representation, along with the utilization of neural networks models, can reach satisfactory results in related tasks like sentiment classification and text categorization. Unlike traditional text representation methods such as one-hot encoding, distributed representations using deep learning techniques can capture semantic and syntactic information in detail.

In many recent studies relating sentence representation learning, neural network models are constructed upon either the input word sequences or the transformed syntactic parse tree. Among them, convolutional neural network (CNN) and recurrent neural network (RNN) are two popular ones. Convolutional neural networks show very promising capabilities in capturing local correlations along with extracting higher-level correlations through pooling. This empowers the CNN to model sentences naturally from consecutive context windows.

As for the RNN model, its nature as a sequence model, makes it capable of dealing with variable-high input sequences and discover long-term dependencies. Moreover, with their ability to modelling time-series data in a clear and detailed manner, RNNs are being increasingly applied to sentence modeling. For example, studies have concluded that adjusting the standard LSTM to tree-structured topologies can lead to more accurate results over a sequential LSTM on related tasks. In this section, we are going to analyse the architectures of various types of neural networks that are produced in order to generate high-level vectors that represent textual information.

2.7.1 Convolutional neural networks (CNNs)

Convolutional models where originally constructed for two-dimensional data and thus, they need to be modified for the processing of textual data. Convolutional model had shown great performance in tasks including visual data (images and videos), but they can also be used for other purposes such as text representation. CNNs greatest asset is their ability to learn from abstract input data through multiple convolutional levels and detect specific patterns on each level (e.g., textures of borders of objects in images, syllables, or word n-grams in text). CNNs that targeted text representation tasks, were initially classification models but they were later combined with other architectures. Such architectures will be analysed in later sections. In figure 10 we present the architecture of a simple convolutional neural network.

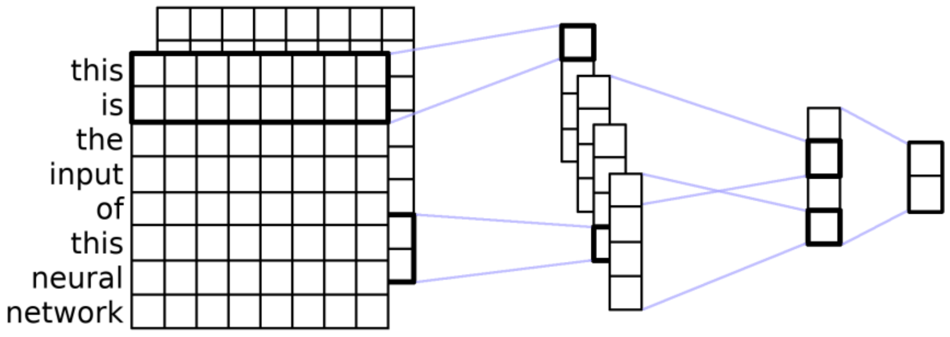


Figure 10. CNN architecture

The notable difference between convolutional neural networks made for image processing and text patterns, is the dimensionality of convolutions. Convolutions for images are two dimensional whereas convolutions for text pattern processing are mostly one-dimensional. In the above figure, we present a CNN made for the latter task. The convolution of this architecture is composed of four filters. Each filter has a kernel (a matrix of weights) that is trained to detect patterns of great importance to the task at hand. The kernel then slides over the values from the previous layer, the input values in this example, and outputs an activation that corresponds to how much that region of text fits the patterns.

Next, the convolution layer is connected to the fully connected layer. Whereas recurrent and recursive models have the ability of recognising inputs of various length, convolutional models need pooling layers to process inputs of different sizes. The pooling layer lowers the previous layer's dimensionality by forwarding (pooling) the maximum or average of each region.

2.7.2 Recursive neural networks (RecNNs)

Recursive neural networks (RecNNs), have the ability to process inputs in a recursive fashion through a tree structure. Each node in a tree represents a word a phrase, a sentence or even a larger textual input. Most of the recursive neural networks utilized for text representation are autoencoding or classification models. The representation of each node can be learned either by an autoencoding method, or through a classification task. In figure 11, we present such a tree structure, that predicts word sentiment classes. The input tokens are represented as leaf nodes, and all the other nodes are representations of combinations of the child nodes. The root node represents the entire input text.

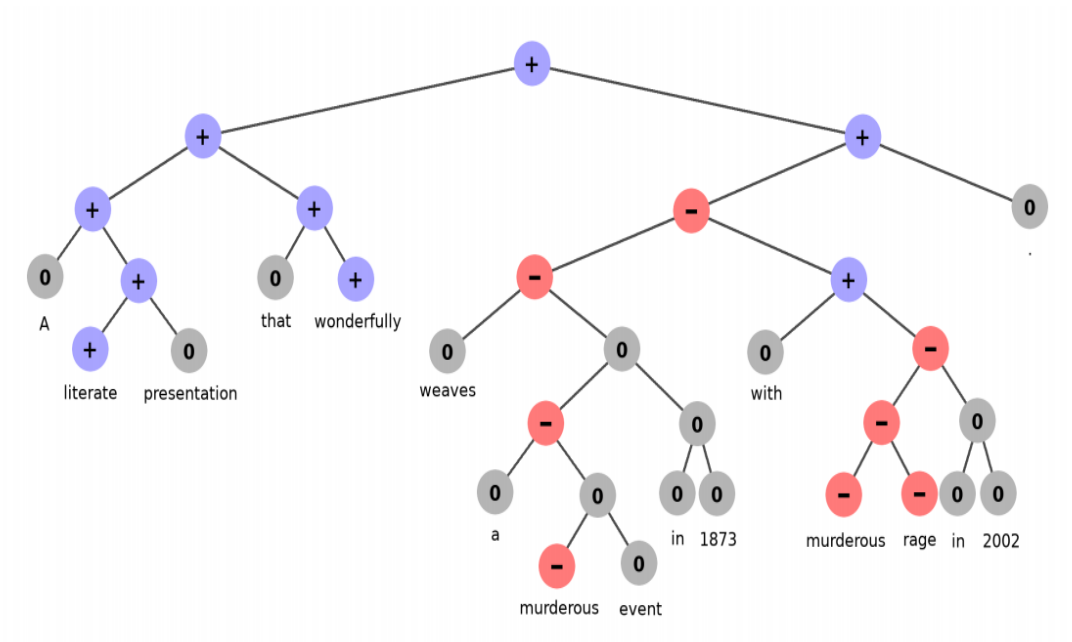


Figure *11. RecNN architecture*

Recursive neural networks are generalized forms of recurrent neural networks. Recursive neural networks read input in any hierarchical structure, in contrast to recurrent neural networks, which read information in a linear chain. Representations are merged in a recursive manner throughout the tree structure, where two or more nodes from a lower level are combined into one higher level node. With this method the learning method of recursive neural network enables combining nodes of granular linguistic meaning like words, into larger linguistic units like sentences, paragraphs, or documents. The merging process is repeated recursively until the root node is reached. Finally, the tree structure can be given along with the input, can be learned from labeled texts (texts paired with their parse trees), or they can be generated by a neural network with no supervision (latent trees).

2.7.3 Attention models

Attention neural network models, particularly transformers, have become a cornerstone in modern text representation tasks in natural language processing. The neural attention mechanism was created to capture long-range dependencies and was inspired by how humans read and understand longer texts. Long-range dependencies exceed the dependencies captured within he limited boundaries of context windows as used in shallow models.

Attention models can focus on parts of a text that is more essential for the task at hand, and thus leading to a better performance regarding long text inputs. As attention models can be implemented in a wide range of architectures, the ones that benefit the most out of the attention mechanism are mostly autoencoding or autoregressive models.

In figure 12, we can observe a self-attention head on a sentence. The visualization showed relations that are learned by the head, between words that the self-attention head has also learned. Each head determines a different kind of relation between the words.

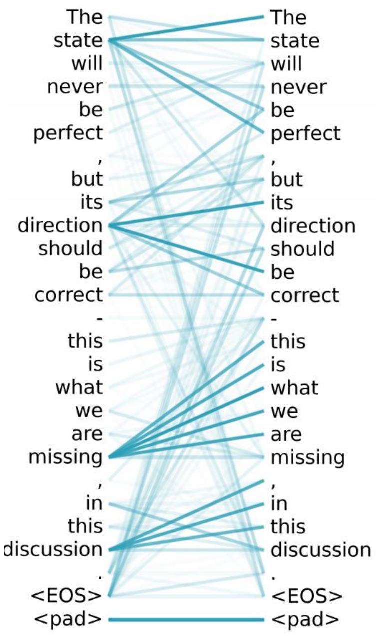


Figure *12. Attention model architecture*

The attention mechanism detects the parts of the input vector that are important for each part of the output vector. For example, when translating a sentence from one language to another, each output word has a greater dependency on specific parts of the input word. Thus, it is easy to conclude that it is important to pay great attention to those specific parts of the sentence, when deciding what the next output will be.

Self-attention mechanism works the same with a slight difference. It learns the dependencies as well, while parsing words in the input rather than the input and the output. For example, when self-attention tries to predict the next word of the sentence “I am eating a green”, it makes sense to pay more attention to words “eating” and “green” to predict the next word “apple”. The attention mechanism has multiple heads, which are similar to filters in convolutions. Each head learns to pay attention in different ways. One head can learn to attend to words that help decide the tense of the next word, whereas another head can learn to attend to entity names.

2.7.4 C-LSTM model

The C-LSTM model was developed for sentence representation and text classification. In order to obtain the sentence representation, a long short-term memory recurrent neural network (LSTM) is fed a series of higher-level phrase representations that C-LSTM extracts using CNN. C-LSTM is able to capture both local features of phrases as well as global and temporal sentence semantics [36].

In figure 13, we display the architecture of C-LSTM. Blocks of the same colour in the feature map layer and window feature sequence layer corresponds to features for the same window. The dashed lines connect the feature of a window with the source feature map. The final layer of the C-LSTM model is the last hidden unit of LSTM.

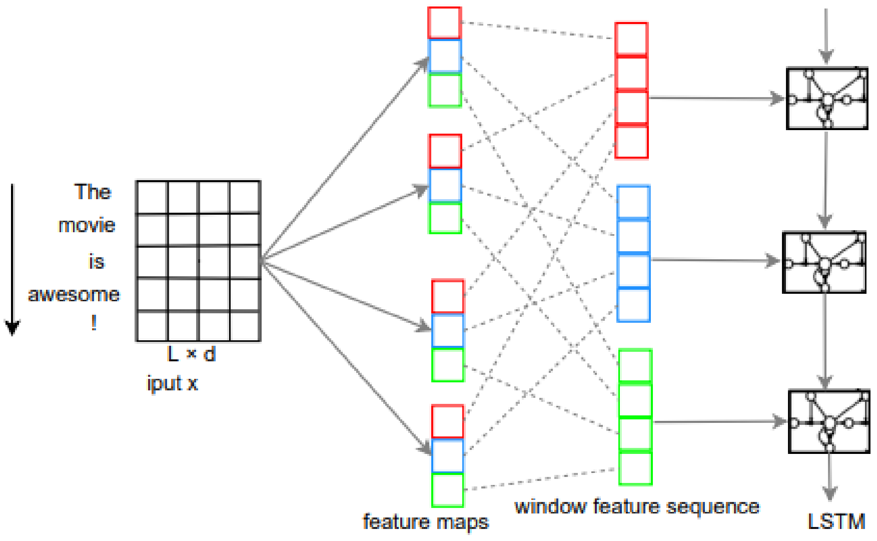


Figure 13. C-LSTM model architecture

2.7.5 Text level graph neural network for text representation and classification

This approach tries to address the problems faced when classifying text using traditional graph neural networks (GNNs), such as high memory consumption, due to the numerous edges, and the limitation in the expressive ability of the edges, due to the single-graph instantiation technique for a whole corpus. It is also challenging for graph neural networks to have a high degree of variance. This is mainly due to the fact that the structure and parameters of the graph are dependent on the corpus and cannot be modified after training.

To address the above limitations, [37] proposes a new GNN based method for text classification. Instead of building a single corpus level graph, they produced a text level graph for each input text. In the text level graph, the word nodes are connected within a small window in the text rather than connecting all the word nodes. It excludes a significant number of redundant words that are far away in the text and have little meaning with the current word and thus reducing the total number of edges. The representations of the same nodes and weights of edges are shared globally and can be updated via a method called message passing mechanism. In this method, a node receives information from neighbouring nodes to update its representation and get precise meaning in specific context.

In figure 14, we present the structure of the graph for a single text “he is proud of you”. For the convenience of the display in the figure, the p value is set to two (p denotes the number of adjacent words connected to each word in the graph) for the node “very” and p=1 for the rest of the nodes that are blue coloured. All the parameters in the graph derive from the global shared representation matrix, which is shown at the bottom of the figure.

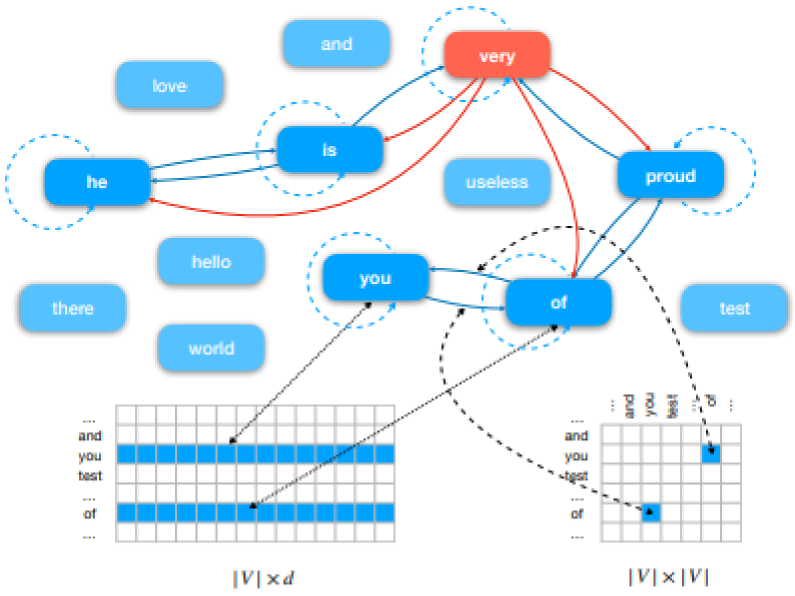


Figure 14. Text-level graph neural network

2.8 Datasets for fake claims detection

In this section, we are going to present some of the most famous benchmark datasets that are used in the automated fact-checking pipeline. Having a dataset that is well-annotated and large, can help in developing robust fact-checking models that perform well in newly emerging check-worthy claims. A well-rounded dataset is also essential in evaluating the overall performance of the pipeline. They provide a mean to evaluate the model's accuracy, and it also helps in fine-tuning the model as well. Additionally, datasets offer insights on how the claims are structured, disseminated, and verified through their semantic meaning, their emotional features, and their linguistic structure. They also ensure that the fact-checking models will be exposed to several types of claims and context from a diverse collection of sources, in order for the latter to efficiently operate in various applications.

2.8.1 The FEVER dataset

FEVER dataset stands for Fact Extraction and Verification. It consists of 185,445 claims generated by altering sequences extracted from Wikipedia and subsequently verified without knowledge of the sentence's derivation. The claims are classified as SUPPORTED, REFUTED and NOT ENOUGH INFO, by human annotators. For the first two classes, the annotators also recorded the sentence(s) forming the necessary evidence for their judgment.

The dataset was constructed in two stages. The first one, called "Claim Generation", is responsible for extracting information from Wikipedia and generating claims from it. The second stage is named "Claim labeling", and its function is to classify a claim. If a claim is supported or refuted by Wikipedia, the relevant documents are collected. If there is not enough information for the claim, then this method decides that there is not enough material to reach a conclusion regarding the claim's validity.

Given the complexity of the second task, three forms of data validation were conducted to ensure the data's integrity: 5-way inter-annotator agreement, agreement against super-annotators, and manual validation by the authors.

- **5-way Agreement**: Random selection of 4% of claims (7506 claims in total), to be annotated by 5 annotators.

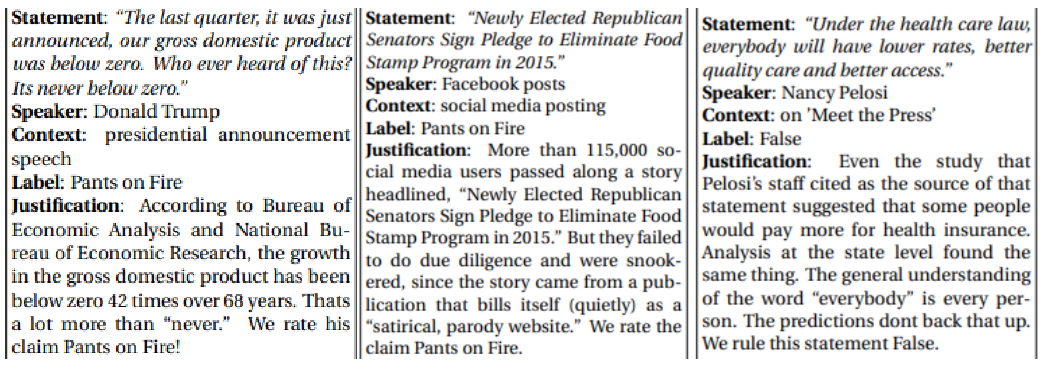
- **Agreement against super-annotators**: 1% of the data was randomly selected to be annotated by super-annotators. Super-annotators are expert annotators with no suggested time restrictions. Their task is to provide as much coverage of evidence as possible. They were also instructed to search over the Wikipedia website for every possible piece of textual information that could be used as evidence.

- **Validation by the authors**: 227 claims were chosen randomly and were examined regarding their label accuracy. It was found that 91.2% of the claims were annotated correctly.

2.8.2 The LIAR dataset

LIAR is a publicly available dataset created for the purpose of fake-news detection. It consists of approximately 12.800 claims, that were manually labeled, and present in a form of short statements in various contexts from POLITIFACT.COM[2]. This dataset can be utilized for fact-checking research and benchmarking as well. The data of LIAR, are harvested from a natural context, such as such as political debates, TV ads, Facebook posts, tweets, interview, news release, etc. In each case, the annotator provides a lengthy analysis report to support each judgment. Moreover, the links to all relevant documents are also provided. The claims of this dataset are labeled for truthfulness, subject, context/venue, speaker, state, party, and prior history.

In figure 15, we present some of the random claims that the LIAR dataset consists of.



Statement: *"The last quarter, it was just announced, our gross domestic product was below zero. Who ever heard of this? Its never below zero."*	**Statement**: *"Newly Elected Republican Senators Sign Pledge to Eliminate Food Stamp Program in 2015."*	**Statement**: *"Under the health care law, everybody will have lower rates, better quality care and better access."*
Speaker: Donald Trump	**Speaker**: Facebook posts	**Speaker**: Nancy Pelosi
Context: presidential announcement speech	**Context**: social media posting	**Context**: on 'Meet the Press'
Label: Pants on Fire	**Label**: Pants on Fire	**Label**: False
Justification: According to Bureau of Economic Analysis and National Bureau of Economic Research, the growth in the gross domestic product has been below zero 42 times over 68 years. Thats a lot more than "never." We rate his claim Pants on Fire!	**Justification**: More than 115,000 social media users passed along a story headlined, "Newly Elected Republican Senators Sign Pledge to Eliminate Food Stamp Program in 2015." But they failed to do due diligence and were snookered, since the story came from a publication that bills itself (quietly) as a "satirical, parody website." We rate the claim Pants on Fire.	**Justification**: Even the study that Pelosi's staff cited as the source of that statement suggested that some people would pay more for health insurance. Analysis at the state level found the same thing. The general understanding of the word "everybody" is every person. The predictions dont back that up. We rule this statement False.

Figure 15. Random claims from the LIAR dataset

LIAR dataset has set a 6-label mapping technique for the truthfulness ratings: pants-fire, false, barely true, half-true, mostly-true, and true. The distribution of labels in the LIAR dataset is relatively well-balanced: except for 1,050 pants-fire cases, the instances for all other labels range from 2,063 to 2,638. In order to examine the dataset's validity, as with the FEVER dataset, various verification techniques were employed.

Finally, to ensure that the dataset covers a significant variety of topics, there is also a diverse set of subjects that the claim cover. The top 10 subjects in the dataset are: economy, healthcare, taxes, federal-budget, education, jobs, state-budget, candidates-biography, elections, and immigration.

Compared to prior datasets, LIAR is the biggest in size, which enables the development of statistical and computational approaches to fake news detection. LIAR's real-world, trustworthy short claims that cover a variety of circumstances, make it feasible to conduct research on constructing a widely reaching false news detector.

[2] https://www.politifact.com/

2.8.3 The ClaimBuster dataset

ClaimBuster, is a well-known fact-checking platform, that utilizes natural language processing and supervised learning to detect important factual claims in political conversations. The ClaimBuster model is based on a human-labeled dataset of check-worthy claims from the United States general election debate transcripts.

This labeled dataset is composed of spoken sentences made by presidential candidates during past presidential debates. Each sentence of the three following labels: “not-factual claim”, “unimportant factual claim” and “important factual claim”. For the development of ground-truth label for each statement, a data collection website was constructed. Its sole purpose was to present its visitors (participants to the experiment) with a collection of unseen statements. Their task was to assign one of the three possible labels mentioned above. We present below a small analysis on each label.

- **Non-Factual Sentence (NFS):** Subjective sentences (opinions, beliefs, declarations) and many questions fall under this category. These sentences do not contain any factual claim. Below are two such examples.
    - But I think it’s time to talk about the future.
    - You remember the last time you said that?
- **Unimportant Factual Sentence (UFS):** These are factual claims but not check-worthy. The present low public interest into resolving its label. Some examples are as follows.
    - Next Tuesday is Election Day.
    - Two days ago, we ate lunch at a restaurant.
- **Check-worthy Factual Sentence (CFS):** They contain factual claims, and the general public will be interested in knowing whether the claims are true. Some examples are:
    - He voted against the first Gulf War
    - Over a million and a quarter Americans are HIV-positive

For the purpose of quality assurance, a small subgroup of claims was selected from the ClaimBuster dataset for screening purposes. The participant was presented with the screening sentence and would annotate it. Three experts would examine the participant's decision on the labeling and would then quantify the participant's quality of answers with a score metric. Participants with a low score would be eventually penalized.

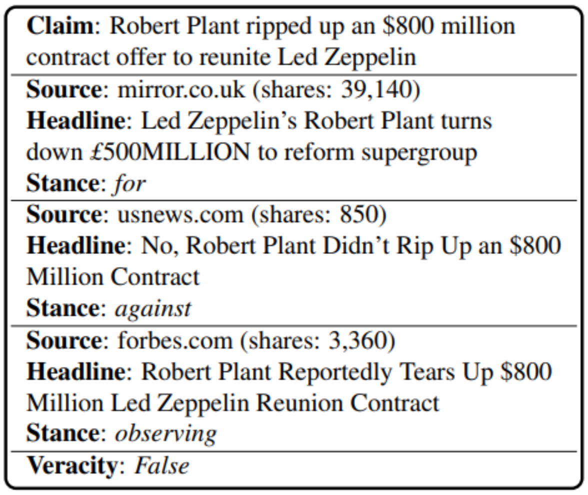
2.8.4 The Emergent dataset

Emergent dataset is derived from a digital journalism project whose task was to debunk false claims. The dataset contains 300 rumours claims and approximately 2,600 associated new articles, collected and labeled by journalists with an attached label (true, false, unverified). Each associated article is summarized into a headline and labeled to indicate whether its stance is for, against, or observing the claim, where observing indicates that the article merely repeats the claim. In general, Emergent provides a real-world data source for a variety of natural language processing tasks in the domain of fact-checking.

The subjects of the claims that were collected include a variety of topics such as world and national news in the United States, along with technology related subjects. Once a claim is identified, the journalist searches for articles that mention the claim and decides on the stance of each such article.

- For: The article states that the claim is true.
- against: The article states that the claim is false.
- observing: The claim is reported in the article, but without assessment of its veracity.

The journalist also summarizes the article into a headline. Besides the article-level stance detection, conclusions are also reached in a claim-level judgement as more articles are examined, that are relevant with the initial statement. When a journalist assumes that enough evidence for a claim is collected, the claim is either labeled true or false. If not enough evidence is collected, the claim remains unverified. In figure 16, we present a claim that has been verified on Emergent.


Claim: Robert Plant ripped up an $800 million contract offer to reunite Led Zeppelin

Source: mirror.co.uk (shares: 39,140)
Headline: Led Zeppelin's Robert Plant turns down £500MILLION to reform supergroup
Stance: *for*

Source: usnews.com (shares: 850)
Headline: No, Robert Plant Didn't Rip Up an $800 Million Contract
Stance: *against*

Source: forbes.com (shares: 3,360)
Headline: Robert Plant Reportedly Tears Up $800 Million Led Zeppelin Reunion Contract
Stance: *observing*

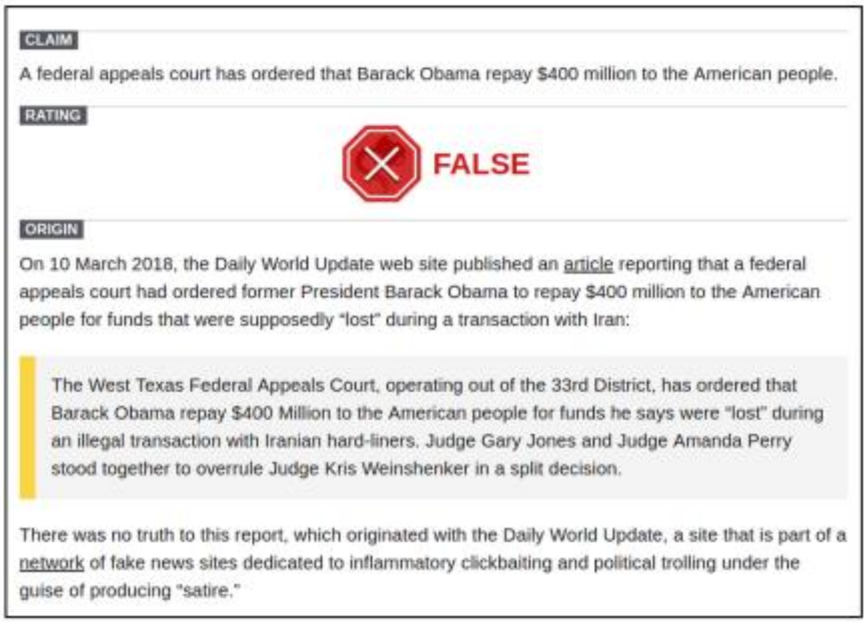
Veracity: *False*

Figure 16.Claim verification by Emergent example

2.8.5 A Richly Annotated Snopes Corpus

In order to address drawbacks of existing datasets, [43] introduced a new corpus based on the Snopes3 fact-checking website. The corpus consists of 6,422 validated claims with comprehensive annotations based on the data collected by Snopes fact-checkers. This collection covers a considerable number of domains such as discussion blogs, news, and social media, which are often found responsible for the creation and distribution of unreliable information. In addition to the claims, the corpus also consists of 14k documents annotated with evidence on two granularity levels and with the stance of the evidence with respect to the claims. The dataset allows training machine learning models for the four steps of the automated fact-checking process: document retrieval, evidence extraction, stance detection, and claim validation. An example of a fact-checking claim from Snopes is presented in figure 17. At the top of the page, the claim and the verdict are given.

[3] https://www.snopes.com/


CLAIM

A federal appeals court has ordered that Barack Obama repay $400 million to the American people.

RATING

FALSE

ORIGIN

On 10 March 2018, the Daily World Update web site published an article reporting that a federal appeals court had ordered former President Barack Obama to repay $400 million to the American people for funds that were supposedly "lost" during a transaction with Iran:

> The West Texas Federal Appeals Court, operating out of the 33rd District, has ordered that Barack Obama repay $400 Million to the American people for funds he says were "lost" during an illegal transaction with Iranian hard-liners. Judge Gary Jones and Judge Amanda Perry stood together to overrule Judge Kris Weinshenker in a split decision.

There was no truth to this report, which originated with the Daily World Update, a site that is part of a network of fake news sites dedicated to inflammatory clickbaiting and political trolling under the guise of producing "satire."

Figure 17. Snopes fact checking example

The site also provides additional information per claim, such as its origin, and evidence in the resolution of the statement are marked in a yellow bar. Additional hyperlinks that lead to documents that provide the forementioned evidence are also provided. This research collects all those information, providing a dataset enriched in contextual information.

3 Data collection and analysis

In this section, we will analyze the dataset that was used for the fact-checking pipeline and examine some of its characteristic with the use of feasible techniques and examine relevant metrics of comparison that are used for text comparison. Generally speaking, the selection of a dataset, along with the preprocessing methods that are applied on it, plays a vital role in the machine learning training process. High quality training datasets lead to more accurate model training, faster convergence, and additionally, it is the main deciding factor on model's fairness and unbiased behavior too. For those reasons, we aim to create a dataset with sufficient size, good distribution, superior quality, complexity, and relevance.

In previous approaches, datasets were generated by collecting claims from social media such as twitter, and manually annotating the claims label. Other studies used Wikipedia to collect a list of proven hoaxes[4] and fictitious people[5], or collect claims from popular fact-checking websites such as snopes.com[6], and PolitiFact[7]. Manual examination of the datasets was another common approach. Various annotators where employed, to label a claim based on the intensity of a certain emotion, or to input a True or False label to the statement itself. Typically, every dataset developed in earlier research, come across its own advantages and restrictions. There is plenty of room for experimentation, depending on the fact-checking domain (e.g., debunking political claims or verifying the accuracy of news articles) and the adopted methods such as text classification or speaker profiling.

[4] https://en.wikipedia.org/wiki/List_of_hoaxes
[5] https://en.wikipedia.org/wiki/List_of_fictitious_people
[6] https://www.snopes.com/
[7] https://www.politifact.com/factchecks/

3.1 Greek fact checking dataset

To further enhance the performance of our pipeline, the first step is to amplify the already existing dataset with new, unseen claims. This will aid our goal of achieving better classification results and thus creating a model that will be able to differentiate between genuine and fraudulent statements. The claims were collected from various Greek-speaking fact-checking websites, from Greece and Cyprus, along with some metadata such as the title of the article, its body, the date the claim was harvested and the author. More information on the features entailed on each separate statement will be analyzed in later sections.

The websites from which we extracted the claims are the Greek Hoaxes[8] website, the Greek AFP Fact check[9], and the Fact Check Cyprus website [10].Additionally, we include the existing check4facts dataset, which was originally composed of a limited amount of 33 statements. Through time, the database containing the above dataset was enriched in size, consisting of 145 claims. More information about each claim's source and its labeling, is presented on the tables below:

Data source	Total no. of claims	Claims labeled ''True''	Claims labeled ''False''	Claims labeled ''Unverified''
Greek Hoaxes	219	0	219	0
Greek AFP Fact check	307	307	0	0
Fact Check Cyprus	123	82	41	0
Check4Facts dataset	145	70	66	9

Table 1. Number of claims in the dataset in respect to their label and their source

3.2 English claims to the Greek fact checking dataset

The main idea of this approach was to further enhance the size of the dataset by adding English claims to the Greek fact checking dataset. To conclude whether this approach would prove to be beneficial to the classification task, we conducted the following experiment. Firstly, we created some dummy sentences, one in Greek and the corresponding translated one in English. We then proceed to use each of sentences as input to OPEN AI's text embedding model called "text-embedding-ada-002". For each sentence, the model generates as output a 1536-sized vector containing the embeddings of each sentence. In order to measure how close two sentences are in terms of semantical meaning, we utilized two text similarity metrics. The dot product, also known as cosine distance, and the Euclidean distance, as we saw in previous section. Both metrics take as input two vectors and produce a number associated with the the degree to which the pointing directions of two vectors line up. When two sentences are close together, the dot product converges to 1, while the Euclidean distance converges to zero. The sentences that will be used to conduct the experimentation, are listed below:

- Sentence 1: 'I like eating apples.'
- Sentence 2: 'Μου αρέσει να τρώω μήλα.'
- Sentence 3: 'My favourite fruit is an apple.'
- Sentence 4: 'Το αγαπημένο μου φρούτο είναι το μήλο.'

[8] https://www.ellinikahoaxes.gr/
[9] https://factcheckgreek.afp.com/list
[10] https://factcheckcyprus.org/

Sentence 1 and sentence 2 have the same meaning but they are presented in different languages. Likewise with sentence 3 and sentence 4. Moreover, sentence 1 and sentence 3 are stated in the same language but the semantic meaning varies slightly, as does sentence 2 and sentence 4. The result of the analysis is listed in the table below:

Pair of sentences as input	Euclidian distance	Cosine distance
Sentence 1 - Sentence 3	0.4084	0.9165
Sentence 2 - Sentence 4	0.4750	0.8871
Sentence 1 - Sentence 2	0.8188	0.6019
Sentence 3 - Sentence 4	0.7962	0.6384

Table 2. Results between the sentences using the Euclidian distance metric and the cosine distance.

Judging from the results above, we can observe that neither cosine distance nor Euclidian distance can capture the linguistic interpretation of between a pair of sentences that are identical in content yet written in different languages. Cosine distance is very low with a score of 0.6 for the sentence 1 and sentence 2 pair and 0.63 for the sentence 3 - sentence 4 pair. This led us to conclude that two sentences written in different languages do not share the same behaviour in terms of text embedding in the multidimensional space. On the other hand, two sentences written in the same language but with slight differentiation regarding the statement, can achieve better scores (0.91 for the English language and 0.83 for the Greek language).

The Euclidian distance, although more sensitive to large vectors, shares the same principal. We can easily deduct that two same sentences written in different languages are not close together. To summarise, this fundamental experiment gave us some insight about incorporating English claims to a Greek fact checking dataset. It would prove to be detrimental to the fact-checking pipeline, and thus, they will not be added to the following procedures.

3.3 English claims translated to Greek

The next approach is to pass the English claims through a Greek translation mechanism and add those statements to the dataset. To achieve this, we utilize Text Blob's translation library[11]. For this purpose, we collected various claims from English speaking fact-checking websites. A label for each of those claims was manually annotated judging from the articles content. The websites that were harvested were Snopes, Full fact[12], and Reuters fact check website[13].

Every claim that originates from a non-Greek source and thus, it is written in English, is passed through the translator. The Greek claims stay as is. Finally, we merge the two types of statements into one dataset and conduct a t-SNE analysis to examine the behaviour of the vector in the multi-dimensional space, with respect to each statements label. The results are presented in the figures below. More information on how the t-SNE analysis works, is presented in later sections.

Figure 18 shows the claims that have been marked as ''False''. It is evident that the majority of the Greek claims are somewhat clustered towards the centre of the plot (ranging on a [-20,20] scale on both dimensions), while the English claims which are translated into Greek, create various small clusters all along the area of the plot. Overall, there is a significant difference in the space occupied by each group of claims, regarding its origin (i.e. whether it has been translated or not).

In figure 19, where we examine the claims labelled as true, we can also observe slight distinctions between the Greek statements and the English ones. The latter occupies space

[11] https://textblob.readthedocs.io/en/dev/
[12] https://fullfact.org/
[13] https://www.reuters.com/fact-check/

roughly between a [-20,40] range on dimension 1 and [0,35] on dimension 2. The Greek claim’s values on the other hand, range primarily between a [-40,-20] range on dimension 1 and a [-40,20] range on dimension 2. This leads us to conclude that once again, the claims behaviour on the t-SNE plot vary depending on whether they originate from Greek speaking web sources or from foreign ones.

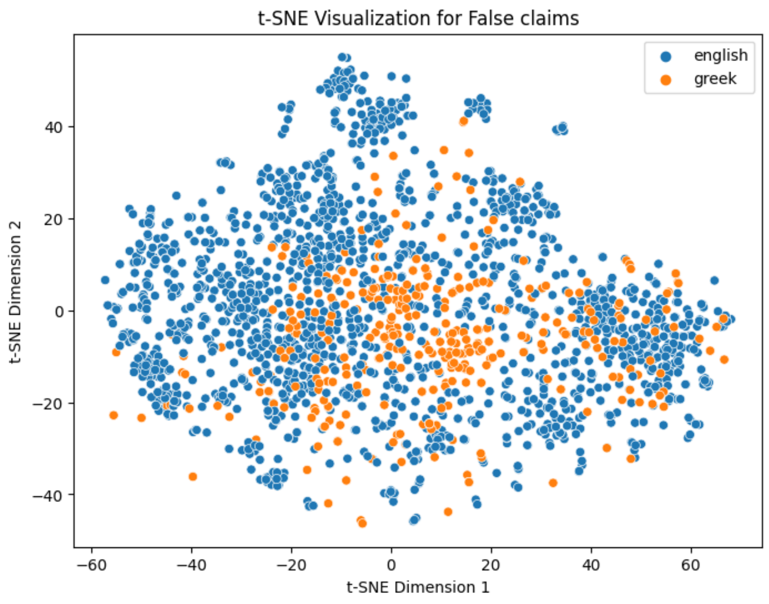


Figure 18. t-SNE visualization for False claims

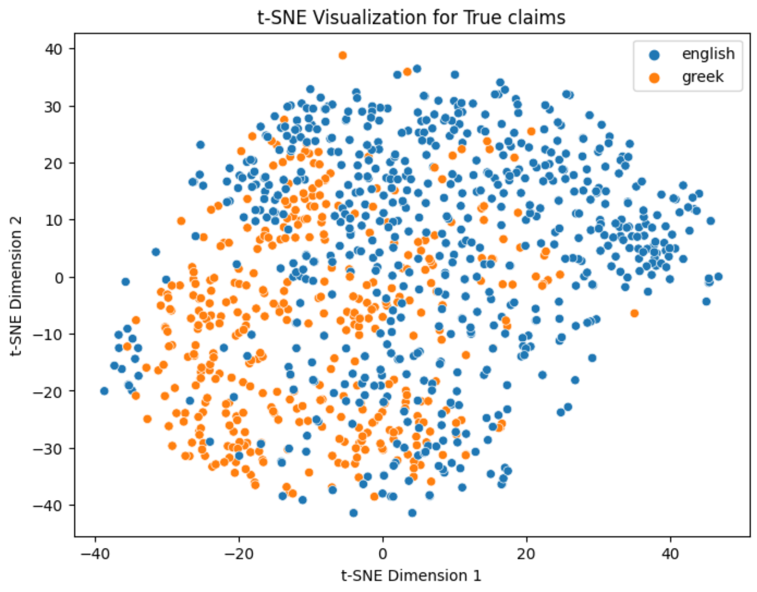


Figure 19. t-Sne visualization for True claims

4 Description of the core pipeline

In this section, we will focus on the fact-checking pipeline that takes place in this thesis. Having a dataset that consist of Greek statements, the first step after selecting a claim for analysis, is to search the web, with the intent of gathering material from web pages that aid the process of supporting or refuting a claim. After the web pages have been collected, we begin with harvesting information from each one of them. This information is in textual form, and consists of the web page's title, paragraphs relevant to the claim, URL etc. More details about the method of harvesting will be discussed in the corresponding section.

The next step is related to the feature extraction processes that are utilized in order to generate features enriched with valuable information, capable of aiding the classification stage of the pipeline. The features that are extracted are in the form of word embedding vectors produced from the information that was harvested from claim-associated web pages. This subsection further examines the type of features used for the classification, the number of them, as well as their usefulness further down the pipeline.

Finally, after selecting the necessary features, we proceed implement deep learning models able to produce a ''True'' or ''False'' label, for each claim, by using the input generated in the previous pipeline steps. We additionally employ various techniques that mitigate the error rate of each model and use accuracy as the measure of each model's performance.

In the figure below, we present the current pipeline that is employed for the purpose of this study. The first stage, Data Collection, is dedicated in expanding the already existing dataset from the check4facts pipeline, enriching it with new claims as we analyzed in the previous section. The searching stage is responsible for browsing the web, when having a claim at hand, in order to gather web sources relating to its veracity. The harvesting stage collects the textual information from the relating web sources such as the body and the title of the web source and generates new features the ''most similar paragraph'' and the ''most similar sentence'' feature. In the feature generation stage, we transform those features into text embedding vectors, using OpenAI's text embedding models. Finally, in the classification stage, a series of neural networks are utilized in order to successfully classify a collection of claims in regard to their veracity. The classification is binary, and generates a True or a False label for each claim, when the forementioned embedding vectors are used as input.

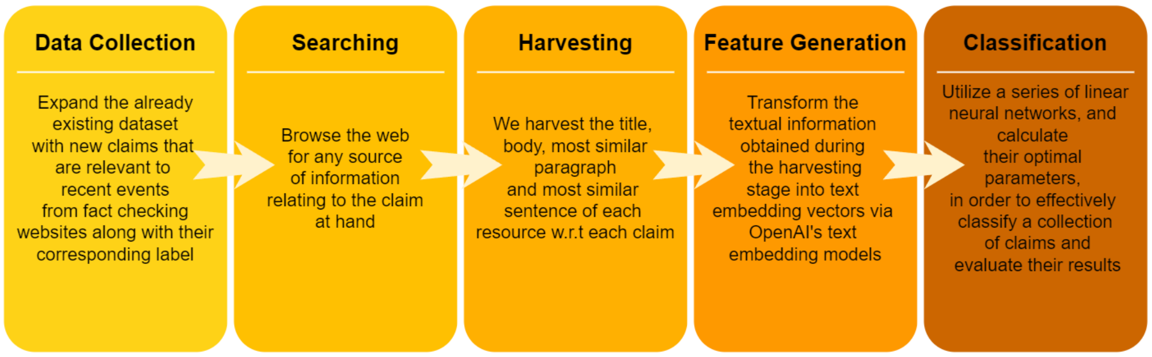

Data Collection
Expand the already existing dataset with new claims that are relevant to recent events from fact checking websites along with their corresponding label
Searching
Browse the web for any source of information relating to the claim at hand
Harvesting
We harvest the title, body, most similar paragraph and most similar sentence of each resource w.r.t each claim
Feature Generation
Transform the textual information obtained during the harvesting stage into text embedding vectors via OpenAI's text embedding models
Classification
Utilize a series of linear neural networks, and calculate their optimal parameters, in order to effectively classify a collection of claims and evaluate their results

Figure 20. The main fact checking pipeline.

4.1 Searching and Harvesting stages

The searching stage begins with using a statement from our database as a google query. We then proceed to store the list of the URLs found per claim on a list. The number of URLs found can vary per claim. Some of them had gathered too much publicity, so there was a big variety of information on the web. Others had a very low rate of publicity such that no web sources could be found. Those cases are skipped and do not participate on the future stages of the pipeline. Ideally, the goal is to collect at least five URLs per claim but for the reasons stated above, this is not always the case.

For each list of URLs discovered per claim, we initiate a harvester instance. The harvester parses the URL list, and for each item on the list, we collect its main body text along with some metadata. Those are the title of the text, the URL itself, and the date that the web source was harvested. The dynamic nature of this task made it very challenging to try to harvest extra metadata, since every web page has a unique layout, and it wouldn’t be time efficient to capture all the possible cases of its structure. Moreover, the majority of the claims were associated with less than 5 URLs which is something to be expected considering the overall minimal attention that fact-checking claims gather in the Greek domain. For this reason, we generated five data frames, with the help of the pandas[14] library. Each one of them has a different number of web sources associated with each claim. This number will from now on be represented with the letter n and it ranges from 1 to 5. As n increases by one, a new web source is added to each claim. For example, given a certain claim, advancing from n=2 to n=3 means that the claim has the same sources with n=2, but an extra one is also added.

The dot product metric is once utilized to create the “most similar paragraph” and “most similar sentence” features. For the first feature, the main body text of each web source relating to a claim, is divided into paragraphs. Each paragraph was then compared to the main claim vector, and its dot score was calculated. The paragraph that produces the highest dot score, when compared to the initial statement, was then stored into the dataset in a new column, as the “most similar paragraph” feature. The same principal was also employed in the “most similar sentence” feature. The only difference is that the main body text was split into sentences, using the nltk[15] tokenizer library. Having produced the extra features, in textual form, the next step is to transform the features into word embedding vectors. Those features are the initial claim and information gathered from a corresponding web source.

4.1.1 URL Analysis

In this section, we are going to visualize the various domains from which the information and the extra features were harvested. The results are shown with three bar plots, which demonstrate the most harvested web sources. Each bar plot corresponds to a group of claims. The first and second group consist of claims that were harvested from Greek fact checking sites, and Cypriot fact checking sites accordingly. The third group consist of claims that were collected from the check4facts database. The results are presented in figures 21 through 23.

From the bar plots, we can conclude that overall, the web sources that were harvested mainly consist of Greek news sites, Greek government websites and even domain names that belong to the European Union. The most known online open content encyclopaedia, Wikipedia, is also included in the results. A considerable number of claims utilized its knowledge for obtaining additional information. Predominantly, the harvested websites emerge from trusted sources that well known and provide reliability and trustworthiness.

[14] https://pandas.pydata.org/

[15] https://www.nltk.org/api/nltk.tokenize.html

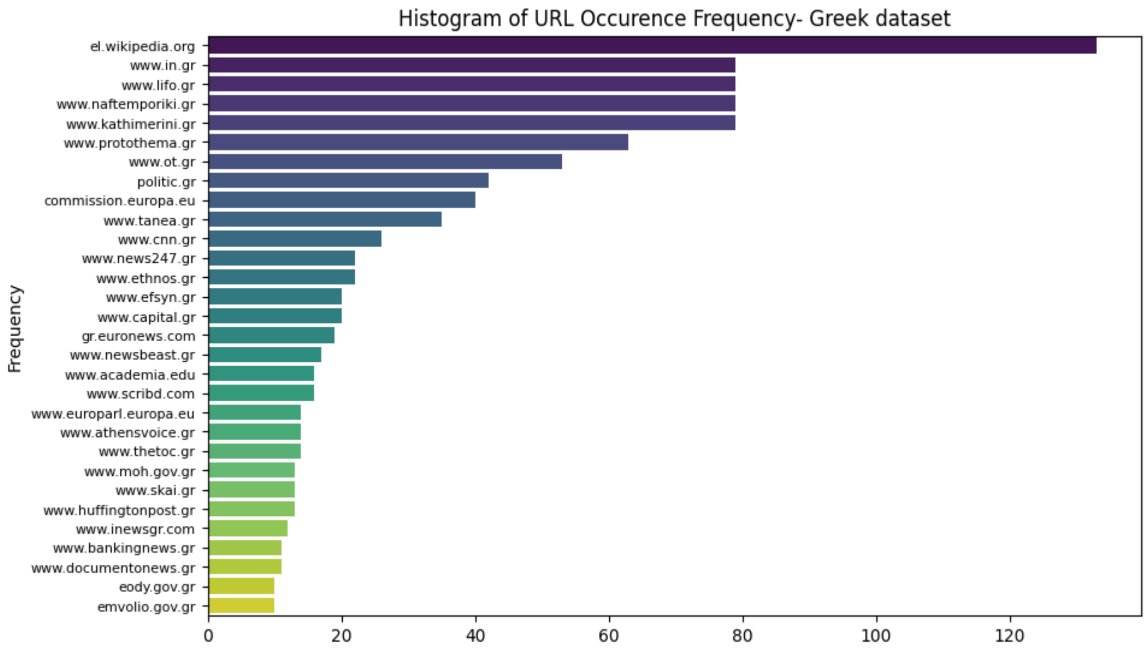


Figure 21. Histogram of URL occurrence for the Greek dataset

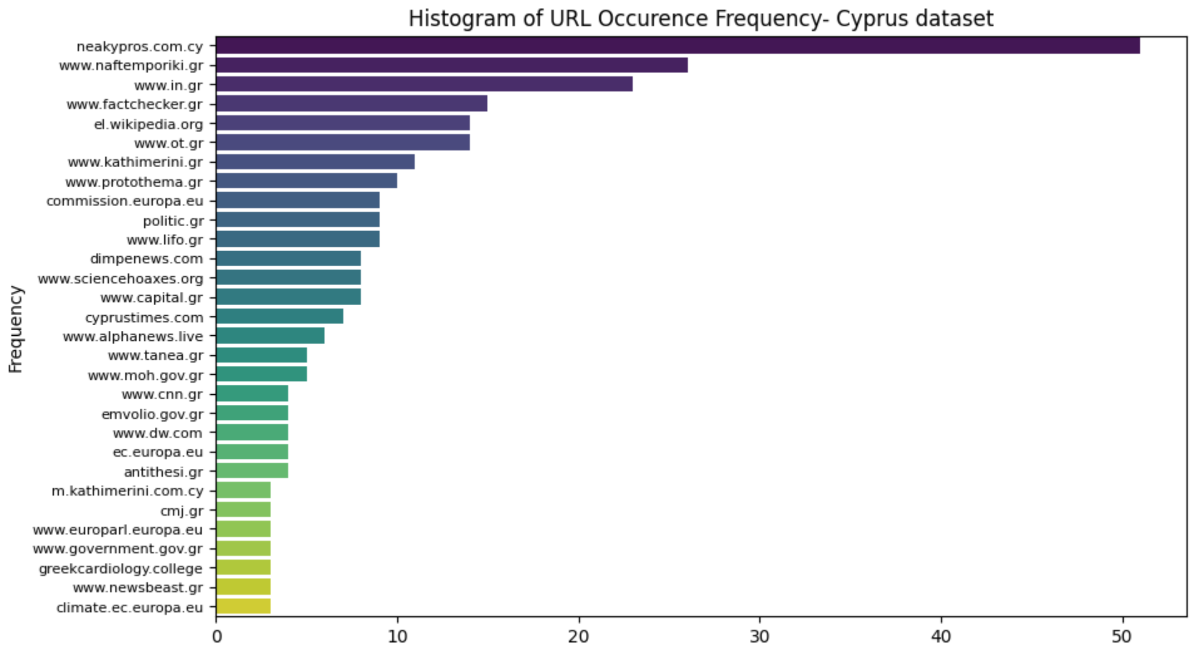


Figure 22. Histogram of URL occurrence for the Cypriot dataset

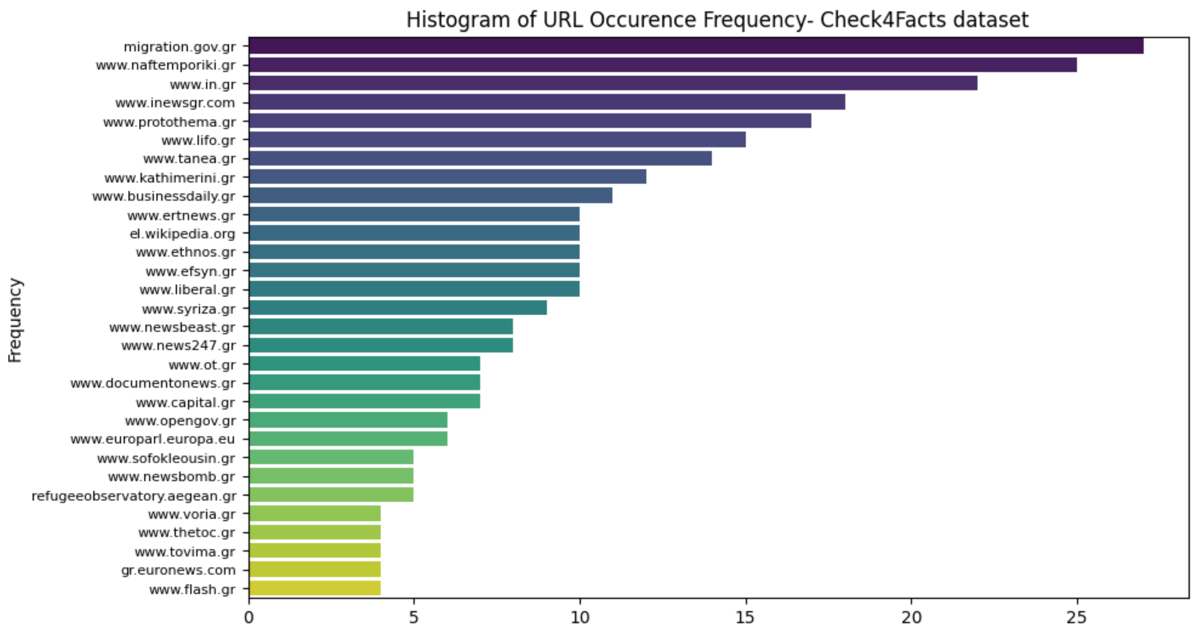


Figure 23. Histogram of URL occurrence for the Check4Facts dataset

4.2 Feature generation stage

In this section, we will present the features that were used for the classification task in the next section, along with some preprocessing techniques. Firstly, we will utilize one of OPENAI's word embedding models called "text-embedding-3-small". This model will use textual information as input and generate a 1536-sized vector as output, regardless of the input size of the text. This vector represents a machine-readable interpretation of the input text and contains text embeddings enriched with information that a deep learning algorithm can take advantage of.

The vector produced by the embedding model, is presented in a form that makes it feasible to develop comparison metrics, as shown in previous sections. The textual information that are transformed into text embedding vectors are the title, the "most similar paragraph" and "most similar sentence" features, and of course the initial claim. Additionally, from now on, the importance of a web source will be measured by the dot product produced between a claim vector and its "most similar paragraph" vector. This feature holds the most crucial information that can support or refute a claim and sorting the dataset with this manner (e.g. a descending order based on this dot product score), would create an overall harmony to the classification task, establishing a more direct method to draw conclusions from, after the end of the pipeline process. In simpler terms, when a claim is under examination, its first web source (n=1) will hold the most essential information, while the n-nth will hold the least. As more sources are added, we also introduce more noise to the pipeline.

To get a better understanding of how the datasets are now formed based on n, we have created the table below, where for each data source of information, with respect to n, we display the total amount of unique claims gathered.

Datasets	n=1	n=2	n=3	n=4	n=5
Greek Hoaxes	216	191	169	123	56
Greek AFP Fact check	296	272	230	154	75
Fact Check Cyprus	123	107	83	55	21
Check4Facts dataset	145	143	134	102	58
Total no. of claims	780	713	616	434	210

Table 3. Total number of claims regarding the no. of web sources associated with them.

It is evident that as n is increased there is a decrease in the relative web sources that can be found, in order to assist the fact checking procedure. Specifically, there is a sharp decline in the total number of claims that are associated with that n≥4 different sources. As mentioned above, most statements that are included in the datasets do not gather a significant degree of popularity which can explain the above phenomenon.

4.2.1 Finding the optimal number of relating web sources

The goal of this approach is to try to define an ideal quantity of web sources that will be concatenated with the claim and used as features in the future. Using up to five external documents as a source of information would create imbalance in the next steps of the pipeline, and thus our aim will be to lessen the use of unnecessary data. For this task, we have created several violin plots that capture the distribution of the dot score calculated between a claim and the most similar paragraph feature of its n^{th} relating web source (n ranging from 1 to 5) that was harvested. For each data source, a separate violin plot is created in order to assess the noise that a data source may introduce to the classification task. The plots are shown below:

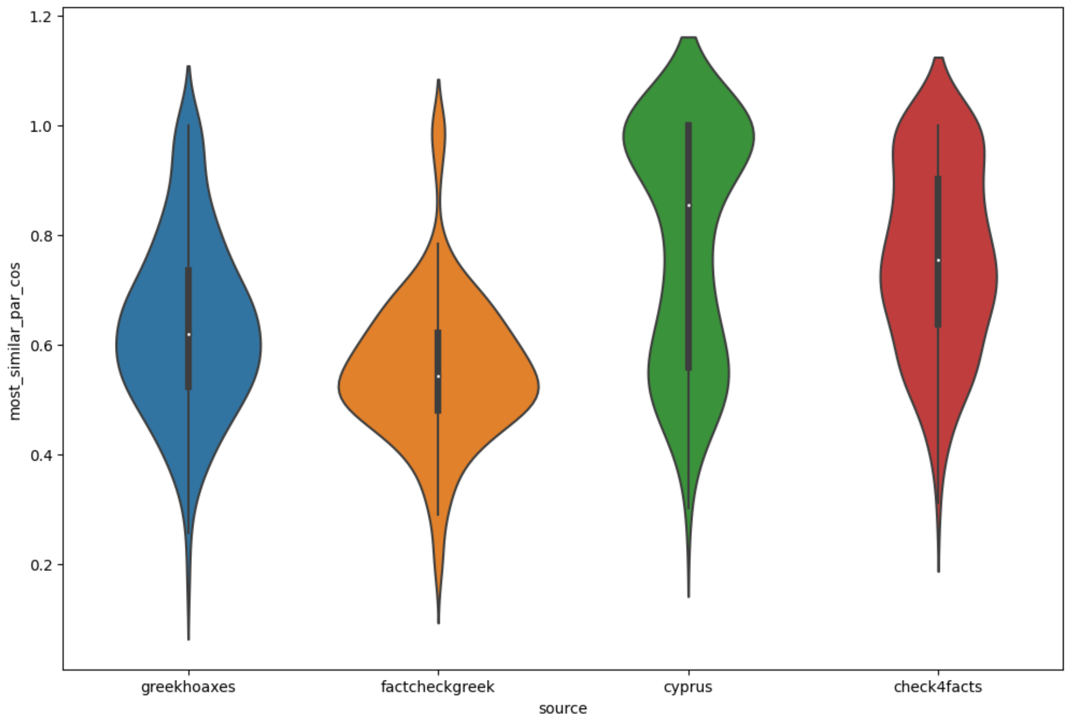


Figure 24. Claim-Paragraph cosine distance score for n=1

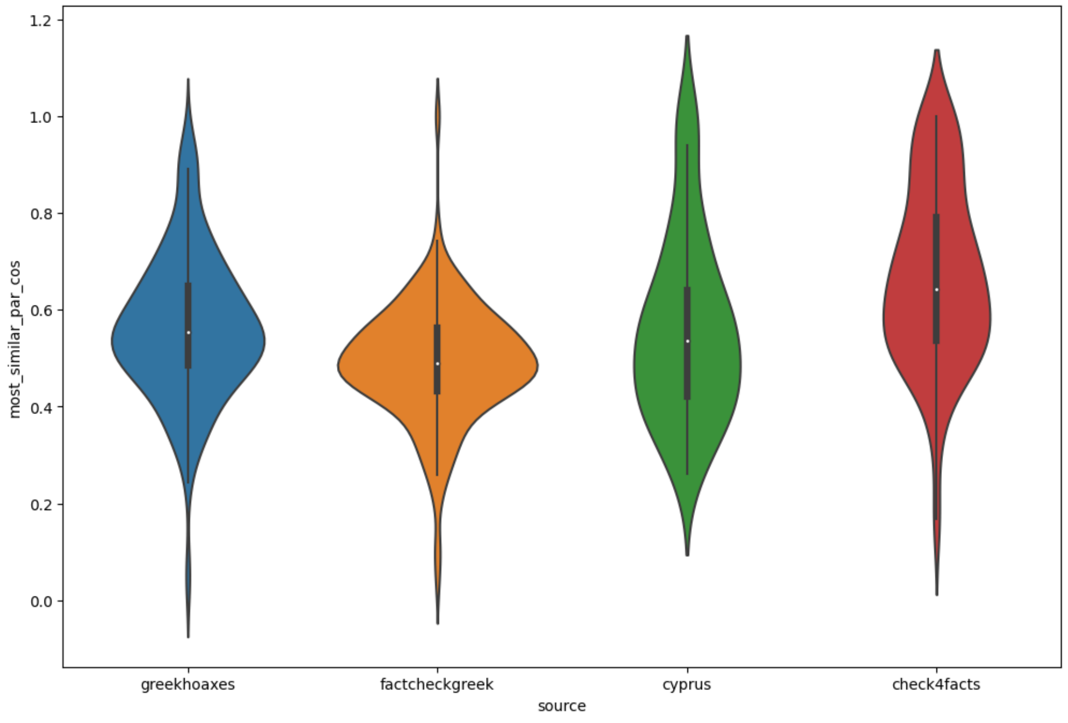


Figure 25. Claim-Paragraph cosine distance score for n=2

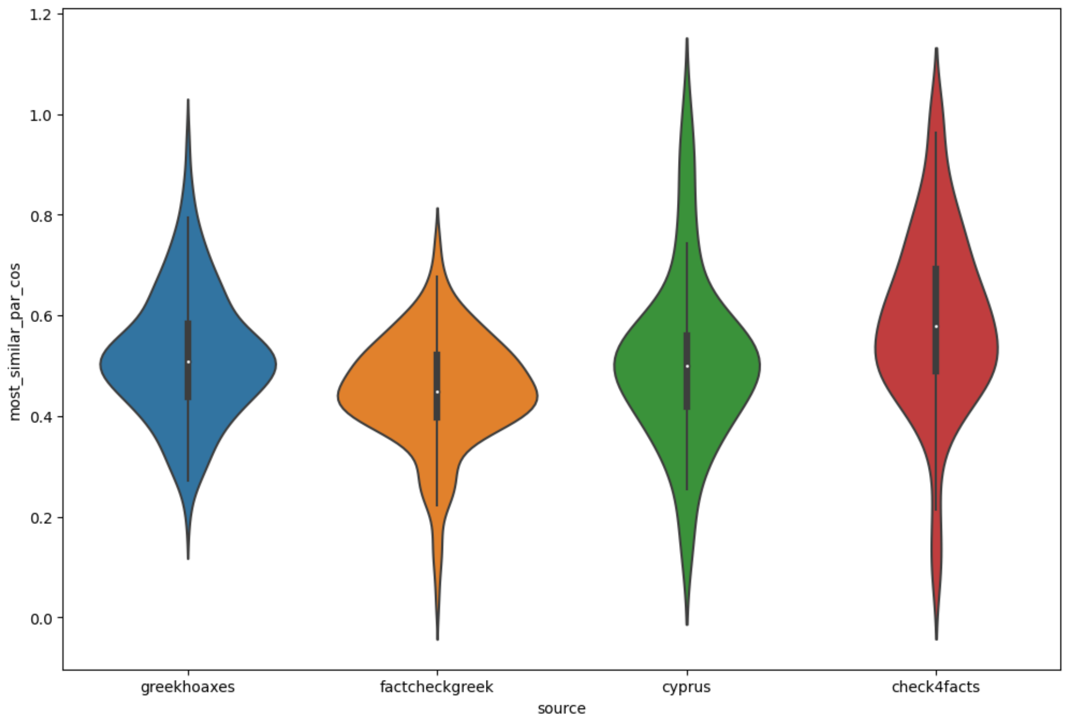


Figure 26. Claim-Paragraph cosine distance score for n=3

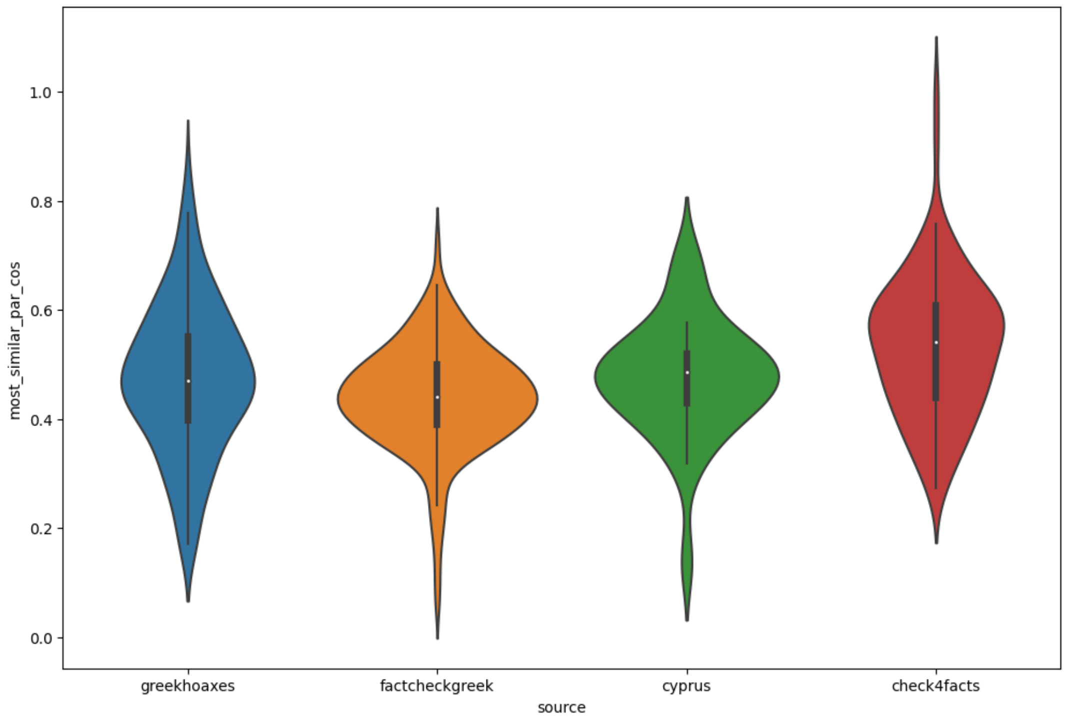


Figure 27. Claim-Paragraph cosine distance score for n=4

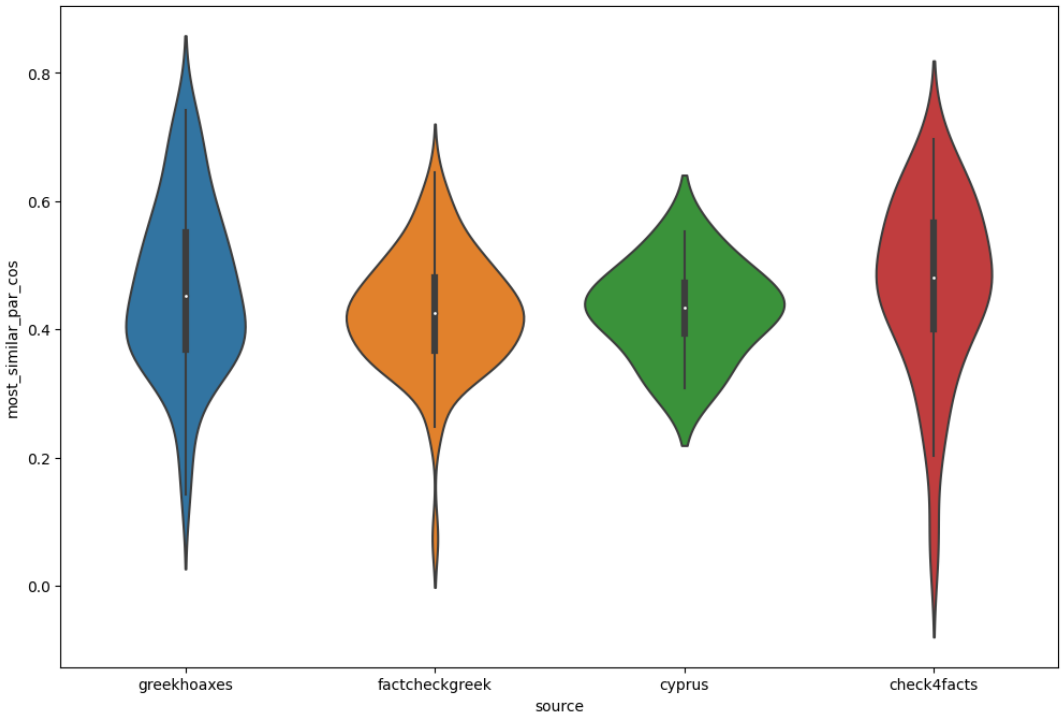


Figure 28. Claim-Paragraph cosine distance score for n=5

Judging from the result shown on the violin plots above we can conclude that the Cypriot dataset has the best score distribution for n=1 followed by the check4facts dataset. This phenomenon occurs because the Cypriot news sites regarding fact checking claims are extremely limited, so the most popular and relatable results are usually found on the first web source, which is also the one closer to the claim in terms of relativity. Additionally, for the check4facts dataset, we can observe that most claims fall into the [0.6, 0.8] score range, which is above average, and a good indicator that the most relating web source will prove a useful source of information. The rest of the data sources have a somewhat average distribution score.

For n=2 there is a significant drop in the dot score distribution. The majority of the claims, regardless of the source, present an average score that is close to the [0.5, 0.6] range. We can see a difference in dot product of even 0.3 from adding just one extra external source of information, which raises questions about the integrity of the websites content that was harvested. With that in mind, and from the knowledge derived from those plot, it is essential to decrease the amount of the external knowledge that will be put to use in the classification task.

As n increased, as show in the above figures, the distribution of the dot score between a claim and its most relating paragraph continues to decline. When n=3, the score is in the [0.4, 0.6] range, which is lower than the n=2 figure. For n=4 and n=5, the score ranges at rates that are low which could potentially create a lot of noise for our model, and create many challenges in its performance, making simple tasks such as generalization and robustness difficult to achieve. In practice, using more than three external sources per claim is redundant. This conclusion can also be supported when considering the limited number of claims that could be affiliated with four or more external documents.

In this section, we have proven that that exceeding a certain amount of additional external documents, creates problems, introduces a lot of noise, and further increases the complexity for the classification. From now on, the range of n will be reduced to a maximum of three paragraphs for each claim.

4.2.2 Oversampling

Class imbalances in classification tasks has been proven to pose significant challenges in developing effective models. In most cases, the performance of the machine learning algorithms is typically evaluated using predictive accuracy. However, this is not appropriate when the data is imbalanced, or the cost of different errors vary. For example, a typical mammography dataset like the one in [21] might contain 98% normal pixels and 2% abnormal pixels. A simple, default strategy of guessing the majority class each time would give an accuracy of 98%. However, the nature of the health domain requires a high rate of accuracy for the minority class. Simple predictive accuracy is clearly not appropriate in such situations.

In general, there are two main approaches when it comes to class imbalance issues. One is to assign different costs to training examples. The other is to resample the original dataset by oversampling the minority class or by undersampling the majority class. In this thesis, we are choosing to oversample the minority class since by undersampling, we will be left with a very small dataset at hand, and this would create different kind of impediments for our classification task.

Relating literature regarding class imbalances, states that by simply under-sampling the majority class, enables better classifiers to be built than over-sampling the minority class. Combining the two, as done in earlier research, does not produce classifiers that perform better than those created using undersampling alone. Nonetheless, the minority class has been oversampled due to replacement sampling from the original data. Our approach takes advantage of an alternative over-sampling technique.

For the purposes of this thesis, we are going to utilize the SMOTE oversampling method. Simply put, SMOTE generates synthetic examples by operating in "feature space" rather than the "data space". The minority class is oversampled by taking each minority class (one class in our case) and introducing synthetic examples along the line segments joining any or all of the k minority class nearest neighbours. Depending upon the amount of oversampling required, neighbours from the k-nearest neighbours are randomly chosen. The default version of the SMOTE uses 5 nearest neighbours. If less synthetic samples are needed, then k may vary. For example, if the amount of oversampling required is 200%, only two neighbours out of the five are randomly chosen and one sample is generated in the direction of each.

The first step of the SMOTE method it to take the difference between the feature vector (sample) under consideration and its nearest neighbour. Multiply this difference by a random number between 0 and 1 and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. On the next page, we present the algorithm of SMOTE, that displays an example of calculation of random synthetic samples.

The reason as to why synthetic minority oversampling is as a mean of improving a classifier's performance instead of oversampling with replacement is fairly simple. When oversampling with replication, the decision regions in the feature space becomes smaller and more specific, with less variance as the minority samples in the region are replicated. This is not the desired effect. On the other hand, when oversampling is achieved via the addition of synthetic examples, causes the machine learning algorithm to build larger decision regions that contain nearby minority class points. This is why SMOTE performs better than traditional sampling techniques and cost-sensitive approaches. We present the SMOTE algorithm below.

Algorithm 1: Algorithm SMOTE(T, N, k)

```
    Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest neighbours k
    Output: (N/100) * T synthetic minority class samples
    (∗ If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd. ∗)
1   if N < 100
2       then Randomize the T minority class samples
3       $T = (N/100) * T$
4       $N = 100$
5   end if
6   N = (int)(N/100) (∗ The amount of SMOTE is assumed to be in integral multiples of 100. ∗)
7   k = Number of nearest neighbours
8   numattrs = Number of attributes
9   Sample[ ][ ]: array for original minority class samples
10  newindex: keeps a count of number of synthetic samples generated, initialized to 0
11  Synthetic[ ][ ]: array for synthetic samples (∗ Compute k nearest neighbours for each minority class sample only. ∗)
12  for $i \leftarrow 1$ to T
13      Compute k nearest neighbours for i, and save the indices in the nnarray
14      Populate(N, i, nnarray)
15  end for

16  Populate(N, i, nnarray) (∗ Function to generate the synthetic samples. ∗)
17  while $N \neq 0$
18      Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbours of i.
19      for $attr \leftarrow 1$ to numattrs
20          Compute: dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
21          Compute: gap = random number between 0 and 1
22          Synthetic[newindex][attr] = Sample[i][attr] + gap ∗ dif
23      endfor
24      newindex++
25      $N = N - 1$
26  end while
27  return (∗ End of Populate. ∗)
28  End of Pseudo-Code.
```

Algorithm 1

4.3 Classification stage

In this section, we will discuss the classification process of the fact checking pipeline. The main goal is to employ deep learning techniques and various methods in order to extensively fine tune machine learning algorithms that are trained on domain specific datasets. Then we proceed to employ those models for the purpose of successfully labeling unseen claims and measure their performance on different sets of features and datasets.

Neural networks overall, have given NLP models a huge capacity for understanding and simulating human language. They have allowed machines to predict words and address topics that were not part of the learning process. To achieve this performance in NLP processes, the neural networks must be trained on a substantial number of documents (corpora) according to the type of text or language to be processed. They have shown strong capabilities that could be utilized within problems of NLP, providing efficient solutions towards problems within the field. As the neural network evolves from its most straight-forward feed-forward neural network, towards more sophisticated recursive neural network, more and more complex problems within NLP could be solved through the utilization of these networks such as sentiment analysis as well as text classification, which is the main subject of this thesis.

Additionally, we also demonstrate the model's parameters and machine learning techniques used in order to generate models capable for the task at hand. For each n, we have produced different collections of models, since with an increment of n, the final sum of the word embedding vectors size, increases dramatically. Considering this, each model collection has a different architecture. The neural network models were initialized with the use of the pytorch[16] library and consist of linear layers in conjunction with intermediate dropout layers with a p value of 0.2 or 0.5 for n=3. To measure the performance of our classification model, a cross entropy loss function was also implemented. Finally, the training dataset was split into 5 folds, and we then employed a k-fold cross validation to estimate the training accuracy of the models. Based on that, and through experimentations with different combinations of different parameter values, we fine-tuned our neural network model by estimating an optimal learning rate value, and a weight decay parameter value which is then used as a form of regularization that helps to prevent overfitting by penalizing large weights, encouraging the neural network to maintain smaller, more generalizable weights. The optimizer algorithm used was the AdamW[17] [17]. Finally, the activation function is a rectified linear unit (ReLU) or a leaky ReLU [18,19] depending on the model's architecture. More details on the optimizer algorithm the activation and loss function(s) and the k-fold technique are presented below. On the tables presented in this section, we demonstrate the architecture of each neural network model, depending on n. In the last column of every table, we present the input size of each layer in the neural network, in the form of an array.

4.3.1 Input vectors and feature reduction

In terms of input vectors, we have created three discrete experiments to evaluate the use of different combinations of word embeddings that operate as feature vectors in the classification task. On the first case, only the claim and the most similar paragraph feature will be utilized. On the second case, the most similar sentence feature will be added, and on the last case, we also include the title of the web source. The total number of 1536-sized vectors used, is calculated with respect to n (e.g. the number of web sources utilized for a single statement). This means that each discrete feature will appear n times as input, since every web source has its own unique set of title, most relating paragraph, and most relating sentence. Finally, with the use of PCA feature reduction technique, we transformed the title and most similar sentence vectors into a 768-sized vector each, from its initial 1536-sized vector. By doing so, we decreased the total input size that the neural network must process, reducing noise and error rates. Additionally, we work under the assumption that the most similar paragraph feature along with the initial claim, hold the most information that can aid the model to determine a final label. With that in mind, we want to prioritize those features, by keeping their vector size intact, simultaneously reducing the noise for the rest of the feature vectors. The final formula for the total size of the feature vectors, when all is equal to n * (768+768+1536) + 1536, where n equals the total number of web sources utilized, the sizes in the parenthesis equal the feature vector sizes to each corresponding feature, and the final 1536-sized vector represents the initial

[16] https://pytorch.org/
[17] https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

claim. When less features are employed in the classification, the number in the parenthesis is reduced accordingly.

In the figure below, we present our future reduction flowchart when given a claim and 1 relating web source, when all the harvested information is utilized. We can observe that the title and the ”most relating sentence” feature vectors were reduced in half, due the fact that they introduce a lot of noise to the classification, and the fact that the final vectors are considerably large in size, which could generate performance issues. Because the final vector size utilized as the neural network's input is large, there is a need for complex deep learning models capable of handling this amount of information.

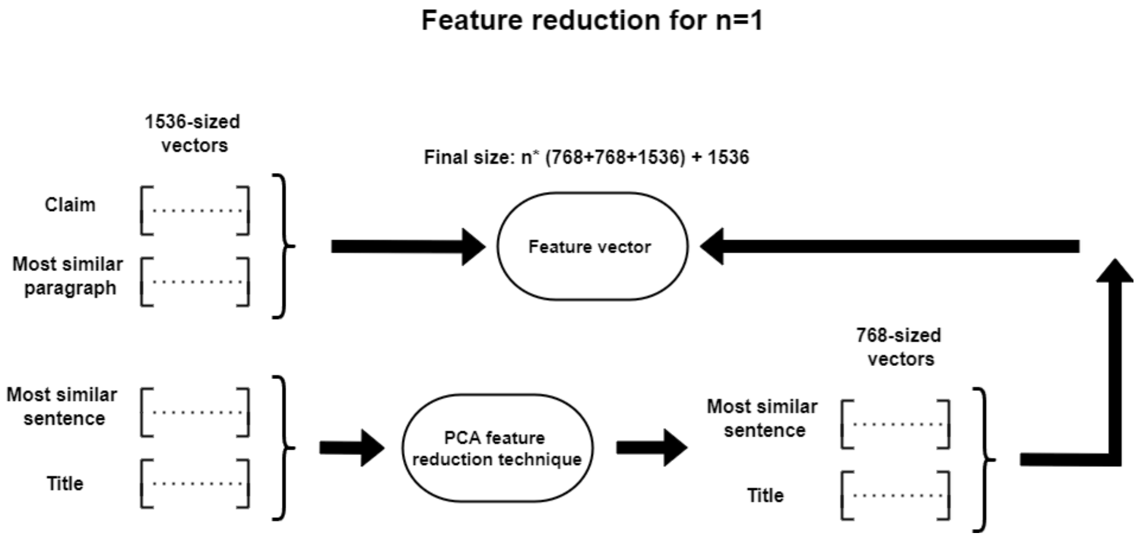


Figure 29. Feature reduction flowchart for n=1

4.3.2 Classification tasks

For the purposes of this thesis, we initialized two classification tasks. We will work in this manner, since we are interested in the performance of the machine learning algorithms in different settings, to get a better understanding of whether the dataset that the neural network is being trained on, creates an impactful shift in the model’s behaviour. Moreover, since this thesis is a continuation of the check4fact’s prior techniques, we are interested in reproducing the prior classification task conditions to a certain degree combined with new logic of approach. To this end, we define the first classification task, where the training datasets are the Greek and Cypriot claims that were harvested from fact-checking websites. The dataset used for testing consists of the check4facts dataset alone. On second classification task, we utilize a train-validation-test split technique with an 64%-16%-20% ratio. The classification is binary, meaning that the model will be trained on and predict claims on a True/False basis.

Neural Network	Layers	Activation function	Input sizes of layers
model_1	2 Linear layers	Leaky ReLU	[768, 192]
model_2			[768, 364]
model_3			[3072, 768]
model_4			[3072, 1536]
model_5	3 Linear layers		[3072, 1536, 768]
model_6	4 Linear layers		[3072, 1536, 768, 384]
model_7	5 Linear layers		[3072, 1536, 768, 384, 192]

Table 4. Table of models and their parameters when utilizing n=1 web sources per claim

models	Layers	Activation function	Input size
model_8	2 Linear layers	ReLU	[768, 192]
model_9			[768, 364]
model_10			[1536, 384]
model_11			[1536, 768]
model_12	3 Linear layers		[4608, 3072, 1536]
model_13	4 Linear layers		[4608, 3072, 1536, 768]

Table 5. Table of models and their parameters when utilizing n=2 web sources per claim

Neural Network	Layers	Activation function	Input size
model 14	2 Linear layers	ReLU	[768,384]
model 15			[6144, 1536]
model 16	3 Linear layers		[6144, 3072, 1536]
model 17	4 Linear layers		[6144, 3072, 1536, 384]
model 18	5 Linear layers		[6144, 3072, 1536, 768, 384]
model 19	6 Linear layers		[6144, 4608, 3072, 1536, 768, 384]

Table 6. Table of models and their parameters when utilizing n=3 web sources per claim

5 Result analysis

On the tables below, we present the results of the first two classification tasks, in accordance with the number of external web sources used (n), and the feature vector that were utilized as input (title vector, most similar paragraph vector etc.). Tables 7 through 24 display the accuracy of each model, the learning rate and weight decay parameters, the total number of external web sources utilized, a short description of the different combination of features that was used on each experiment, and the classification task that is at hand. In some cases, some models do not participate in the classification task, when a new feature is added, due to input size differentiations. On each table, the best accuracy achieved is highlighted. The display of the classification performance is followed by an extensive analysis of the results, with commentary and explanation in regard to the classification task, the number of external web sources, and the combination of the different feature vectors utilized for each experiment.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	43.44%	0.005	0.005	1	First classification task using only the "most similar paragraph" feature of the web source
model 2	**52.90%**				
model 3	42.75%				
model 4	48.27%				
model 5	40%				
model 6	48.27%				
model 7	45.51%				

Table 7. First classification task using the ''most similar paragraph" feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	45.45%	0.01	0.01	2	First classification task using only the “most similar paragraph” feature of each web source
model 9	**48.15%**				
model 10	39.86%				
model 11	45.45%				
model 12	44.75%				
model 13	42.65%				

Table 8. First classification task using the ‘’most similar paragraph” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	47.01%	0.01	0.005	3	First classification task using only the “most similar paragraph” feature of each web source
model 15	55.22%				
model 16	53.73%				
model 17	55.22%				
model 18	**59.70%**				
model 19	50.74%				

Table 9. First classification task using the ‘’most similar paragraph” feature, with n=3

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	55.88%	0.001	5	1	First classification task using the “most similar paragraph” feature and the “most similar sentence feature” of the web source
model 2	56%				
model 3	55.88%				
model 4	55.88%				
model 5	52.20%				
model 6	49.26%				
model 7	**56.61%**				

Table 10. First classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	48.14%	0.01	0.01	2	First classification task using the “most similar paragraph” feature and the “most similar sentence feature” of each web source
model 9	42.96%				
model 10	-				
model 11	-				
model 12	54.07%				
model 13	**57.77%**				

Table 11. First classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	51.58%	0.01	0.005	3	First classification task using the “most similar paragraph” feature and the “most similar sentence feature” of each web source
model 15	48.41%				
model 16	51.58%				
model 17	51.58%				
model 18	**52.38%**				
model 19	51.58%				

Table 12. First classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=3.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	48%	0.001	5	1	First classification task using the “most similar paragraph” feature , the “most similar sentence feature” and the title of each web source
model 2	50%				
model 3	47.05%				
model 4	47.06%				
model 5	**53.67%**				
model 6	52.94%				
model 7	51.47%				

Table 13. First classification task using the ‘’most similar paragraph” , ‘’most similar sentence” and ‘’title” feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	49%	0.01	0.01	2	First classification task using the “most similar paragraph” feature , the “most similar sentence feature” and the title of each web source
model 9	50%				
model 10	-				
model 11	-				
model 12	51.85%				
model 13	**55.55%**				

Table 14. First classification task using the ‘’most similar paragraph” , ‘’most similar sentence” and ‘’title” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	53.17%	0.01	0.005	3	First classification task using the “most similar paragraph” feature , the “most similar sentence” feature and the title of each web source
model 15	49.20%				
model 16	51.58%				
model 17	53.17%				
model 18	48.41%				
model 19	**57.14%**				

Table 15. First classification task using the ‘’most similar paragraph” , ‘’most similar sentence” and ‘’title” feature, with n=3.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	86.45%	0.005	0.005	1	Second classification task using only the “most similar paragraph” feature of the web source
model 2	**90.32%**				
model 3	85.16%				
model 4	87.09%				
model 5	84.51%				
model 6	85.16%				
model 7	87.09%				

Table 16. Second classification task using the ‘’most similar paragraph” feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	82.26%	0.005	0.5	2	Second classification task using only the “most similar paragraph” feature of each web source
model 9	80.85%				
model 10	81.56%				
model 11	82.26%				
model 12	**90.78%**				
model 13	89.36%				

Table 17. Second classification task using the ‘’most similar paragraph” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	71.31%	0.01	10	3	Second classification task using only the “most similar paragraph feature” of each web source
model 15	81.96%				
model 16	**86.88%**				
model 17	84.42%				
model 18	77.86%				
model 19	49.18%				

Table 18. Second classification task using the ‘’most similar paragraph” feature, with n=3.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	84.51%	0.005	0.005	1	Second classification task using the “most similar paragraph” feature and the “most similar sentence feature” of the web source
model 2	85.16%				
model 3	85.80%				
model 4	**90.97%**				
model 5	83.87%				
model 6	85.58%				
model 7	85.16%				

Table 19. Second classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	81.56%	0.005	0.5	2	Second classification task using the “most similar paragraph” feature and the “most similar sentence feature” of each web source
model 9	84.40%				
model 10	-				
model 11	-				
model 12	**85.11%**				
model 13	83.68%				

Table 20. Second classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	71.31%	0.01	10	3	Second classification task using the “most similar paragraph” feature and the “most similar sentence feature” of each web source
model 15	**80.32%**				
model 16	75.40%				
model 17	78.68%				
model 18	74.59%				
model 19	74.59%				

Table 21. Second classification task using the ‘’most similar paragraph” and ‘’most similar sentence” feature, with n=3.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 1	56.77%	0.005	0.005	1	Second classification task using the “most similar paragraph” feature , the “most similar sentence feature” and the title of the web source
model 2	61.93%				
model 3	82.58%				
model 4	**83.22%**				
model 5	82.58%				
model 6	**83.22%**				
model 7	82.58%				

Table 22. Second classification task using the ‘’most similar paragraph” ,‘’most similar sentence” , and ‘’title” feature, with n=1.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 8	52%	0.005	0.5	2	Second classification task using the “most similar paragraph” feature , the “most similar sentence feature” and the title of each web source
model 9	53.90%				
model 10	-				
model 11	-				
model 12	**82.97%**				
model 13	80.85%				

Table 23. Second classification task using the ‘’most similar paragraph” ,‘’most similar sentence” , and ‘’title” feature, with n=2.

Neural Network	Accuracy	learning rate	weight decay	No. of web sources	Information
model 14	54.10%	0.01	10	3	Second classification task using the “most similar paragraph” feature , the “most similar sentence feature” and the title of each web source
model 15	74.59%				
model 16	**81.97%**				
model 17	76.23%				
model 18	78.69%				
model 19	72.95%				

Table 24. Second classification task using the ‘’most similar paragraph” ,‘’most similar sentence” , and ‘’title” feature, with n=3.

On table 7, we present a classification where the check4facts dataset was utilized to test the performance of the models shown on the table (first classification task). The number of web sources utilized was one, and the only textual information utilized is the “most similar paragraph” feature. The optimal model parameters (learning rate, weight decay) were equal to 0.005. In terms of accuracy, we didn’t achieve considerable results. The best model was model 2 which achieved an accuracy of **52.90%.** Model 5 produced the weakest performance, of **40%.** The rest of the models did not surpass the **50%** threshold, and thus we can conclude that this experiment did not yield satisfactory results. In most cases, the models failed to successfully classify more than half of the training dataset correctly.

On table 8, we conduct the same experiment as above, but this time the number of web sources that are associated to a claim is two. The learning rate and weight decay parameters are set to 0.01. We observe that the best model produces an accuracy of **48.15%,** which is lower that our previous experiment. Increasing the number of web sources worsened the model’s performance.

On table 9, the number of web sources was further increased, creating a total of 3 external online documents relating to a claim. The learning rate and weight decay parameters were set to 0.01 and 0.005 accordingly. The best model’s performance was **59.70%,** which is considerably higher than the top performing models from previous experiments. This can be attributed to the complexity of the models that were initialized for this task, which allows them to detect patterns within the text embedding vectors with greater ease. Nevertheless, the top accuracy is still relatively low, but the improvement in comparison to the previous tables is worth noting. The rest of the models on this table also produce more satisfying results. All of them, expect one, surpass the **50%** threshold. This phenomenon was not observed in the previous experiments.

Table 10 presents the results of a classification experiment, with the same train/validation/test split as the previous ones. In this case however, we proceed with employing an extra feature, named “most similar sentence”, (it is translated as a 768-sized text embedding vector) in addition to the “most similar paragraph”. The number of web sources utilized is equal to one. The learning rate and weight decay parameters were set to 0.01 and 5 accordingly. The best performance was achieved from model 7, and as equal to **56.61%.** The rest of the models, (except from model 6), successfully surpassed the **55%** threshold. The addition of an extra feature proved beneficial to the classification task, when compared to table 7, where the same experiment conditions are met, with a difference on the features utilized. Learning rate and weight decay were set to 0.001 and 5.

Table 11 presents the results on an experiment with the same conditions as above (Table 10), and with the increment of external web sources (n=2). The learning rate and weight decay parameters were set to 0.01. While model 13 presents an increased accuracy in comparison to the previous experiment (**57.77%**), some models dropped below the **50%** threshold, probably due to the fact that two extra text embedding vectors were introduced, when we incremented the number n. This introduced a high degree of complexity to the classification task, that the neural networks could not follow. Models 10 and 11 did not participate in the classification because their input layer could not match the sum of the text embedding vectors.

Further increasing the number n, as shown on table 12, improved the average classification performance (most of the model surpass the **50%** threshold), but did not achieve a superior performance when compared to tables 10 and 11. Model 15 had the worst performance. The learning rate and weight decay parameters were set to 0.01 and 0.005 accordingly.

On table 13, we proceed to add textual information in addition to the previous two (most similar paragraph and sentence). We introduce a text embedding vector of size 768, holding information regarding the title of each web document. Starting from n=1, a learning rate of 0.001, and a weight decay parameter of 5, we observe that our best performing model achieves an accuracy of **53.67%**. Some models with simpler architectures on this table, do not surpass the **50%** threshold. This phenomenon occurs because of the high degree of complexity that is introduced with the addition of an extra feature as input, and with the noise that the title feature introduces.

On table 14, we further increase the number of n to two. The learning rate and weight decay parameters are equal to 0.01. The most complex model on this table yields the best results (**55.55%**). As the model becomes less complex, the accuracy drops. The lowest performance recorded was **49%**. Again, the need for complex models is highlighted in this experiment.

On table 15, with further increased the number of web sources associated with a statement when compared to the previous experiment. In spite of the fact that with the increment of n, we introduced three new text embedding vectors, (each web source introduces its own unique title, most similar paragraph and most similar sentence features) the models of on this experiment had considerable higher number of input layers and architectural parameters, that resulted in an increase in performance. The best model had the higher complexity and produced an accuracy of **57.14%**. The increase in accuracy was very slight overall. Learning rate was equal to 0.01 and weight decay was equal to 0.005.

Tables 16 through 24, provide the results of the second classification task, where the split was random in an 64/16/20 ratio (we reserved 25% of the training set for validation). Table 16 presents the experiment results where along with the claim, we utilized the ‘’most similar paragraph’’ feature alone, with n=1. Additionally, we set the learning rate and weight decay parameters to 0.005. The accuracy results were overall very satisfying. Model 2 achieved the greatest accuracy results, with a **90.32%** score. The worst performance was measured at **84.51%**, which is still higher considering the results from the first classification task.

On table 17, we worked in the same manner as before but with an increment of n, which is now equal to two. Models with higher complexity yielded superior performance. The best model on this table, obtained an accuracy of **90.78%**. The second-best accuracy was equal to **89.36%** (model 13). The rest of the models did not perform in the same manner. Their accuracy is close to **80%.** Such a drop in performance is expected considering the increment of relating web sources.

Continuing with the increase of n, on table 18, we present the results achieved when the number of relating web sources are three and the only feature utilized is the ‘’most relating paragraph’’. Learning rate is set to 0.01 and weight decay to 10. The best performance was achieved from model 16 (**86.88%**). The worst performance observed was **49.18%.** The rest of the models had varying scores ranging from **71.31%** (model 14), to **84.42%** (model 17). The average accuracy dropped significantly which was expected, given the increment of the input vectors.

On table 19, we start by setting the number n to one, the learning rate and weight decay to 0.005, and by adding the “most similar sentence” feature along with the “most similar paragraph” feature. Results were very satisfying. The best model achieved an accuracy of **90.97%**, and the rest of the models converged to **85%** accuracy in most cases, which shows that the classification was not correlated to the neural network architectures. Learning rate and weight decay were set to 0.005.

The experiment on table 20 produced inferior results when compared to the previous one. The increment of n, once again, introduced a lot of complexity to the classification that caused the neural networks to produce lower accuracy scores. The best model on this table had an accuracy of **85.11%**, and the worse one (model 1), an accuracy of **81.56%.** The rest of the models produced results that were below the **85%**, which was the threshold that most of the models from the previous experiments converged to. Learning rate and weight decay were set to 0.005 and 0.5 accordingly.

Table 21 utilized a third web source. Most of the models converge to an accuracy of 75%. The best model (model 15) produced an accuracy of **80.32%,** and the lowest score was produced by model 14 (**71.31%**). Again, we detect a drop in accuracy with the introduction of a third web source that is relating to a claim. Learning rate and weight decay were set to 0.01 and 10 accordingly.

Table 22 introduces the “title” feature and sets the number of external web sources to one. The less complex models (model 1 and 2), produce poor results with an accuracy of **56.77%** and **61.93%** accordingly. The best performance is shared by models 4 and 5, with an accuracy of **83.22%.** The rest of the models share an accuracy of **82.58%.** It is evident that the title feature was not beneficial to the classification, when compared to tables 19 and 16. Learning rate and weight decay were set to 0.005.

Further increment of n (Table 23), when 3 features are utilized (title, most similar paragraph, and sentence), lowers the accuracy of the experiment. Once again, models having a simple architecture (model 8 and 9), produce poor results with an accuracy of **52%** and **53.90%** accordingly. The best performance was produced from model 12 (**82.97%**). Model 13 produced an accuracy of **80.85%**. Learning rate was set to 0.005 and weight decay to 0.5.

Table 24 is the most complex experiment of this section. We utilize the maximum number of web sources (n=3) and the all the available features. The best accuracy is achieved by model 16 (**81.97%**). The lowest score is **54.10%** (model 14). The rest of the accuracies range from **72.95%** to **78.69%**. Learning rate is set to 0.01 and weight decay is equal to 10. The accuracy dropped in comparison to tables 22 and 23, but it is something to be expected when taking under consideration the large number text embedding vectors and the noise that is introduced to this experiment.

5.1 Comparison of the different classification tasks

One of the most important insights from the classification stage, is that the first task didn't yield the results we expected. The experiment where we trained the neural networks on the whole dataset, and then tested its performance on the check4facts claims, provided us with a maximum accuracy of **59.70%**, as we can see on table 9. We weren't able to replicate the results from the check4facts pipeline by using a different collection of claims. This may occur due to the high variance of the datasets and more specifically, from the fact that the check4facts dataset is primarily build on claims that refer to the migration subject in Greece. The rest of the dataset reflects on worldwide news, such as war news, news referring to the United Stated politics, new about the COVID19 pandemic etc. Generally, they are not constricted in Greece and Cyprus exclusively. Thus, it is reasonable to conclude that data variation plays an essential role in the classification, given the fact that both classification tasks had an approximately similar ratio of train-test data split.

In second classification task, where the train, validation test split was done in an 64/16/20 ratio randomly, had a substantial increase in performance. Models surpassed the **90%** threshold in accuracy multiple times. The best recorded model had an accuracy of **90.97%**. Randomly splitting the claims into distinct sets for training and testing, where both sets include claims from all the data sources (Greek, Cypriot, and from check4facts), allowed for better generalization and decreased the model's sensibility to variance. We can safely conclude that the 64/16/20 split was far more superior that the split in the first classification task, where only the check4facts dataset was used to evaluate the performance of our neural networks, as explained above.

The different combinations feature vectors that were utilized along the classification tasks, had an important impact to the accuracy of the models as well. When the test set consisted of claims that stemmed from the check4facts dataset, adding the most similar sentence feature along with the most similar paragraph feature, would lead to an increase in accuracy for n=1 and a slight decrease after the addition of the title feature. For n=2, adding the most similar sentence increased the accuracy and the overall accuracy. Adding the title increased the overall accuracy of the models but the best accuracy recorded didn't surpass the **57.77%** threshold. For n=3, the best results were achieved when only the most similar paragraph feature was utilized. This is expected considering the high number of feature vectors and embeddings that are used as input for n=3. Adding extra features introduces a lot of complexity and noise to the classification. Overall, the title feature vector didn't yield better results when it was utilized, and the most similar sentence vector aided in specific cases.

On the other hand, when the train/test split was random, while $n \leq 2$, gradually adding extra features such as the most similar sentence and the title feature, only lowers the average accuracy. When n=3, adding the title feature as a third feature increases the average accuracy slightly. The neural networks were the most efficient when only the most similar paragraph feature was utilized. Using the most similar sentence feature only aided in a specific model implementation for n=2, where we achieved the highest accuracy overall (**90.97%**), which was not the case in the first classification task (utilizing the most similar sentence feature helped in many cases). The title feature only worsened the results in both classification tasks. This feature contains a lot of noise and is not helpful to the neural network.

To summarize, the second classification task, where the train, validation and test split was done in an 64/16/20 ratio, we observed a substantial increase in performance. Models surpassed the **90%** threshold in accuracy multiple times. The best recorded model had an accuracy of **90.97%**. Randomly splitting the claims into train, validation, and test sets, where both sets included claims from all the data sources (Greek, Cypriot, and from check4facts), allowed for better generalization and decreased the model's sensibility to variance. We can safely conclude that the 64/16/20 split was far more superior that the split in the first classification task, where only the check4facts dataset was used to evaluate the performance of our neural networks, as explained above.

5.2 Other parameters

In conclusion, the analysis conducted showed the results below:

- **In terms of the number of web-sources:** Utilizing multiple web sources is an optimal technique when there is a lack of textual information that can be utilized. For example, when using only the "most similar paragraph feature", we can observe that making use of the maximum number of web sources (n=3), produces superior results than when additional information is at play (e.g. "title", "most similar sentence"). The surprising result is that in both classification tasks, when utilizing 3 web sources and 3 features, the accuracy is higher than the case where 3 web sources and 2 features where used. We worked under the assumption that extra features would produce noise to the task and affect the results negatively. For the rest of the cases, utilizing one or two web sources is the optimal solution for the majority of the experiments.

- **In terms of feature sets:** In the first classification tasks the combination of textual information that was used as input, did not affect the results in a significant manner. The differences in accuracy, regarding the features sets is very slight. We conclude that other parameters should be considered. In the second classification task, the "most similar paragraph" and "most similar sentence" combination produced the best results. Utilizing the "most similar paragraph" only also yielded promising results but came second in terms of accuracy. Using all three features introduced a lot of noise and worsened performance of the models.

- **In terms of model complexity:** In the first classification task, models of higher complexity in terms of layers produced better results in the majority of the cases. When increasing n from two to three, we achieved better average performance, due to the fact that models initialized on the n=3 experiments, were deeper and more complex, than the ones initialized on the n=2 experiments. In the second classification task, a model of medium to high complexity usually generated superior accuracy. This is probably due to the high number of embeddings used that generated the requirement for complex models that are more capable of locating patterns within the text embedding vectors.

- **In terms of learning rate and weight decay:** The learning rate and weight decay parameters were assigned on each experiment through various trials and experimentations, especially through the observation of the validation's test accuracy.

6 Conclusion

In this thesis, we discussed how the automated fact checking pipeline works, along with an extensive analysis of its components and its architecture. Moreover, we presented related work on how the use of the large language models can benefit the pipeline, by producing more accurate results without the need for human intervention or the constant monitoring of the pipeline. We presented relevant research that was conducted on the fact-checking domain in non-English languages and presented the results. Finally, we presented deep learning techniques and models capable of managing the automated fact-checking pipeline, and various datasets that are commonly used as benchmarks to evaluate those models. Text representation and text similarity methods as were essential to our pipeline.

The pipeline that was established in this thesis dissertation, was based on the grounds of the check4facts pipeline. We extended the previous dataset to a considerable degree, reaching a total of 794, from which just a small portion was not used due to preprocessing procedures. The expanded dataset allows for better generalization of the models, reduced variance and it encapsulates a wider range of claims for a more comprehensive coverage. The feature extraction stage was also reconstructed. With the use of the OPENAI's text embeddings, we discarded the traditional natural language approaches, which were heavily reliant on sentiment analysis, and human annotation, that require a significant amount of manual labor. Each on of the features of a textual information was translated into a 1536-sized vector that contained all the needed information in the form of embeddings. Those embeddings vectors are a core element to the architecture of many popular LLMs. Finally, the classification stage was updated with the utilization of deep learning models that provide flexibility in designing and can handle large-scale datasets and complex model architectures that are suited for challenging tasks such as the classification of textual claims that utilize multiple 1536-sized vectors as input.

The results demonstrated the promising aspects of the fact-checking pipeline when LLM architectures are utilized. Accuracy surpassed the 90% threshold in many cases, proving that a straightforward plug-and-play embedding API, can successfully replace complex NLP features (like the ones generated at the check4facts pipeline) that require multiple hours of annotation and construction, that sometimes result in uneven shaped feature vectors. Furthermore, the new established pipeline is less static, and easier to customize. One may experiment with ease on the text embedding models that will be employed, the feature vectors that are utilized, and the classification models are also very simple to personalize to meet someone's needs. The check4facts pipeline on the other hand, would face various challenges in terms of flexibility, such as the static nature of the Greek lexicon, that cannot follow the development of neologisms and lexical evolution, like a text embedding model can. Additionally, the prior classification models were not suited for the current classification tasks that are more complex and often require a considerable amount of data.

Overall, despite the drawbacks that a non-English fact checking pipeline introduces, incorporating LLM and deep learning technologies to an existing pipeline, proved to be beneficial in terms of performance, time and reusability. Future examinations should include the following:

- Extension of the current dataset, in order to encapsulate a wider range of subjects that concern the public and have gathered the equivalent attention from the media.
- Collection of claims that encompass a larger spectrum of linguistic variations, such as different syntax, etc.
- The implementation of alternative neural networks like CNN or RNN, that present different capabilities, and the further experimentation on the already established neural networks by altering its architecture (loss and activation function, number of layers etc.)
- Utilization of other multimodal data such as videos or images as features that can aid the classification process.
- Introduction of new labels, that the current pipeline could not handle, due to lack of relevant data.
- The production of justification prompts, with the use of visual aids, such as diagrams and relating images.

7 References

[1] Check4Facts*:* Public Discourse Fact Checking. Athens: Hellenic Foundation for Reasearch and Innovation, General Secretariat for Reasearch and Technology.

[2] Nayeon Lee, Belinda Z Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. 2020. Language models as fact checkers? In FEVER Workshop, pages 36–41.

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In NeurIPS, volume 33, pages 9459–9474.

[4] Nayeon Lee, Yejin Bang, Andrea Madotto, and Pascale Fung. 2021. Towards few-shot fact-checking via perplexity. In NAACL, pages 1971–1981.

[5] Zhang, X. and Gao, W., 2023. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method. arXiv preprint arXiv:2310.00305. w

[6] Sheikhi, G., Touileb, S. and Khan, S., 2023, May. Automated Claim Detection for Fact-checking: A Case Study using Norwegian Pre-trained Language Models. In Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa) (pp. 1-9).

[7] Lee, N., Li, B.Z., Wang, S., Yih, W.T., Ma, H. and Khabsa, M., 2020. Language models as fact checkers?. arXiv preprint arXiv:2006.04102.

[8] Ahmadi, N., Lee, J., Papotti, P. and Saeed, M., 2019. Explainable fact checking with probabilistic answer set programming. arXiv preprint arXiv:1906.09198.

[9] Ahmadi, N., Lee, J., Papotti, P. and Saeed, M., 2019. Explainable fact checking with probabilistic answer set programming. arXiv preprint arXiv:1906.09198

[10] N. Nakashole and T. M. Mitchell Language-aware truth assessment of fact candidates. In ACL 2014

[11] Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F. and Flammini, A., 2015. Computational fact checking from knowledge networks. PloS one, 10(6):1-13

[12] Tedmori, S. and Awajan, A., 2019. Sentiment analysis main tasks and applications: a survey. Journal of Information Processing Systems, 15(3), pp.500-519.

[13] Karadzhov, G., Nakov, P., Màrquez, L., Barrón-Cedeño, A. and Koychev, I., 2017. Fully automated fact checking using external sources. arXiv preprint arXiv:1710.00341

[14] Popat, K., Mukherjee, S., Strötgen, J. and Weikum, G., 2017, April. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 1003-1012)

[15] Popat, K., Mukherjee, S., Strötgen, J. and Weikum, G., 2016, October. Credibility assessment of textual claims on the web. In Proceedings of the 25th ACM international on conference on information and knowledge management (pp. 2173-2178).

[16] Thorne, J. and Vlachos, A., 2018. Automated fact checking: Task formulations, methods and future directions. arXiv preprint arXiv:1806.07687.

[17] Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[18] Mastromichalakis, S., 2020. ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance. arXiv preprint arXiv:2012.07564.

[19] Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

[20] Shlens, J., 2014. A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100.

[21] Woods, K., Doss, C., Bowyer, K., Solka, J., Priebe, C., & Kegelmeyer, P. (1993). Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in Mammography. International Journal of Pattern Recognition and Artificial Intelligence, 7(6), 1417–1436.

[22] Li, B. and Han, L., 2013. Distance weighted cosine similarity measure for text classification. In Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings 14 (pp. 611-618). Springer Berlin Heidelberg

[23] G. Salton and C. Buckley, "Term-weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol.24, no.5, 1988, pp.513−523.

[24] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357

[25] Gomaa, W.H. and Fahmy, A.A., 2013. A survey of text similarity approaches. international journal of Computer Applications, 68(13), pp.13-18

[26] Huang, A., 2008, April. Similarity measures for text document clustering. In Proceedings of the sixth new Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand (Vol. 4, pp. 9-56).

[27] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781

[28] Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[29] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[30] Konstantinovskiy, L., Price, O., Babakar, M. and Zubiaga, A., 2021. Toward automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. Digital threats: research and practice, 2(2), pp.1-16.

[31] Ferreira, W. and Vlachos, A., 2016, June. Emergent: a novel data-set for stance classification. In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies. ACL.

[32] Atanasova, P., 2024. Generating fact checking explanations. In Accountable and Explainable Methods for Complex Reasoning over Text (pp. 83-103). Cham: Springer Nature Switzerland.

[33] Guo, Z., Schlichtkrull, M. and Vlachos, A., 2022. A survey on automated fact-checking. Transactions of the Association for Computational Linguistics, 10, pp.178-206.

[34] Zeng, X., Abumansour, A.S. and Zubiaga, A., 2021. Automated fact-checking: A survey. Language and Linguistics Compass, 15(10), p.e12438.

[35] Augenstein, I., 2021. Towards Explainable Fact Checking. arXiv preprint arXiv:2108.10274.

[36] Zhou, C., Sun, C., Liu, Z. and Lau, F., 2015. A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630.

[37] Huang, L., Ma, D., Li, S., Zhang, X. and Wang, H., 2019. Text level graph neural network for text classification. arXiv preprint arXiv:1910.02356.

[38] Babić K, Martinčić-Ipšić S, Meštrović A. Survey of Neural Text Representation Models. *Information*. 2020; 11(11):511. https://doi.org/10.3390/info11110511

[39] Thorne, J., Vlachos, A., Christodoulopoulos, C. and Mittal, A., 2018. FEVER: a large-scale dataset for fact extraction and VERification. arXiv preprint arXiv:1803.05355.

[40] Wang, W.Y., 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.

[41] Hassan, N., Arslan, F., Li, C. and Tremayne, M., 2017, August. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1803-1812).

[42] Ferreira, W. and Vlachos, A., 2016, June. Emergent: a novel data-set for stance classification. In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies. ACL.

[43] Hanselowski, A., Stab, C., Schulz, C., Li, Z. and Gurevych, I., 2019. A richly annotated corpus for different tasks in automated fact-checking. arXiv preprint arXiv:1911.01214.

[44] Cao, H., Wei, L., Chen, M., Zhou, W. and Hu, S., 2023. Are Large Language Models Good Fact Checkers: A Preliminary Study. arXiv preprint arXiv:2311.17355

[45] Li, M., Peng, B. and Zhang, Z., 2023. Self-Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models. ArXiv abs/2305.14623 (2023).

[46] Kim, K., Lee, S., Huang, K.H., Chan, H.P., Li, M. and Ji, H., 2024. Can LLMs Produce Faithful Explanations For Fact-checking? Towards Faithful Explainable Fact-Checking via Multi-Agent Debate. arXiv preprint arXiv:2402.07401.

[47] Choi, E.C. and Ferrara, E., 2024. FACT-GPT: Fact-Checking Augmentation via Claim Matching with LLMs. arXiv preprint arXiv:2402.05904.

[48] Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. Journal of machine learning research, 9(11).

[49] Ashfaque, J.M. and Iqbal, A., 2019. Introduction to support vector machines and kernel methods. no. April, pp.1-9.