universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

„Fact Checking Approaches Based on Structured and Unstructured Textual Data"

verfasst von / submitted by

Dipl.-Ing. Nour Jnoub

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doktor der technischen Wissenschaften (Dr. techn.)

Wien, 2022 / Vienna 2022

Studienkennzahl lt. Studienblatt / Degree programme code as it appears on the student record sheet:	A 786 880
Dissertationsgebiet lt. Studienblatt / Field of study as it appears on the student record sheet:	Informatik / Computer Science
Betreut von / Supervisor:	Univ.-Prof. Dr. Wolfgang Klas

Dedicated to my beloved ones
Mama,
Papa,
Chrennoo,
Farah,
Zen,
Jaafar,
and Pip
I Love you all!

Declaration

I hereby declare that this thesis has been completed by myself by using the listed references only. Any sections, including tables, figures, etc. that refer to the thoughts, listed parts of the Internet or works of others are marked by indicating the sources.

(Location, Date) (Nour Jnoub)

Acknowledgments

Foremost, I would like to thank my supervisor, Prof. Dr. Wolfgang Klas, who instructed and guided me in each step of my Ph.D. All the time he spent with me writing each research paper and all the notes and guidance was why this work came to this stage. I have got great support from him, and all the discussions helped me set a solid direction for my work and improve and shape it until the thesis submission. Furthermore, I would like to thank Prof. Quirchmayr for encouraging me in every step of my Ph.D. work and continuously providing me with helpful notes. Moreover, I want to thank my colleague Elaheh from the deep of my heart. Ella provided me with much support, and I am happy to have her as a colleague and a real friend. I would also like to thank Belal for the fruitful discussions we had, all the coffee time with friendly conversations and scientific discussions. I would also like to thank Peter and mention his initiative idea about fact-checking. His work in the field inspired me to continue working on the same topic; working with the MIS team was a pleasure and a rich experience shaping my scientific capabilities and shaping my character on a very personal level. I am delighted and grateful. Thank you a lot to all MIS staff, especially Diana, Ramona, Jan, and Peter Kindermann, nothing better than having you in a team, always ready to help. Furthermore, I'd like to express my appreciation to Manuela, who constructed all administrative steps and motivated me with her kindness. Having Manu is like having a unique lovely spirit and joyful moments even through busy working days or stressful days like the pandemic; she kept asking, caring and giving advice. A tremendous thank goes to my family, particularly my Mom, whom I admire everything she did to see me where I am; Mirna, the title I am looking to achieve, you deserve it. A great thank you to my dad and to all his support and effort done for me, and appreciation from the deep of my heart

to the most beloved Rania, my aunt, who supported me in each step I had in my studying time, and to all the love and motivation she's always providing me. Thanks to my friends Fadi, Yara, Dede, Mouhannad, Ola, Wadi, Omar, and Mai for their love, help, and support. You are a significant part of this journey. I would also like to thank Erda, my Austrian grandmother, for always being there for me, listening to me, and considering me as a part of her family and for all the time we spent. I benefited from her life experience and how we shaped an adorable relationship, and I had a warm family place whenever I needed it. I would also like to thank my students whom I supervised and taught me that teaching is about kindness and passion, and I can learn from them as I can teach. Last but not least, From the bottom of my heart, I want to thank my beloved one, Philipp, who motivated me to finish my work and listened to me constantly, discussed my ideas, cared a lot, and encouraged my work; I am such luck to have you Philipp.

Abstract

As the amount of online textual and non-textual data on the Web continues to increase, it is subject to inconsistencies and contradictory data, which affects the quality of data provenance and the quality-related performance of search and retrieval. Therefore, appropriate methods need to be developed and integrated into the data source retrieval and search infrastructure so that inconsistent data is detected, filtered, or highlighted to the end-user, thereby improving the quality of content consumed by users.

This thesis addresses the following research questions, which are the current problems in the field of fact-checking, and addresses these problems scientifically by developing various approaches that contribute to appropriate solutions. The first question refers to the limitations of existing methodological approaches in the field of fact-checking, which influence the following questions. Since textual data on the Web is published in both a structured and an unstructured format, the second and third research questions aim to define advanced approaches for detecting and measuring conflicting textual data in different formats.

The last question illustrates the possibility of combining the developed approaches that process structured and unstructured data into one universal approach. Therefore, it can handle both formats and provide a generic solution to the problem described in this thesis.

To answer all the above elaborated research questions, this work proposes several approaches that use structured and unstructured textual data. The developed approaches enable the detection of inconsistent data by providing techniques for investigating discrepancies between data sets, detecting and highlighting suspicious behaviors behind the source that causes the dissemination of inconsistent information, and investigating and measuring the

sentiment of end-user feedback and its essential role in detecting such inconsistent/contradictory data. Finally, all the developed approaches were integrated into an algorithmic approach using logic programming as a reasoning system.

Zusammenfassung

Die Menge der textuellen und nicht-textuellen Online-Daten im Web nimmt ständig zu und enthält viele widersprüchliche Daten, was sich auf die Qualität der ursprünglichen Daten quellen, und die qualitätsbezogene Leistung des Suchens und Findens, des Abrufs auswirkt. Daher müssen geeignete Methoden entwickelt und in die Infrastruktur integriert werden, die dem Abruf und dem Durchsuchen von Datenquellen dienen, damit die Möglichkeiten eröffnen wird, und inkonsistente Daten zu bemerken, zu löschen oder für den Endbenutzer hervorgehoben zu werden, wodurch die Qualität der von den Benutzern konsumierten Inhalte verbessert wird. Die Dissertation befasst sich mit den folgenden Forschungsfragen, die die aktuellen Probleme im Bereich der Fakten-Überprüfung von Textdaten darstellen, und geht diese Probleme auch wissenschaftlich an, indem sie verschiedene Ansätze als geeignete Lösungen entwickelt. Die erste Frage bezieht sich auf die Grenzen der bestehenden methodischen Ansätze im Bereich der Fakten-Überprüfung, die die nachfolgenden Fragen beeinflussen.

Da Textdaten im Web in einem strukturierten und einem unstrukturierten Format veröffentlicht werden, zielen die zweite und die dritte Forschungsfrage darauf ab, wie man fortschrittliche Ansätze definieren kann, um widersprüchliche Textdaten in beiden Formaten erkennen und geeignet behandeln zu können. Die letzte Frage hebt die Möglichkeit hervor, die entwickelten Ansätze für strukturierte und unstrukturierte Daten in einem universellen Ansatz zusammenzufassen, der für beide Formate geeignet ist und eine generische Lösung darstellt. Weiterhin wird deutlich, wie wichtig es ist, geeignete Bewertungsmetriken zu definieren, um die Leistung der einzelnen entwickelten Ansätze zu bewerten. In dieser Arbeit werden verschiedene Ansätze für strukturierte und unstrukturierte Textdatenformate vorgeschlagen, um die

oben genannten Forschungsfragen zu beantworten. Die entwickelten Ansätze ermöglichen es, widersprüchliche Daten zu erkennen, indem sie Techniken zur Untersuchung von Abweichungen zwischen Datensätzen, zur Erkennung und Hervorhebung verdächtiger Verhaltensweisen hinter der Quelle, die die Verbreitung inkonsistenter Informationen verursacht, sowie zur Untersuchung und Messung der Stimmung des Endbenutzer-Feedbacks und seiner wesentlichen Rolle bei der Erkennung solcher inkonsistenter/widersprüchlicher Daten bereitstellen. Weiterhin wurden alle entwickelten Ansätze in einem Ansatz integriert, die mithilfe von Logikprogrammierung als schlussfolgerndes System funktioniert.

Contents

"There's a world of difference between truth and facts. Facts can obscure the truth." -Maya Angelou

CHAPTER 1

Introduction

1.1 Motivation

The widespread of misleading information, which varies between misinformation, disinformation, hoaxes, and propaganda, is increasingly becoming a big issue of our era, especially since the popularity of Social Media. Social media platforms make the spread of misinformation, false information, and fake news faster, easier, and less costly than before [40]. As a consequence, an enormous amount of such data has been observed to distribute online suspiciously, often with worrying impacts in the offline world [60] [85] [66] [78]. Thus, developing automated fact-checking systems has focused on many research communities, especially in data mining, data retrieval, and other related fields. Furthermore, all this online misinformation, disinformation, and hoaxes cause a danger to our society and economy [16], and cause a threat to our peace and democracy. Hence, to highlight and increase the awareness of its threat, many organizations in Europe detecting fake news and disinformation established. The number of active fact-checking projects worldwide has reached more than 300 projects this year, about 100 more than the Duke Reporters' Lab stated in 2019 [1]. The reasons behind such projects' rapid growth are the recent events happening worldwide, such as the United States'

[1] https://reporterslab.org/category/fact-checking/

2020 election, the consequences of the 2016 Brexit vote in the U.K., and the pandemic caused by COVID-19 [90]. An additional evident example would be the anti-vaccine people and the data they keep spreading online against the vaccination. Nevertheless, by checking some online pages [16], one might be convinced to believe the opposite, that vaccines are inefficient, useless, or even harmful [84]. These are only some opinions written by the anti-vaccination campaign, a network of people carrying various thoughts and written statements that disseminate one essence commonality: resistance to vaccines. The wide use of the Internet nowadays has eased such beliefs' transmission [14]. An additional evident example would be the anti-vaccine people and the data they keep spreading online about the COVID-19 vaccination. Many people check online topics related to COVID-19, which influences people's decision-making. Thus, it is vital to comprehend what is shared online [92]. Coronavirus pandemic has opened the doors to new fact-checking projects to decrease people's fear and avoid the danger of fake information by spotting falsehoods, hoaxes, and fake news on social media and other online websites. The fast spread of hoaxes related to an uncertain situation, such as the COVID-19 pandemic, demands cooperation between various scientific disciplines. The dynamics rising from infodemics have substantial effects on generating complex behavioral patterns. To respond correctly, it is of utmost relevance for the health domain and essential in business and Economics to understand these dynamics [68]. It depends on the statement itself. Moreover, disseminating a misleading online statement requires only seconds, and within a few hours, these statements could be widespread over the Web and already seen by many people. To prove or disprove such a statement needs a minute to hours or even more time. Fact-checking has formerly been known as a responsibility of journalists. A shortlist of well-known examples are the following: BBC Reality Check, PolitiFact.com, Factuel, FactCheck.org, FullFact.org. Manual fact-checking may be more accurate in some circumstances than automated once [40], but yet, it bears many challenges:

- The main challenge is that it is time-consuming.
- It is intellectually demanding.
- The difference in time, from when a statement is issued until it gets

checked, is essential and must be quick.

- It is not easy to judge whether a statement is correct or not until doing in-depth research, which often requires advanced search techniques.

A large portion of false information available to people is, among others, influenced by algorithms mainly intended to give personalized information. In contrast, automated fact-checking algorithms can support decreasing the volume of circulating false information. Consequently, automated fact-checking systems are in high demand to support people in identifying factual information from misleading ones, propaganda, and hoaxes [41]. Recently, different examples from different aspects confirm demand and the great necessity of reliable fact-checking techniques. The following work [68] includes a discussion regarding the effect of infodemic (false information in the time of pandemic) on people regarding online buying behaviors. The authors stated that the misinformation in the pandemic times causes a shift in buying behaviors, affecting household spending. They claimed that it impacts in the short run the consumer's expenses and, in the long run, the online market and investments, which will lead to complex interactions between them. In [40], the authors pointed out a fully automatic real-time fact-checking system as a "Holy Grail" and the goal to achieve. However, it is still out of the way from being conclusively achieved. The fact-checking systems developed so far are intended to support journalists in their work and support people to be conscious of what they are reading by clarifying the check-worthy statements, linking them with already checked ones, or highlighting the conflicting information. Current systems and approaches are discussed in further detail in the related work section.

1.2 Research Questions

In this research, robust, well-tested algorithmic approaches are developed and validated. The aim is to provide scientific contributions toward a robust concept that can detect and highlight the deviated/conflicting textual data. The proposed approaches address the following research questions:

RQ1: What are the major methodological approaches of fact-checking systems and the limitations of the state-of-the-art approaches?

This research question focuses on analyzing the state-of-the-art approaches and extracting the proposed approaches' significant advantages and current limitations. The importance of this question relies upon the need to identify the liabilities of the current approaches and look in-depth at the powerful features of the current fact-checking methods to find proper solutions that overcome the found limitations and improve the overall performance of such fact-checking models.

RQ2: How can one define advanced precision metrics to detect and measure conflicting *structured* data related to a specific topic in various heterogeneous sources?

The concept of advanced precision metrics is to provide a kind of extended similarity predicates that allow us to measure the similarity or deviation of given data. Such metrics should be configurable to the specific needs of a given context a user or application is operating. The importance of this question relies upon the need to identify conflicting data and detect the deviation in data records. Since data is not always entirely conflicting, some data can be similar but vary in the description. An example from the real world would be two descriptions of the same movie but different information regarding its release date. In other words, advanced precision metrics will steer the degree of precision employed by the comparison logic.

RQ3: How can one define advanced precision metrics to detect and measure conflicting *unstructured* data related to a specific topic in various heterogeneous sources by proper feature extraction techniques applied to unstructured textual data?

In contrast to research question RQ2, this research question focuses on the aspects of *unstructured* data. The importance of this question relies upon the complexity of analyzing unstructured textual data and the need to identify how conflicting/deviating unstructured textual data can exist and be detected. Since data publishers who intend to publish fake data may have similar behavioral patterns and often share the same sentiment toward a specific topic, those patterns can be analyzed and detected. An example from the real world would be people who write fake reviews on e-commerce platforms to demote or promote sale items and deceive other people interested in the same items. Those people are often called spammers, and their action is called spamming.

RQ4: How can one aggregate the algorithmic approaches developed and clarified in this thesis for both research questions RQ2 and RQ3 in one precision metric to take advantage of each proposed algorithm depending on the different provided data?

The answer to this question is by using a declarative solving approach. This thesis will provide a dynamic precision metric that utilizes the developed approaches mentioned in this work and operates as a reasoning system using answer set programming (ASP). A knowledge base and specific rules detect suspicious information or user abnormal behavioral patterns.

Finally, it is important to mention that measuring the performance of such approaches is highly critical due to the complexity of working with various types of data. It is still considered a challenging research field due to the expected frequent occurrence of false positives, providing a proper performance evaluation metric.

1.3 Contributions of the Thesis

This thesis provides a scientific contribution toward a robust concept that can detect, highlight, and filter the deviated/conflicting data on the Web. All significant results have been described in this thesis, and they have been evaluated and disseminated through appropriate publications. Moreover, all the research questions have been comprehensively addressed and answered.

Chapter 3 refers to the paper [55], where an algorithm has been developed to detect and measure conflicting deviated structured textual data.

Chapter 4 demonstrates two different approaches that have been developed for unstructured textual data. One approach can detect the suspicious behavioral patterns of the publishers of the fake texts, while the other approach is a generalized neural model that can classify the sentiment of the published texts. Furthermore, the first approach [54] is a rule-based approach that can detect a fake written text, integrating the behavior of authors of that text incorporated with properties derived from the content of the text. Although unstructured textual data is more complex to deal with than structured textual data, natural language processing techniques have eased fact-checking researchers' tasks. Hence, the second approach [52] presents an NLP approach that works as a domain-independent classification model for sentiment analysis using neural models. The approach has been developed and tested on different unstructured texts from various datasets.

Chapter 5 demonstrates a fact-checking reasoning system [53] with more extensive rules than the previous rule-based approach [54] presented in Chapter 4. This approach integrates three different approaches:

- one can detect conflicting/deviating data in a structured text
- spot the malicious behavioral patterns of the authors of the fake texts
- determine the sentiment presented in the written text

This approach is also referred to as the universal approach since it can integrate different approaches in one model, as presented in the research paper [53]. However, it should be mentioned that the research on detecting conflicting data and measuring the deviation of similar facts is still a challenging

topic. Therefore, it needs to be investigated further to propose novel solutions, especially for handling *unstructured* data as defined in the introduction.

In the future, it would be great to consider this work as one of the basic building blocks in the field of fact-checking, considering structured and unstructured textual data. Accordingly, it would be great if the research community of this field would be pleased about the formalism and implementation of the proposed algorithms. At the same time, the Web users would also be pleased to receive assistance when surfing the Web.

1.4 Organization of this Thesis

Chapter 1

This chapter demonstrates the motivation behind this thesis and presents the research questions that are answered later in the upcoming chapters; it also summarizes the challenges of the fact-checking topic and the contribution this thesis delivers.

Chapter 2

This chapter answers research question 2 comprehensibly by proposing an algorithm capable of detecting conflicting data and examining the deviation among structured data records published on the Web. This chapter presents a specific technique for identifying conflicting information using embedded structured data. The proposed approach is simple but effective for investigating and detecting some specific kind of conflicting or deviating data on the Web. The detection algorithm uses (a) Levenshtein distance, (b) cosine similarity, and (c) a novel concept of the control parameter, which we call sensitivity vectors. A sensitivity vector allows for selecting a specific kind of deviation of data values that one would like to investigate further in the dataset to detect conflicting data. By applying different sensitivity vectors, one can analyze the dataset concerning various deviations of data values compared to some reference data elements. The algorithm presented in this chapter has been utilized and experimented on a data collection of movies from various Webpages, illustrating how it can be applied to detect and calculate the degree of conflicting/deviating information.

Chapter 3

Chapter 3 answers the third research question comprehensibly by demonstrating two approaches:

The first approach presents a mechanism to detect fake user feedback (taking the reviews of users on movies as an example of unstructured text (movie reviews)). It incorporates the behavior of authors of the reviews combined with properties derived from the content of their published texts. This approach utilizes answer set programming (ASP) to define a rule-based system that detects suspicious patterns specified via different conditions. It provides a dynamic yet easy configurable mechanism to create robust models that consider different properties simultaneously. Such properties are, e.g., the number of similar published pieces of data, the examination of the time the data has been published, and other qualitative characteristics that relate to the published text or its owner. The output of this approach is an answer to which piece of text (e.g., review) can be viewed as genuine or not and which one requires examination.

The second approach is a generalized sentiment analysis approach, which utilizes natural language processing techniques to investigate the sentiments each published text provides. Investigating the content of the text and the sentiments helps detect whether the text owner is promoting or demoting a specific subject and whether it is genuine or spamming behavior.

Chapter 4

Answer set programming is a powerful tool for creating a rule-based model to integrate the approaches developed in chapter 2 and chapter 3 in so-called universal approaches. This chapter proposes a hybrid rule-based fact-checking framework using Answer Set Programming (ASP) and natural language processing. The proposed framework incorporates the behavioral patterns of data publishers combined with the qualitative and quantitative properties/features extracted from the content of their published text. As a case study, we evaluated the framework using a movie review dataset, consisting of user accounts with their associated reviews, including the review title, content, and the star rating of the movie, to identify reviews that are not trustworthy and labeled them accordingly in the output. This output is used later to classify the re-

views as fake or genuine and show their sentiment. The evaluation of the proposed approach showed promising results and high flexibility.

"Beware of false knowledge; it is more dangerous than ignorance." -George Bernard Shaw

CHAPTER 2

Related works

2.1 General Overview

Data on the Web is evolving tremendously and fast; some of these data are made in unstructured form, making the querying task for a specific record challenging. Although, structured data extraction still has a significant focus in database and machine learning domains due to its convenient and practical usage in diverse objectives like querying complex data records or its practical use in data integration systems.

The contributors in [5] introduce the EXALG algorithm, which was developed to resolve the problem of automatically extracting structured data from various web pages without human interferings, like having different rules or training sets. A comparable approach is introduced in the RoadRunner project [20]. However, it is not apparent to which degree the system is scalable for any web pages, especially those with complex schemas.

In [114], the paper introduces the uClaim approach, which is developed mainly to discover and detect claims of different types of facts. This paper presents an integration of different approaches described in [48], [105], [121], and [41].

Moreover, in [58], the contributors introduce the FactCheck approach that shows noteworthy progress in reasoning about the quality of data. The suggested system can identify conflicting data and enable the data consumers to

interact with the data providers and deliver their notes and recommendations through content management tools.

The approaches introduced in [11], [49], [24], and [12] all share in common the rule of majority voting algorithms in order to address the fact-checking problem. Nevertheless, the sources' trustworthiness in these works has not been considered. Therefore, such approaches do not always achieve the best or even accurate results, particularly when the data providers keep publishing misleading or conflicting data. In addition to that, they are starting from the critical consequence of source reliability. Moreover, other models tackle the fact-checking problem without using supervised learning approaches; for example, [10] introduces an algorithm that measures the distribution probabilities and uses Bayesian models that can handle complex data.

TruthFinder [86] is an approach to confirm the truthfulness of data issued on a large scale using an iterative approach. This approach defines the correlations between the web pages and their contents, collecting conflicting data about chosen objects found on different websites. The system attempts to answer how to filter the trusted information about every object. The answer is found by calculating the probabilities of facts' accuracy and the web pages' reliability. The different web pages are deduced depending on an iterative computational approach, which finds the correlation between source correctness and fact dependability.

FEVER [109] is an approach based on a large-scale dataset for fact extraction and verification that aims to verify textual selected data against textual origins. The authors propose an openly available dataset for confirming the textual data sources, consisting of 185,445 claims generated by manually selecting sentences from Wikipedia pages. After selecting the sentences comes the annotation step of the claims, where different claims have different labels. The labels are SUPPORTED, REFUTED, or NOTENOUGHINFO.

The approach presented in [107] introduces two binary classification models of Facebook posts as hoaxes or non-hoaxes, considering Facebook users who "liked" those posts. The authors used Logistic Regression as a classification model and a novel harmonic algorithm from the boolean label crowdsourcing (BLC). The dataset contains 15,500 Facebook posts and 909,236 users.

A different project from [109], is the work presented in [101], which is an

ongoing data collection project for fake news investigation and detection. The data collection consists of comprehensive selected data. They are a collection of headlines and body texts of fake news reports from the PolitiFact[1] and Gossipcop [2]. The work process mainly confirms the articles with news context (the articles from the source website of fake and real news) and social context (temporal pattern of the social engagements).

The authors of [117] introduce "LIAR," an openly available dataset for fake news detection. The dataset includes 12,800 manually labeled short records in different contexts from the PolitiFact [3]. There are six labels for the trustworthiness of the statements: pants-fire, false, barely-true, half-true, mostly-true. The approach presented in [37] uses the "LIAR" data set to test the implemented learning approaches for detecting fake data, mainly online news. It covers RNN algorithms (Vanilla, GRU) and LSTMs. The models were introduced for detecting online fake news.

The authors of [94] introduce an analytic study on the language of news media in the meaning of political fact-checking and fake news detection. The language of trustworthy news is compared with satire, lies, and propaganda to find linguistic features of unreliable text [14]. This approach experimented on a dataset of labeled records from the Politifact project on a 6-point scale.

The work in [59] presents a general-purpose technique for fact-checking that relies on retrieving supporting information from the Web [9]. The dataset includes 761 claims from Snopes [4] of various fields. Based on this dataset, a binary classification (factually true or false rumor) of claims is performed. The authors use Google and Bing search result snippets, and in some cases, they use the complete websites of the requested results.

The paper [75] offers a real-time rumor detecting algorithm for Twitter. The authors propose the first real-time rumor disproving algorithm for Twitter. This work depends on the 'wisdom of the crowds.' The authors focus on detecting a rumor as an event that may incorporate conflicting information. Their approach observes the rumored event and dynamically renders real-time updates based on any further information obtained. The authors

[1]https://www.politifact.com/
[2]https://www.gossipcop.com/
[3]https://www.politifact.com/
[4]https://www.snopes.com/

demonstrated that utilizing real-time data to accurately and efficiently detect rumors is achievable. The claims for the dataset got from Emergent [5], and Twitter's online data. The aggregated dataset for the evaluation includes 421 true and 421 false cases. In [6], the authors build a classifier that can predict whether a portion of the news is fake or not based only on its content. They address the problem by only using natural language processing. The authors use the dataset that contains 13,000 fake news reports from a public Kaggle dataset and 50,000 original news reports from the Signal Media News dataset. Various classification approaches were tested on this dataset.

ClaimBuster project introduces an automated fact-checking system [41]. This project uses supervised machine learning techniques, natural language processing, and database query methods as an extra assessment for fact-checking on political talks. It incorporates an assortment of check-worthy claims. The manual labeled dataset includes 20788 check-worthy facts from the U.S. general election debate transcripts. Furthermore, news and social media are considered; in addition to that, live news streaming is also processed. The claims are highlighted in different colors regarding their importance of being checked regarding the data visualization. It is essential to mention that this work does not incorporate automated fact-checking of claims, but the selection of check-worthy claims. The sentences in the database are divided into three classes: Non-Factual Sentence, Unimportant Factual Sentence, Check-worthy Factual Sentence. For the classification method, selected features are used. They are sentiments, length of sentences, the words of the statement, Part-of-Speech tags, and Entity Types.

Furthermore, another example for identifying misinformation in textual data, [21], the authors describe three varying steps to accomplish this: First, using supervised learning, where training data is used to classify records. The second step is applying unsupervised techniques like a Bayesian network and clustering. The third step examines so-called hybrid methods, which are combinations of multiple methods for misinformation detection in textual data, where fake reviews are spotted. According to [54], the same problem can be solved by either analyzing the content, measuring deviations between ratings, or analyzing people's behavior, which will be presented in detail in this thesis.

[5] http://www.emergent.info

[9] proposes a framework to detect fake reviews and highlight misleading content by analyzing and ordering data content features. This approach assists in identifying fake data by examining and labeling a dataset by user-centric features such as review activity, trustworthiness, and personal and social features. Furthermore, additional centric features like sentiment analysis are considered in this work. It uses behavioral features: content activity characteristics such as timestamps or the user's IP address. Moreover, different fact-checking frameworks use different kinds of such behaviors to identify fake content. Some behavioral rules introduced by [82], [83], [71] and [70] were very beneficial for the work presented in this thesis, even though all previous papers focused on the research behind the user behavior instead of the creation of a fact-checking tool. The authors of [118] presented two different techniques for detecting fake data by using supervised machine learning algorithms to classify the data from a Yelp dataset. The authors declare that by using the mentioned technique, the classification accuracy will increase compared to n-grams. The new semantic features proposed in this work are the "readability features and topic features." the readability feature used in this approach is a mathematical equation, where it calculated and evaluates the usefulness and readability of the data itself. The other features the authors used are similar to those presented in this thesis. It uses different behavioral characteristics such as date interval, percentage of positive/negative reviews, review length, which can give more information about the user, and its negative behavioral patterns. The authors used a machine-learning algorithm support vector machine (SVM), claiming that significant results were obtained. Later in their work, they reported that they achieved better results by using behavioral features.

The authors of [69] introduced a slightly different approach. They have worked with a Chinese dataset rendered by the hosting website Dianping. They started by first creating a supervised learning algorithm extended with a "Collective Positive-Unlabeled Learning Model (CPU)." The idea is that potentially misleading content has not been recognized by the algorithm they used in the unlabeled data. They have noticed a significant improvement by using the CPU model, and many potential fake textual content has been discovered. The advantage of CPU, according to the authors, is that it uses

"language-independent features," which makes it easier to be generalized for other languages. Moreover, the fact-checking community has investigated misleading information from various data sources and fields. An example of data sources would be online news pages, social media, and e-commerce platforms. Where e-commerce has also been under focus since they are playing a significant role in today's society and investigating the quality of their data is demanding since many data consumers are interested in reading the reviews on a specific product to decide whether they buy the product or not. Data providers are also interested in knowing about buyers' opinions and how that reflects on the quality of their sales. This problem was introduced first by Jindal [21] and Liu [1] concerning product reviews in 2008. Researchers tried to address the problem using different approaches, either by examining the review content or the behavioral characteristics of the people who published online content or applying sentiment analysis techniques to study the interaction between online users and items they choose and check whether this interaction is real normal interaction or actually a spamming action. Many fact-checking approaches have been developed concerning textual data, focusing mainly on public opinions, where researchers have been investing efforts to create state-of-the-art techniques for detecting fake reviews. Various detection levels should be taken into consideration as stated in [18], mainly the following ones:

- review content-based detection,
- deviations among rating-based detection, and
- review content along with user behavior-based detection.

Supervised models use the first two categories, but it may not be easy to obtain precise results, as training the models may require a big dataset to get accurate results. Thus, the authors recommended considering the third category (c) to enhance the overall performance. Lately, intensive research has been done for developing spamming detection systems using machine learning techniques.

2.2 Natural Language Processing-based Approaches

Sentiment analysis or opinion mining is an approach in machine learning which analyzes people's opinions or emotions regarding specific topics and organizations [110]. NLP-based approaches can be used when trying to detect a fake piece of text, for example, fake reviews, only if the extracted opinions are used to derive more relevant features for the task[71]. Opinion mining can classify sentiments in different categories, positives, and negatives unusual, where the unusual sentiments can be considered the ones that need further investigation. Such approaches consist of two phases [52], features extraction using natural language processing techniques, and then applying machine learning models for the classification task. For example, the authors in [26] tried to check the effects of preprocessing steps on the overall performance of reviews spam detection. They applied Support Vector Machine (SVM) and Naïve Bayes (NB), and a labeled dataset of Hotels reviews. Additionally, they listed a set of preprocessing steps in their evaluation, e.g., POS tagging, n-gram term frequencies, stemming, stop word, and punctuation marks filtering. Moreover, the following paper [74] describes the relation between truthful and untruthful reviews and subjectivity. As mentioned in the literature, fake review detection systems would work more efficiently if the sentimental analysis is used, but natural language processing (NLP) is not easy for the machines. Furthermore, the authors mentioned that similar approaches are essential because they provide more insights to solve fake review problems.

Researchers from [99] proposed the technique of detecting spam reviews by employing sentiment features of product reviews and comments. Here, three classification approaches have been used: Gradient Boosting, Random Forest, and Support Vector Machine, having as input features for training the rating of a review, review sentiment, and comment sentiment.

2.3 Rule-based Approaches

Rule-based approaches have been considered in many fact-checking projects. For example, authors in [98] reported a generalized system to process and analyze fake customer reviews inside the posts about electronic products on

social networking websites. Thus, they selected keywords and emotion-based ontologies to build their rule-based system. Furthermore, in [51] authors focused on unusual review patterns that may lead to fake reviews. Consequently, they defined a set of unexpected rules. The approach was applied using the Amazon.com[6] review dataset and detected different unexpected rules and rule groups that indicate spam activities. Moreover, [22] considers the fact that fake review's problem is basically a classification problem, and thus it can be considered as a discrete classification problem. Therefore, they proposed a fuzzy modeling-based approach using four fuzzy input linguistic variables and considered a spammer group's suspicious level to be Ultra, Mega, Immense, Highly, Moderate, Slightly, and Feebly. Subsequently, they defined a Deduction Algorithm that generates 81 fuzzy rules and a Fuzzy Ranking Evaluation Algorithm (FREA) to detect suspicious reviews, and similar approaches can be checked from Table 4.1.

2.4 Behavioral-based Approaches:

Behavioral-based approaches are approaches that consider user metadata. For example, in [9], a feature framework for fake review detection has been introduced based on a classification approach. The work addresses the problem by scraping the real Yelp dataset[7] to construct a new dataset concerning the user's electronics domain. The framework did use two types of features extraction: review-centric features related to review itself as a text, and user-centric features, which concern the behavioral patterns of the reviewers like personal, social, and review activity. In [118] two different approaches for fake review recognition are presented, where supervised learning techniques have been used for extracting and classifying new semantic features from real Yelp review datasets. The paper proposes a set of behavioral features and proves that using such features helps achieve better accuracy than using n-grams.

The approach presented in [127] showed that non-verbal behavioral features such as the number of likes/dislikes, average posting rate, review updates, etc., can be practical to detect fake reviews as much as verbal features such as the review length or review content. The paper recommended

[6]https://www.amazon.com/
[7]https://www.yelp.com/dataset

non-machine learning techniques, such as graph-based or pattern matching methods, where these different techniques examine the relationship between the reviewers, reviews, and content similarities. Furthermore, the work in [83] demonstrates that using the reviewer's behavioral features boosted the accuracy of approximately 20 percent in comparison to n-grams. However, they also mentioned that more information could be used to improve the results, like IP addresses, user logs, or duration of sessions. Other approaches, e.g., [42] and [83], proposed a fake review recognition system by capturing uncertain time intervals of reviews. The authors of both papers agree that using private information like IP addresses and MAC addresses can boost the fake detection of the system performance. Both papers agree that there are no quality standards for datasets that can be used to achieve 100 percent accuracy to tell whether a review is fake or not. Moreover, the approach presented in [72] demonstrates various features that help to spot fake reviews. Those features are mainly used to spot the review content similarities, including the reviewers, their reviews, and repeat reviews. Moreover, the items that have been reviewed and the frequency of each review are taken into consideration.

Studies in [116] showed the direct impact of online reviews on the box office revenue of a given movie, implying that people are using these online services to sometimes decide whether they are going to watch a movie or not. Based on the previous papers and as mentioned in [56], one of the significant challenges when using machine learning approaches arises from the imbalance in the state of two classes, false and factual information from the previously mentioned approaches. Obtaining labels for fraudulent information is also a challenging task. Often, these are obtained manually by experts, trained volunteers, or Amazon Mechanical Turk workers. The process requires considerable manual effort, and the evaluators cannot classify all misinformation they face. In contrast to those approaches, rule-based techniques are recognized as a white box, which provides traceability and transparency for significant decisions that require a higher degree of explanation than usually produced by machine learning approaches. Furthermore, whether to go for a rule-based or machine learning system depends on the problem, and it is mainly a trade-off between effectiveness, training costs, and understanding.

2.5 ASP-based Approaches

The concept of answer set programming (ASP) denotes a provided computational problem by a logic program. The answer sets match the resolutions and then employ an answer set solver to locate an answer set for this program. The applications of answer set programming vary between the domain, while it has become popular recently in machine learning and AI. Furthermore, the following papers describe how Answer Set Programming can be used. The authors in [28] give a good explanation and introduction on Answer Set Programming use cases. It is more interesting to look at if and how it can be used to create recognition systems and if it can be suitable to solve other problems like the one proposed in [25]. According to the authors, Answer Set Programming can solve classification problems and detect inconsistency of information. They have given an example of using it for an e-tourism portal. It can be similar to the proposed approach [54] as a recognition system because both systems are knowledge-dependent and have particular rules/algorithms to follow. However, [31] emphasizes how straightforward the implementation of ASP can be and how easy, difficult-looking problems can be solved. In this paper, the authors demonstrate a simple process of building an artificial intelligence system for games, which is usually not easy for imperative programming. They reported that logic programming implementation is excellent in dealing with knowledge incomplete and intensive applications. Besides, they hinted that Answer Set Programming could be used to detect fake news and inconsistency in data. In addition, [33] and [93] are both focusing on discovering inconsistencies by using logic programming. Although they are different papers and have different application domains, it is possible to see similarities in using the approach and motivation behind choosing logic programming as a technique. Both papers search for inconsistencies in a specific type of data by determining minimal representations of conflicts. Both papers claim that Answer Set Programming provides a straightforward method to model and represent the problem. The following paper [74] describes the relation between truthful and untruthful reviews and subjectivity.

"Misinformation is not like a plumbing problem you fix. It is a social condition, like crime, that you must constantly monitor and adjust to." -TOM ROSENSTIEL

CHAPTER 3

Fact-Checking for Structured Data

3.1 Introduction

This chapter is based on the paper *A Flexible Algorithmic Approach for Identifying Conflicting/Deviating Data on the Web* [55]

Data on the Internet often has discrepancy and conflicting information, which influence the quality of data sources and the quality-related performance of search and retrieval. Thus, suitable techniques need to be improved and integrated into the software infrastructure performing the retrieval and browsing of data sources such that contradicting data are disclosed, can be deleted, highlighted or blocked, to improve the quality of data used by data consumers. The proposed approach is capable of detecting conflicting data by providing an algorithm for inspecting the deviation between values in data records obtainable from structured data on the Web. Our approach consists of multiple steps: First, data pre-processing has been done on all the data from targeted data sources to prepare it to be properly comparable. Second, Levenshtein distance is calculated between data records to inspect the degree of conflict between data records. Third, calculating the cosine similarity between vectors of Levenshtein distance values and a user-configurable sensitivity vector, encoding the characteristics of a specified kind of conflict that is used for

detection and investigation, Finally allowing to compute ranked detection of the conflicting/deviating data records is computed. This approach has been used and tested on a data collection of movies from different Web-pages, explaining how the techniques can be applied to detect and measure the degree of conflicting/deviating information on the Web.

3.2 Approach

Given a selected data record from a collection of data records as a reference item our approach allows for investigating different kinds of deviations between the values of the reference item and the values of the data elements in the dataset.

The proposed approach consists of three phases.

First, preprocessing the data including cleaning, unifying, etc. to make data elements comparable may be needed. For example, these preprocessing steps may include the following:

- For attributes of type date all dates should be converted to a unified date format, e.g., day/month/year or year-month-day.
- For attributes of type string only characters from a well-defined alphabet (e.g., converted to unified lower or upper case) should be kept using, e.g., regular expressions.
- For numeric attributes, only numeric values may be kept and well-structured using, e.g., regular expressions.

The second phase is about computing the pairwise deviations between the values in the data records of the dataset and the values in the reference data record by means of normalized Levenshtein Distance. The resulting matrix of Levenshtein distance values will hence encode the extensiveness of, e.g., misspellings in string values or the deviation of actors names. Obviously, as Levenshtein distance counts the editing steps needed to transform a value to another value, the notion of deviation is very much a syntactic one.

The third phase is based on the resulting matrix of Levenshtein distance values. It is about the computing of a ranked deviation list which can be

controlled by a so-called sensitivity vector. The sensitivity vector allows specifying a specific kind of deviation that one would like to investigate in the dataset. For example, one might be interested to find the most deviating records compared to the reference item, or to find records showing some very specific deviation in some parts of the record, or one might be interested to find the non-deviating records, etc..

In the following, we will present the algorithm for phase two and three.

3.2.1 Computing deviation between data records

Algorithm 1 shows the pseudocode for computing the pairwise deviation between the reference item and the individual records in the dataset. The algorithm needs the following input parameters:

- *Dataset* denotes the globally available dataset encoded as a table.
- $RefItem$ denotes the data record consisting of attribute-value pairs, that serves as the reference item all the other records in the dataset are to be compared with.

The algorithm checks the similarity between the $RefItem$ and each record in the *Dataset* using Levenshtein distance (see Algorithm 1, line 4). Levenshtein distance is a string metrics for measuring the difference between two character sequences. It indicates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string to the other [113].

The Levenshtein distance is then normalized by dividing each Levenshtein distance value by the length of the longest string of the compared attributes within the pairwise similarity calculation (see Algorithm 1, line 5 and 6). The normalization equation used is shown in Equation (3.1),

$$NormalizedLevenDist(str1, str2) = \frac{LevenDist(str1, str2)}{max(length(str1), length(str2))} \quad (3.1)$$

where $LevenDist$ is the Levenshtein distance function, $str1$ and $str2$ are the strings to be compared, max computes the maximum of the $length$ of the

two given strings. The result of computed similarities are then returned as normalized Levenshtein distance matrix.

Algorithm 1 CompDev

Input:
global $Dataset$: table with M data tuples, each consisting of D elements
$RefItem$: vector consisting of D attribute-value pairs

Output:
$norm\_LD\_matrix$: Levenshtein Distance Matrix (of dimension M x D)

```
1:  function CompDev(RefItem)
2:      for i = 1 to M do
3:          for j = 1 to D do
4:              ld ← LevDistance(RefItem[j].value, Dataset[i, j])
5:              len ← length(RefItem[j].value)
6:              maxLen ← max(len, length(Dataset[i, j]))
7:              norm_LD_matrix[i, j] ← ld/maxLen
8:          end for
9:      end for
10:     return norm_LD_matrix
11: end function
```

3.2.2 Computing ranked deviation list

Algorithm 2 shows the pseudo code for computing the cosine similarity between sensitivity vector $SenVec$ and the normalized Levenshtein distance matrix $LDmatrix$, resulting from the previous computing step. The sensitivity vector $SenVec$ denotes a control parameter encoding a specific degree of deviation of data values that one is interested to investigate.

A sensitivity vector is of the same dimension D as the $RefItem$, the reference record used in Algorithm 1. The set of possible sensitivity vectors is $Value^D$ with $Value \in [0, 1]$, because the Levenshtein distance values from Algorithm 1 we use for comparison are between 0 and 1.

Algorithm 2, line 4 computes the Cosine Similarity between $SenVec$ and each row in $LDmatrix$, iterating over all rows, constructing an intermediate matrix $DeviationMatrix$ consisting of the index of a data record and its cosine similarity value. The Cosine Similarity function used is given by Equation (3.2):

$$CosineSimilarity(SenVec, LevenDist[i]) = \frac{SenVec \times LevenDist[i]}{|SenVec| \times |LevenDist[i]|} \quad (3.2)$$

where $SenVec$ is a non-zero length sensitivity vector consisting of D values $v \in [0, 1]$, and $LevenDist[i]$ is a non-zero length vector consisting of normalized Levenshtein distance values $v \in [0, 1]$ for a data element i.

In the special case that the record in the dataset does not show any deviation with respect to the reference item, i.e., $|LDmatrix[i]| = 0$, the corresponding similarity value in the $DeviationMatrix$ is set to 0 (see Algorithm 2, line 5 and 6).

In the special case that the sensitivity vector has been chosen to express that we are interested in the exact or most similar records (no or almost no deviation with respect to the reference item), i.e., a vector with only 0 values, $|SenVec| = 0$, the corresponding similarity value in the $DeviationMatrix$ is set to a value depending on the average deviation encoded for the D attributes of the data record (see Algorithm 2, line 7 and 8).

Finally, the constructed $DeviationMatrix$ is sorted by the similarity values computed before in descending order, which creates the final ranked list of records to be returned.

3.2.3 Application scenarios of investigating data deviation

Let $Dataset$ be a collection of movie descriptions, consisting of movie title, release date, the name of the movie director. Let us assume $Dataset$ has been preprocessed according to phase 1 described above.

Let $RefItem$ be a description of a single movie subject to our interest of investigating the data collection, e.g., it might be a movie description known to be perfectly correct. E.g., $RefItem = (\text{"}Title\text{"} = \text{"}Flubber\text{"}, \text{"}ReleaseDate\text{"} = \text{"}1997 - 11 - 15\text{"}, \text{"}Director\text{"} = \text{"}LesMayfield\text{"})$.

Algorithm 2 CompRankDevVecs

Input:

$LDmatrix$: Levenshtein Distance Matrix (of dimension M x D)
$SenVec$: sensitivity deviation vector (of dimension D)

Output:

$RankedDevMatrix$: ranked deviation of data filtered by sensitivity vector

1: **function** CompRankDevVecs($LDmatrix$, $SenVec$)
2: **for** $i = 1$ to M **do**
3: **if** $|SenVec| \neq 0$ and $|LDmatrix[i]| \neq 0$ **then**
4: $DeviationMatrix[i] \leftarrow (i, CoSim(SenVec, LDmatrix[i])$
5: **else if** $|SenVec| \neq 0$ and $|LDmatrix[i]| = 0$ **then**
6: $DeviationMatrix[i] \leftarrow (i, 0)$ ▷ as there is no deviation in data
7: **else if** $|SenVec| = 0$ **then**
8: $DeviationMatrix[i] \leftarrow (i, 1 - \frac{1}{D} \sum_{j=1}^{D} LDmatrix[i, j])$
9: **end if**
10: **end for**
11: $RankedDevMatrix \leftarrow$ sort $DeviationMatrix$ according to similarity values in the 2nd column, descending.
12: **return** $RankedDevMatrix$
13: **end function**

Scenario 1: Looking for significantly conflicting data

Looking for those movies that show high deviation in their data compared to RefItem, ranked from highest to lowest, this can be computed by choosing a sensitivity vector $SenVec = (1, 1, 1)$, i.e.,

$$\textit{CompRankDevVecs(CompDev(RefItem)}, (1, 1, 1))$$

This results in a ranked list of movies containing movies with an entirely different title, release date, and director name at the top of the list, and movies with the identical/same title, release date, and name of director at the end of the ranked list.

Scenario 2: Looking for non-conflicting data

Looking for those movies that show no or lowest deviation in their data compared to RefItem, ranked from non-deviation to high-deviation, can be computed by choosing a sensitivity vector $SenVec = (0, 0, 0)$, i.e.,

$$CompRankDevVecs(CompDev(RefItem), (0, 0, 0)).$$

This results in a ranked list of movie descriptions containing movies with exactly the same title, release date, and director name at the top of the list, and movie descriptions with the diverging title, release date, and name of director at the end of the ranked list.

Scenario 3: Looking for partially conflicting data (1)

Looking for movies that show the highest deviation in the movie title (first attribute), but no or lowest deviation in their other attributes compared to *RefItem*, ranked from non-deviation to high-deviation, can be computed by choosing a sensitivity vector $SenVec = (1, 0, 0)$, i.e.,

$$CompRankDevVecs(CompDev(RefItem), (1, 0, 0)).$$

This results in a ranked list of movies containing movies with an entirely different title, but similar release date and director values at the top of the list, and movies with the identical/same title at the end of the ranked list.

Scenario 4: Looking for partially conflicting data (2)

Looking for movies that show the highest deviation in the movie title (first attribute) and the name of the director (third attribute), but no or lowest deviation in the release date compared to RefItem, ranked from non-deviation to high-deviation, can be computed by choosing a sensitivity vector $SenVec = (1, 0, 1)$, i.e.,

$$CompRankDevVecs(CompDev(RefItem), (1, 0, 1)).$$

The outcome results are a list of movies ranked from the most conflicted movies on the top to the identical ones at the bottom of the list.

3.3 Results

In this section, we want to demonstrate the performance of the proposed approach. The prototype is implemented in Python to gain sufficient information and prove the applicability of our approach.

3.3.1 Dataset

The dataset is extracted from the website IMDb[1]. The data includes different conflicting descriptions for movies. For example, the $Title$ attribute may contain different additional unknown values, such as "25th Hour (2002)" and "25th Hour". Similarly, the $ReleaseDate$ attribute may contain different entries; e.g., "24 March 2008" and "2008.03.24". Additionally, regarding the $Actors$ attribute, some actors may appear for a movie, and another record of the same movie may not contain the same list of actors.

We extracted 500 records (movies). Each record included attributes $Title$, $ReleaseDate$, and $Actors$. The reason of choosing 500 movies is that the considered movies are used as a validation (test) dataset, since the exact number of data conflicts in movies is known. The 500 movies have been checked carefully. Thus, we can investigate the performance of the proposed algorithm. The dataset contains 346 unique movies. The rest, i.e., 154 movie descriptions, can be considered as conflicting ones, deviating from the unique movies, or - as a special case - are exact duplicates of one of the unique movies.

3.3.2 Evaluation

In the experiment we present here, we have chosen 5 sample movies to serve as reference records ($RefItem$). In order to investigate different kinds of deviations of movie descriptions present in the dataset, we applied several sensitivity vectors encoding these different kinds of deviations[2].

Table 3.1 shows examples of the number of top ranked conflicting movies from the validation dataset for the five sample movies serving as reference items. All these reference records (i.e., $RefItem$) consist of three attributes, $Title$, $ReleaseDate$, and $Actors$.

For the illustrated examples, the result for the sensitivity vector $(0, 0, 1)$ for a given movie as $RefItem$ tells us that all the existing conflicting movies in the dataset (see last column) show deviating values for the third attribute, i.e., the list of actors. E.g., when comparing to the reference movie "Going on", all the 9 movie descriptions in the dataset, which show conflicting values

[1] http://www.imdb.com/title/tt0119137/

[2] Because the experiment does not look for non-conflicting data elements, the sensitivity vector $(0, 0, 0)$ is not included in the table.

Reference Movie/ SenVec	**[0 0 1]**	**[0 1 0]**	**[1 0 0]**	**[1 0 1]**	**[1 1 0]**	**[0 1 1]**	**[1 1 1]**	**#Real Conflicting Movies**
5th Hour	6	2	2	2	2	6	2	6
Jump Street	7	1	1	1	1	7	1	7
A Space Odyssey	7	3	3	3	3	4	3	7
Going on	9	2	2	1	2	7	1	9
Til We Meet Again	4	1	1	1	1	1	1	4

Table 3.1: Number of top-ranked conflicting movies for 5 sample movies applying different sensitivity vectors with respect to attributes:
Title, *ReleaseDate*, *Actors*

in the list of actors, but show the same values for *Title* and *ReleaseDate*, are ranked on the top 9 positions by our algorithm. In this case, the algorithm shows excellent accuracy.

The results for the sensitivity vector $(0, 1, 1)$ tell us that for the first two reference movies in the table all the existing conflicting movie descriptions in the dataset (see last column) show deviating values for both, the second (*ReleaseDate*) and third (*Actors*) attributes and all are ranked on the top of the list. For the last reference movie in the table, only one of the four conflicting movie descriptions was top ranked. The other three movie descriptions were not positioned in the top four positions, but ranked lower or handled as exact duplicates of the reference movie and not received as conflicting elements. Table 3.1 also illustrates the variety of deviations one can investigate, as the variations of sensitivity vectors shown in the example encode different kinds of deviations in the movie descriptions.

3.4 Summary of the Chapter

The proposed algorithm could filter the conflicting data well, as it has been presented in Section 3.2.

In our discussion, we consider three essential issues:

(a) The algorithm has a drawback in the current version, which is the $MaxNumbRetrItem$, that is needed to be defined by the user as an input parameter. The limitation is that this number cannot be estimated easily. The solution for that is to estimate the number of conflicted records based on

probabilistic or statistical models.

(b) In reality, consistent sources often exist where the system can easily help with estimating precise information, so the model that concludes from a variety of data types will generate a correct estimation of the source reliability. Therefore, rather than having upright knowledge of individual data types, we have to work on a consolidated system that is capable of making a joint estimation among different types of data simultaneously, although it is not an easy work to unify various types of records in one system. Furthermore, we need to compare how much the source input is relative to the correct answer with respect to the differentiation of data, which demands a different way of handling in the source estimation reliability phase. For categorical data, every given estimation can be considered either true or false.

(c) In case of categorical data, there are only two options for every observation:true or false. It is true when the observation is identical with the correct information, and false if it is not. However, in the case of the contentious values, the situation is different. The values that are close to the correct value can be accepted. For instance, if the correct value of the temperature is 85F, but the observation is 84F, the value can be accepted as a true value since it is so close to the real one, but in some cases, we may fail with the estimation when not taking the source reliability in to account. Thus, we need a platform which can ideally integrate different types of data based on predicting the quality of information.

Regarding the issue (a) this can be overcome by whether estimating the possible number of conflicted data or applying clustering algorithm. In the future work, one could apply clustering approaches to solve this problem.

Regarding issues (b) and (c), using sensitivity vectors does not just return true or false values, rather tuning the sensitivity vectors can fetch exact, less and very similar records with respect to all attributes.

Furthermore, the proposed algorithm overcomes the problem of dealing with different data types, as it has been mentioned in Section 3.2. This challenge has been covered in the preprocessing phase by unifying and cleaning the content of each attribute. The preprocessing phase has the advantage of focusing more on the content. Moreover, it should be mentioned that, although there are numerous machine learning approaches that can be applied

for fact check purposes, applying such techniques still have some disadvantages [46]. For example, for supervised learning techniques, accumulating a sufficient training set presents a challenge in our application. On the other hand, unsupervised learning techniques have advantages, e.g., they can recognize new classes that may not have been considered before. Therefore, this can be done without explicit training that leads to the advantage of unknown classes or unusual classes. Extending the algorithm utilizing unsupervised learning techniques will allow the fact check detection system to perform also in a more autonomous manner.

"People on systems like Facebook are increasingly forming into 'echo chambers' of those who think alike. They will keep unfriending those who don't, and passing on rumors and fake news that agrees with their point of view." -STARR ROXANNE HILTZ

CHAPTER 4

Fact-Checking for Unstructured Data

4.1 Behavioral-Based Analysis

This part of this chapter is based on the following paper *Declarative Programming Approach for Fake Review Detection* [54].

4.1.1 Introduction

The growing portion of unstructured text on the Web is massive. Text repositories come from Web 2.0, e-commerce, and social media platforms, publications, directories, and portals disseminate different types of data in the form of unstructured data. The published data includes news reports, jobs announcements, movie reviews, product reviews, real estate registrations, restaurant reviews, and a lot more. One can say that 80-90 percent of forthcoming development of data is available in the form of unstructured text databases that include interesting patterns and trend [125]. Since online reviews play an essential role in our daily lives, robust approaches for detecting fake reviews are increasing demand. The following work represents an approach to detect fake reviews as an example of unstructured textual data, incorporating the behav-

ior of authors of the text combined with properties derived from the content of their published text.

4.1.2 Criteria defining fake online reviews

According to [71], different criteria are checked to spot fake reviews; typically, they are classified into four classes:

- Content-Centric
  This class refers to the content of the published review, and it can be identified by the content similarity to the other written texts [80] [50], the length of the text, by utilizing many digits and capital words, providing a personal connection details and the similarity between the item description and the written text.

- Online-Users-Centric
  When directing to this classification, features of user behavioral patterns are also a concern, for instance, the reported opinion regarding the given rating [71]. Some valuable measures can be the rating deviation, number of written texts by a specific user, the active duration of the person who published the text, etc.

- Time-Centric
  Here, time-related measures are a concern to witness the people who publish fake reviews; they are often called spammers [122]. One feature can be the early published item duration, meaning that a user writes a review soon after the item was announced online. Another measure of concern is the period between the reviewer's arrival time on the webpage and his written text.

- Relationship-Centric
  This measure concerns the relationship between the user's interactions so that groups of fake reviewers can be recognized. This class can be detected by the spammer's content similarity or deviations [128]. All these features can be an indicator of untrustworthy texts.

After looking intensely at the characteristics of fake reviews and the behavior of the online users, the result determination is to design a white-box approach

that is becoming a requirement nowadays in the industry and the research. This is since there are increasing social concerns about decisions made based on personal information. In other words, we seek to design a white-box model that can let users understand what is going on regarding their personal data. In contrast to black-box models, such as deep-learning that are hard to be explained in general.

Consequently, we propose a rule-based fake review detection system using Answer Set Programming (ASP) which is a powerful tool to declare malicious behavior patterns specified via a variety of constraints. This way we can create powerful models that combine, e.g., information about the number of reviews, the number of dislikes, the analysis of the point in time reviews have been written, qualitative properties of the content based on similarity measures and derived classification of reviews and products.

4.1.3 On Answer Set Programming and Declarative Problem-Solving

Declarative Programming has been utilized in a notable variety of projects and is a potent mechanism for knowledge representation and reasoning KRR), mainly for its characteristics of being highly demonstrative and dealing with incomplete knowledge. In answer set programming, the coder of the problem focuses on declaring the problem, while the output of each answer set represents a one-to-one resolution to the declared problem at hand.

The bases of answer set programming are mainly logic-based knowledge representation, constraint decoding, and satisfiability check. It is commonly used to solve P and NP problems, especially in the applied disciplines of planning, optimizing, and decision supports. As described in [34] and [93] ASP got popular "mainly due to its appealing combination of a rich yet simple modeling language with high-performance solving capacities". In ASP, search problems are reduced to compute stable models and answer sets. ASP uses solvers that are reasoners, which lead to stable model generation. It belongs to the group of so-called constraint programming languages [34].

Figure 4.1 shows the solving steps in ASP. The steps can be described as following:

- Starting with understanding the problem

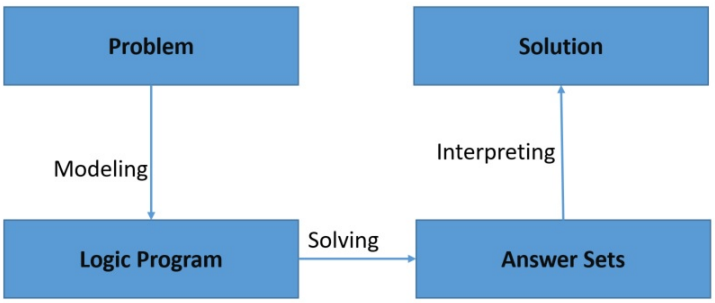

Figure 4.1: Solving steps in ASP

- Model the problem with a logic program
- Solving the problem is done by a solver
- The solver gives answer sets
- Answer sets help to interpret the solution

The goal is to achieve one or multiple solutions to the given problem. The first step is to model the problem into a logic program that consists mainly of rules, first-order predicates, and variables. Afterward, the grounder will eliminate the variables by replacing them with ground terms in the language. As a result, a propositional program will be produced that only consists of propositional atoms[1]. The program is then taken by the solver, which will compute all the stable models of the program and interpret the solution [32]. Thus, as we see, to start building our knowledge base, we need first to model and understand the data and the problem itself. This allows for creating a formal model of the problem and the input data through a knowledge base, constraints, and rules that have to be satisfied (see Sections 5.3.5 and 5.3.5. In the illustrating example, we show the approach considering only three movies, just for simplicity. This simplification does not cause any limitation of the approach.

[1]https://potassco.org

4.1.4 Demo Application Scenario

In our demo application, we assume that there are movies, each of them having assigned some reviews written by some users. Each review has assigned several dislikes reflecting the opinion of other users, a timestamp encoding the time the review was posted, and an IP address encoding the user's machine as the source of the review.

All information about movies, reviews, dislikes, timestamps, and IP addresses are encoded in a knowledge base which we briefly discuss in section 5.3.5.

In our application, it is assumed that reviews may be considered *fake* or *genuine*. We include many rules that will lead us to detect *fake* and *genuine* reviews, *spam* reviews, and classify movies as being *goodReviewed* or *badReviewed* helping the users to decide whether a review on a certain movie can be trusted or not.

All these rules determine the results, i.e., labeling a review as *fake* or *genuine*, determining which reviews were posted from the same IP address, and labeling reviews as candidates for spam in case they have been written within a short time interval and stemming from the same IP address (e.g., a pattern often observed in the case of sockpuppetry [64]). In Section 5.3.5 we illustrate some sample rules.

Knowledge Base

In the knowledge base (see Listing 4.1), first, all constants should be declared. In our example scenario, we declare the threshold *fake_rev_threshold* as the number of dislikes of a review used to decide whether a review is treated as a fake review or genuine review. The threshold is the average number of dislikes for fake reviews which to be found in the training data.

```
1
2  #const fake_rev_threshold = 18.
3
4  #const t1 = 2. t2 = 4.
5
6  #const t3 = 6. t4 = 7. t5 = 8. t6 = 13.
7  ...
8
9  movie(m1;m2;m3; ...;mn).
10
11 review(a;b;c;...;t;u;...;zn).
12
13 hasRev(m1,(a;d;g;j;m;p;s;c)).
14 hasRev(m2,(b;e;h;k;n;q;t)).
15 hasRev(m3,(c;f;i;l;o;r;u)). ...
16
17 dislike(a,19).
18 dislike(b,11).
19 dislike(c,12).
20 dislike(s,21).
21 dislike(t,15).
22 dislike(u,13).
23 ...
24
25 timestamp_of_Review(a,t1).
26 timestamp_of_Review(b,t2).
27 timestamp_of_Review(c,t3). ...
28
29 hasIp(a,131120001001).
30 hasIp(b,145120145032).
31 hasIp(c,131120001001).
32 hasIp(s,230112140099).
33 hasIp(u,166224110015).
34 hasIp(t,166224110015). ...
35
36 reviewLabel(fakeReview).
37 reviewLabel(genuineReview).
38
39 movieLabel(goodReviewed).
40 movieLabel(badReviewed).
41
42
43
```

Listing 4.1: Abstracted samples of facts from the knowledge base

Next we declare some timestamps t_i, followed by the declaration of movies m_i and reviews $a, b, c, ...$ Subsequently, we declare which reviews are assigned to a movie (see *hasRev*) and how many dislikes each review received (see *dislike*). Then the timestamps and the source IP addresses for all reviews are

modelled (see *timestamp_of_Review* and *hasIp*). Finally, we declare the labels *fakeReview* and *genuineReview* used for the rating of reviews, and the labels *goodReviewed* and *badReviewed* for eventually classifying movies.

```
1
2  #1:
3  { rev_with_label(X,C) : reviewLabel(C) } == 1 :- review(X)
       .
4  { movieClassified(X,C): movieLabel(C)} == 1 :- movie(X).
5
6  #2:
7  :- rev_with_label(X,fakeReview), dislike(X,Y),
8     Y<fake_rev_threshold.
9
10 #3:
11 :- rev_with_label(X,genuineReview), dislike(X,Y),
12    Y>=fake_rev_threshold.
13
14 #4:
15 sameIp(X1,X2):- review(X1), review(X2), hasIp(X1,Y1),
       hasIp(X2,Y2),
16                 Y1=Y2, X1!=X2.
17
18 #5:
19 % testing for a pattern based on sockpuppetry behaviour:
20 spam(X;Y):- hasRev(M1,X), hasRev(M2,Y), M1=M2, X!=Y,
21             hasIp(X,I1), hasIp(Y,I2), I1=I2,
22             review(X), timestamp_of_Review(X,Z1),
23             review(Y), timestamp_of_Review(Y,Z2),
24             |Z2-Z1|>3, |Z2-Z1|<6.
25
26 #6:
27 % testing for some other pattern of "bad" behaviour:
28 spam(X; Y):-
29      "some other pattern that indicates spam candidates."
30
31 #7:
32 spamFromSameIp(X,Y) :-  spam(X), spam(Y),
33                     hasIp(X,I1), hasIp(Y,I2), I1=I2.
34
35 #8:
36 goodMovie :- movieClassified(X,goodReviewed),
37           hasRev(X,Y),
38           not rev_with_label(Y,fakeReview).
39           :- not goodMovie.
40
41
```

Listing 4.2: Constraints and Rules

Constraints and Rules

Based on the facts from the knowledge base, constraints, and rules satisfying conditions on the facts are declared such that they formally encode the problem that needs to be solved. In our scenario the problem we want to solve can be stated like this: *Given some facts about reviews on movies we want to detect fake and genuine reviews, identify candidates for spam reviews, and classify movies as being well-reviewed or badly reviewed.*

In our case, a set of illustrating constraints and rules (see Listing 4.2) has been declared which state the following:

1. Each review and movie has to be assigned a single label. For example, for reviews it has to be one of *fakeReview* or *genuineReview*, for movies it has to be one of *goodReviewed* or *badReviewed*.

2. The given constraint states that the number of dislikes for each genuine review should not be greater than the threshold *fake_rev_threshold*.

3. The given constraint states that the number of dislikes for each fake review should be greater than or equal to the threshold *fake_rev_threshold*.

4. The given rule states that if source IP addresses are identical for two different reviews then this fact of having the same IP addresses is derived and concluded for the final solution (see *sameIp*).

5. The given rule states that if two different reviews of a certain movie have an identical source IP address and both reviews were written within a short time interval (e.g., $|z_2 - z_1| > 3$ and $|z_2 - z_1| < 6$) it has to be derived and concluded that both reviews are spam candidates.

6. Additional rules can test for some other patterns of 'bad' behaviour like sockpuppetry and spamming that might indicate further spam candidates.

7. The given rule states how the special subclass of spam reviews originating from the same IP address is determined.

8. The given rules state that a movie having assigned the label *goodReviewed* should not have any fake reviews.

4.1.5 Computed Results

The results are computed based on an ASP solver, and the programming language used is Python. We used the Clingo ASP package that is available from Potassco[2], and also offers an online Answer Set Collection. The results computed from our sample of facts, constraints, and rules shown in Listings 4.1 and 4.2 include the assignment of *fakeReview* and *genuineReview* to reviews, the identification of reviews originating from the same IP address, the determination of spam candidates for reviews, and the assignment of *goodReviewed* and *badReviewd* to movies. Note, a result computed by an ASP solver always fulfills all the constraints and rules. Hence, the computed result can be considered an optimal solution to the problem with the highest accuracy, precision, and specificity compared to all the papers we have mentioned previously from the state-of-the-art. Most of the papers are focusing on machine learning approaches and natural language processing to tackle the fake review's detection problem.

```
1
2  rev_with_label(a,fakeReview).
3  rev_with_label(k,fakeReview).
4  rev_with_label(n,fakeReview).
5  rev_with_label(p,fakeReview).
6  rev_with_label(q,fakeReview).
7  ...
8  rev_with_label(b,genuineReview).
9  rev_with_label(c,genuineReview).
10 rev_with_label(d,genuineReview).
11 rev_with_label(e,genuineReview).
12 ...
13 rev_with_label(r,genuineReview).
14 rev_with_label(t,genuineReview).
15 rev_with_label(u,genuineReview).
16
17 sameIp(c,a). sameIp(u,t).
18 spam(a). spam(c). spamFromSameIp(a,c).
19
20 movieClassified(m3,goodReviewed).
21 movieClassified(m1,badReviewed).
22 movieClassified(m2,badReviewed).
23
24
```

Listing 4.3: Sample results computed by the ASP solver

[2]https://potassco.org/

Listing 4.3 illustrates some part of the computed solution to the problem modelled by the illustrating knowledge base, constraints and rules.

In ASP, the maximization and the minimization are covered well and can be declared using the instructions #maximize and #minimize. In our sample scenario, we only have the maximization function that maximizes the number of dislikes each review has. We plan to include a minimization function that will refer to the ratio of positiveness each review may have. The input data for the positiveness ratio will be generated by a sentiment analyses module. Obviously, the set of constraints and rules can be easily extended to cover much more complicated scenarios built upon a variety of behavior patterns, and not just a few characteristics the spammers or the spams may have, which are well understood and studied in the literature.

To check our ASP-based detection approach, we performed several tests using our reviews that have been created in our laboratory. In particular, we considered the 217 reviews as we have described in Section 7. The system could detect all 70 fake reviews as fake out of all 217 reviews, which proofs that the formulated rules in our ASP are well-defined. The overall execution time was between 0.007 seconds using Laptop Hardware CPU Intel Core i7-8650U @ 1.90GHz, RAM 16.0 GB, OPERATING SYSTEM Windows 10, 64-bit operating system, x64-based processor.

4.1.6 Summary of the Approach

The problem of fake review detection is a complex problem. This is due to the fact that fake reviews are related to at least two major sources: first, the content of the review and, second, the behavior of the reviewer. Usually, the content can be analyzed and classified using natural language processing techniques. However, defining the behavior of a fake reviewer is a complex problem. There are different spatial-temporal patterns of behavior, which can be modelled as part of an optimization problem, and because of the complexity—difficult to solve and consuming a lot of time and computational resources. ASP is an interesting and powerful tool to address such problems, in particular offering a declarative way to specify the problem and properties of the desired solution(s). ASP inherits the advantage of rule-based approaches e.g., (a) flexibility which means that adding new rules or updating the system

requires only few changes compared to other paradigms, (b) it does not require a huge amount of training data, and (c) rule-based techniques are considered a white box, which means that it provides traceability and transparency for critical decisions that demand a higher degree of explanation than usually provided by machine learning techniques. In contrast to that, our approach shows some limitations, for example, some thresholds that are used during rules definition must be extracted from some data samples using statistical analyses, which means that any changes in the behavior of fake reviews may require a new calculation of the thresholds. However, such threshold values can be calculated regularly and automatically based on the information system under observation.

It should be mentioned that black-box or gray-box machine learning approaches are not preferred in such an application because we cannot understand the system's decisions. Moreover, on what bases should we trust the output of such models? Furthermore, the decision whether to go for a rule-based system or learning system depends on the problem, and it is always a trade-off among efficiency, training costs, and understanding. There are many successful rule-based detection systems in the state of the art [106] [47] [112].

4.1.7 Conclusion

This approach proposes a way to create a recognition system for detecting fake reviews, spams, and spammers on the Web introducing fake reviews. Looking at the different types of user behavior known from state-of-the-art literature, a set of illustrating constraints and rules have been designed such that the problem could be transformed into an optimization problem that can be solved using Answer Set Programming (ASP).

Such rule-based systems and the knowledge base they are based on need to be well-designed. This then allows for achieving a higher degree of accuracy and a properly working model. A proper design also leads to an extensible model, which allows us to process much more complex scenarios at the same level of high accuracy. One of our next steps is to expand the model by integrating a sentiment analyses approach, processing the content of reviews.

This work shows the high capability of ASP to solve such problems, and we believe that much scientific work can be done by further developing our

proposed approach for fake review detection.

4.2 NLP-Based Analysis

This Part of this chapter is based on the following paper *A domain-independent classification model for sentiment analysis using neural models* [52].

4.2.1 Introduction

Sentiment analysis is the task of recognizing positive and negative opinions of users regarding different purposes, e.g., users' opinions about movies, products, music albums, and many other fields. To provide a better definition, the sentiment is referred to as a judgement, opinion, attitude, or emotional state prompted by feeling. Sentiment analysis is an automated process in which, by using the natural language processing (NLP), the subjective information is computationally identified, analyzed, and classified into positive, negative, or neutral to specify the sentiment of that text which is the result of its author's attitude [15]. There are different types of sentiment analysis, the most popular types are classified and described in the following:

1. Grained Sentiment Analysis: The results in this type are more than binary classification results where two labels (positive, negative) are presented. It is achieved in fine-grained granularity varying from strong negative, weakly negative, neutral, weakly positive to strong positive based on the determined polarity, mainly used when the polarity precision is highly important and binary results like negative or positive could not be useful and may provide incorrect classifications [119].

2. Emotion detection: It classifies different emotions in the text such as fear, anger, sadness, joy, disgust, etc. Sophisticated machine learning algorithms [104] are used to detect emotions for different goals.

3. Aspect-based Sentiment Analysis: The results in this type are achieved after splitting the text into different aspects and then assign each aspect a corresponding sentiment. For instance, the result of aspect-based sentiment analysis on a special product's review "It is so easy to use but Insanely Expensive" would be (a) Ease of use: positive and (b) Price:

negative due to the nature of this type, it is mostly utilized in customer-centric businesses to have a deeper understanding of customer's requirements [126].

The sentiment analysis field has many applications. For example, in businesses and organizations, they need to find consumer or public opinions regarding their products and services. Individual consumers may also need to know the evaluation of other users of a product before purchasing it. Moreover, they might be interested in others' opinions concerning political candidates before making a voting decision in a political election. Furthermore, nowadays, data are published enormously and freely on the Web, but with no data quality assurance, it is left to the readers to decide whether they believe it or not. This results in high demand for advanced fact checking techniques and applications that contribute to the assurance of data quality. In particular, users surfing the Web are more often inflicted with harm/damage by inconsistent information. In addition, designing and developing such fact checking systems need robust models of sentiment analysis. This task for fact checking detection can be fulfilled when fact checking systems are provided by a general or universal model that can be trained once and then applied to other reviews. Surely, this model should show high performance to increase the accuracy of such fact checking systems.

Hussein in [45] discussed the importance and effects of the challenges in sentiment analysis is the domain-dependence. Moreover, the author concluded that the nature of the topic and the review structure determine the suitable challenges for the evaluation of sentiment reviews. Hence, building a generalized model is a challenge that should be considered by researchers in this research field.

This work focuses on providing a generalized model for sentiment analysis. It has two main contributions: (a) it shows that convolutional neural networks (CNN) combined with our review to vector algorithm can lead to design models that can be trained once and work well using other types of data that might even be related to a different domain and (b) it shows high performance using different precision metrics compared to other approaches from the state of the art that use same datasets for evaluation. We believe that this is one of the few works that address the generalization capabilities

of deep models, w.r.t. domain-Independence.

4.2.2 Similar Approaches

Although linguistics and natural language processing (NLP) have a long history of research, few works were published concerning sentiments before the year 2000 [73]. After 2000, the field has attracted the attention of researchers and many research groups to work on.

To provide an example, [77] studies the prediction of every review for being negative or positive in the aspect-oriented opinion in the opinion mining domain at the sentence level. The authors of this work propose groups of selected models based on conditional random fields (CRFs) with an added multi-label presentation that not only models the opinion in a review, but also models a set of opinions in a single review. In [100], the authors suggest a sentiment analysis system that can identify and relate the sentiment to every rated product or item in the reviews. They present a probabilistic model to investigate the structure of each review and to which cluster each of them is related to, where it represents a specific sentiment.

In [130], the researchers offer a flexible automated classification system that uses supervised machine learning techniques using Markov Logic for sentiment classification on a sub sentence level and incorporates polarity differentiation from different origins.

Furthermore, in [115], an enhanced latent aspect rating analysis model is presented. This model does not require predefined keywords that are associated with specific aspects. This work investigates the reviews to define the topical aspects, the ratings of the individual aspect and assigning weights that differentiate depending on the aspects from a reviewer perspective. [129] proposes a simple hierarchical clustering approach (unsupervised model) for product aspects extraction, clustering, and also defining the relations between aspects (relevant and irrelevant). In [38], the authors introduce a novel supervised approach for joint topic aspects for choosing specific reviews that are considered to be helpful among a set of reviews. Moreover, in [7], an employee dataset is created and a novel ensemble model for sentiment analysis is proposed on aspects level. In [96], a sentiment analysis is conducted on movie reviews. New features are extracted that have influence on determining the

polarity scores of the opinion more accurately. Natural language processing approaches are applied using the impact of the unique extracted features. In [95], supervised and semi-supervised approaches are investigated for text classification. Additionally, deep learning is also used for sentiment analysis, e.g., in [95], authors use bidirectional long-short term memory models which are applied to the IMDb dataset.

Table 4.1 shows a summary of the state-of-the-art approaches for sentiment analysis. More information regarding the performance of different approaches can be found in Section 4.2.7.

Based on the previous works, this research has started to be one of the highlights for scientific contributions because: (a) it has different applications for recommender systems and fact checking systems, and (b) it contains several challenging research problems that motivate researchers to work and improve their works on them.

4.2.3 The Approach

The work presented in this chapter proposes an intensive study regarding domain-independent classification models for sentiment analysis that should be trained only once. The study consists of two phases: the first phase is based on a deep learning model, which is training a neural network model once after extracting robust features and saving the model and its parameters. The second phase is based on applying the trained model on a totally new dataset, aiming at correctly classifying reviews as positive or negative. The proposed model is trained on the Internet Movie Database (IMDb) dataset and then tested on three different datasets: IMDb dataset, Movie Reviews dataset, and our own dataset collected from Amazon reviews that rate users' opinions regarding Apple products. The work shows high performance using different evaluation metrics compared to the stat-of-the-art results. In this section, we present the preprocessing, the review to vector algorithm and the design details of the proposed neural models for sentiment analysis, and then the evaluation metrics and the overall evaluation. We aim at training a neural model once using a batch of IMDb dataset and test it on other reviews' datasets to see how far the generalization is possible.

Paper	Dataset	Labels	Approach
[77]	Hotel Reviews	Multi-labels	Supervised machine learning techniques using conditional random fields models for aspect detection sentiment analyzing.
[100]	Restaurant reviews, medical descriptions, Yelp	Two labels	Unsupervised machine learning technique using probabilistic topic modeling approaches for sentiment content clustering.
[130]	Product reviews	Two labels	Supervised machine learning techniques using Markov logic for sentiment classification.
[115]	Hotel Reviews and MP3 player product review from Amazon	5 star rating	Unsupervised machine learning techniques for Latent Aspect Rating Analysis Model.
[129]	Chinese product reviews	Two labels	A hierarchical clustering approach for product aspects extraction and clustering.
[38]	Companies employee reviews	Ratings	A novel supervised joint topic model approach to select helpful reviews among a set of reviews.
[7]	Different products reviews	Ratings	A novel hybrid approach to implement aspect-level sentiment analysis that assigns sentiment labels to the reviews.
[96]	IMDb	Two labels	N-grams followed by a random forest classifier.
[95]	IMDb	Two labels	Bidirectional LSTM.
[88]	IMDb	Two labels	Maximum entropy classification combined with support vector machines using unigrams and bigrams.
[97]	IMDb	Two labels	Lexical filtering.
[23]	IMDb	Two labels	Context-Free Grammars (CFGs).
[61]	Movie Review	Two labels	Convolutional Neural Networks (CNN).
[102]	Movie Review	Two labels	Novel machine learning frame-work based on recursive autoencoders.
[111]	Movie Review	Two labels	Multiple classifiers—a hybrid approach.

Table 4.1: A summary of the state-of-the-art approaches for sentiment analysis.

Review to Vector

Before features extraction, we removed the stop words from the given dataset, e.g., "the", "a", "an", and "in". Then, the next step includes removing punctuation. In this step, we extracted feature elements from a batch of IMDb dataset for positive and negative reviews. The batch size has been determined using grid search (see Section 4.2.6). We formulated a function which works like a dictionary where the keys are the words in the text and the values are the count associated with that word. The output is saved in $word\_features$.

Algorithm 3 illustrates the procedure of converting reviews to vectors. It takes two inputs and returns the input vectors for all reviews saved in $all\_features$. The output vectors will be fed later into our proposed neural models The input parameters are $word\_features$ which is the first 4900 words after calculating the frequency distribution of each word from both training data, mainly the positive and negative reviews. The number 4900 words have been selected after applying grid search using different lengths which give the highest performance. The second input parameter is the $reviews$. The summary of the algorithm is as follows: for each review in the reviews, the function $word\_tokenize$ splits the current review into sub-strings (words). After that, for each word in $word\_features$, it should be checked whether that word is a word in the current review. If yes, 1 is added to the features list or 0 is added if it is not. Finally, the function returns $all\_features$, which is a matrix. The vectors of the review to vector algorithm (see Algorithm 3) are used for training different classification models (see Section 4.2.3).

Classification: Convolutional Neural Network (CNN)

To perform the sentiment classification task, we propose a neurocomputing based approach. A CNN is a kind of feed-forward neural network that consists of multiple layers of convolutional filters followed by sub-sampling filters and ends with a fully connected classification layer. The classical LeNet-5CNN was first proposed by LeCun et al. [67], which is the basic model of different CNN applications for object detection, localization, and prediction. First, the output vectors of the review to vector algorithm are converted to matrices, where the goal is to make the application of CNN model possible.

As illustrated in Figure 4.2, the proposed CNN model has one convolu-

tional layer, one sub-sampling layer, and an output layer. The convolutional layers generate feature maps using five (2×2) filters followed by a Scaled Exponential Linear Units (SELU) [63] as an activation function. Additionally, in the sub-sampling layers, the generated feature maps are spatially dissembled. In our proposed model, the feature maps in layers are sub-sampled to a corresponding feature map of size 2×2 in the subsequent layer. The final layer, which is a full CNN model that performs the classification process, consists

Algorithm 3 The review to vector algorithm

```
1: Input: word_features, Reviews
2: function REVIEW_TO_VECTOR(Reviews)
3:     all_features = [ ]
4:     for review in reviews do
5:         words = word_tokenize(review)
6:         features =[ ]
7:         for w in  word_features do
8:             if (w in words) then
9:                 Features[w] = 1
10:            else
11:                Features[w] = 0
12:            end if
13:        end for
14:        all_features+=[Features]
15:    end forreturn all_features
16: end function
```

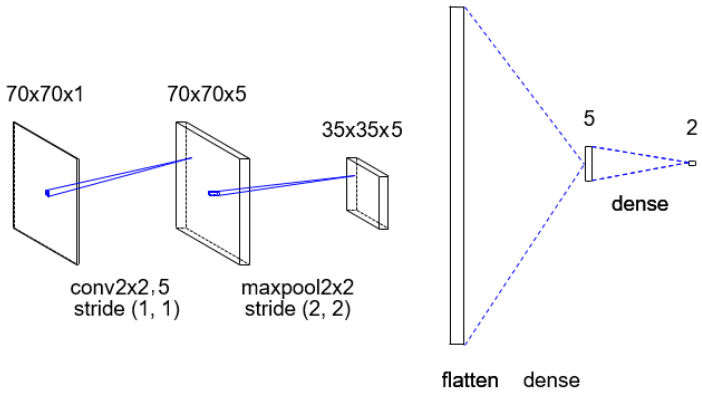

Figure 4.2: The proposed CNN model

Layer	Kernel, Units	Other Layers Parameters
Convolution	(2 × 2), 5	Activation = Selu, Strides = 1
Max Pooling	(2 × 2)	Strides = 2
Dropout	0.75	
Fully Connected		Units = 5, Activation = Selu
Softmax	-	Numbr Of Classes = 2

Table 4.2: Parameters used for all the layers of the proposed CNN model.

of three layers. The first layer is the input layer which has 6125 nodes and the second that has five nodes. The final layer is the softmax output layer. The result of the mentioned layers is a 2D representation of extracted features from input feature map(s) based on the input features from the reviews.

The proposed CNN consists of one convolutional layer and a max-pooling layer is because the small size of our input dimension does not require additional layers to extract features/patterns. The reason for using SELU is due to the fact that (a) SELUs performed better than Rectified Linear Units (RELUs), (b) SELUs offer self-normalization [63], and (c) they never lead to vanishing gradients' problem. Since the dropout is a regularization technique to avoid over-fitting in neural networks based on preventing complex co-adaptations on training data [43], our dropout for each layer was 0.75, which is related to the fraction of the input units to drop. The proposed CNN model has been trained on IMDb. Then, the model has been saved to be tested on other datasets. The length of the considered feature vectors is 4900 words that are converted to matrices of size (70 × 70). The parameters of CNN are selected by using grid search from a Scikit-learn library considering different settings. Table 4.2 shows parameters used for all the layers of the proposed CNN model.

Classification: Shallow Neural Network (SNN)

To perform the sentiment classification task, we use a neural model [44, 30]. First, the output vectors of the review to vector algorithm are fed into the neural model to the hidden layer which consists of three neurons and a hyperbolic activation function; then, the final layer is the output layer which consists of a softmax activation function, Adam optimizer [62], and a cross

Layer	Units	Other Layers' Parameters
Hidden layer	3	Activation = Hyperbolic tangent activation (tanh)
Output layer	2	Activation = softmax

Table 4.3: Parameters used for all the layers of the proposed SNN model.

entropy loss function. The parameters are selected by using a grid search from Scikit-learn library (`https://scikit-learn.org`, see Figure 4.3), where the optimizer is Adam and the loss function is the binary cross entropy.

The proposed neural network model has been trained on a batch of IMDb datasets. Then, the model has been saved to be tested on other datasets. Table 4.3 shows parameters used for all the layers of the proposed CNN model.

Classification: Others

Additionally, we examine several classifiers to compare the performance of the existing models and the proposed ones, particularly Support Vector Machines (SVM) [19], K–Nearest Neighbor (KNN) [4], Naive Bayes [120], and Random Forest [13]. In addition, selecting the previous classifiers has different advantages such as the objective of random forests that they consider a set of high-variance, low-bias decision trees, and the ability to convert them into a model that has both low variance and low bias. On the other hand, K-nearest neighbors is an algorithm which stores all the available cases and classifies new cases based on a similarity measure (e.g., distance functions). Therefore,

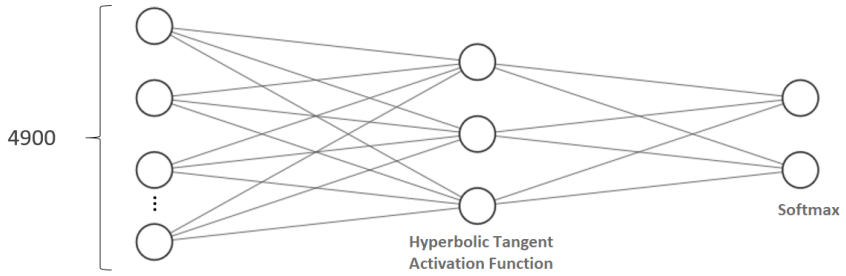

Figure 4.3: The proposed SNN model.

KNN has been applied in statistical estimation and pattern recognition from the beginning of 1970*s* on as a non-parametric technique [4].

SVM are well-known in handling non-linearly separable data based on their nonlinear kernel; e.g., SVM with a polynomial kernel (SVM (poly)) and the SVM with a radial basis kernel (SVM (rbf)). Therefore, we classify the reviews data using three types of SVMs; the standard linear SVM (SVM (linear)), SVM (poly), and SVM (rbf). Finally, we used a simple probabilistic model, which is the Naive Bayes. The purpose of using such a probabilistic model is to show how it behaves w.r.t. in different contexts.

Table 4.4 shows values of parameters for the proposed SNN, CNN, and all other classifiers.

4.2.4 Evaluation Metrics and Validation Concept

To evaluate the overall performance of the classifiers, we consider several performance metrics. In particular, we use precision, recall, f-measure, and accuracy, as in [91].

Equations (4.1)–(4.4) show mathematical expressions of the metrics accuracy, precision, recall, and f-measure, respectively, where TP, TN, FP, and FN refer respectively to "True Positives", "True Negatives", "False Positives", and "False Negatives", respectively:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{4.1}$$

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4.4}$$

Model	**Parameters**
SVM (poly)	Degree of the polynomial kernel function = 3, $\gamma = \frac{1}{numberoffeatures}$
SVM (rbf)	$\gamma = \frac{1}{numberoffeatures}$
Random Forest	Number of estimators estimators = 10 trees, criterion = Gini impurity, The minimum number of samples required to split an internal node = 2
Naive Bayes	Prior = probabilities of the classes
Proposed (CNN)	Loss = Softmax, optimizer = Adamax, $batch\_size$ =1000, epochs = 30
Proposed (SNN)	Loss = cross entropy, optimizer = Adam, $batch\_size$ =128, epochs = 40, lr = 0.001, $beta_1 = 0.9$, $beta_2 = 0.999$, epsilon = 0.01, decay = 0.0

Table 4.4: Values of parameters of proposed CNN, SNN, and other classifiers.

Regarding the evaluation scenarios, we consider two cases: the domain-dependent and domain–independent cases. Domain-dependent means training and testing have been performed for each dataset. Domain–independent means the training has been performed on IMDb datasets of subjects and testing has been performed on a totally new datasets. The reason for training on IMDb is due to its large size and thus can support a better generalized model if the training has been preformed and regularized properly.

4.2.5 Datasets

1. IMDb Dataset ACL-IMDb [76] dataset is a collection of reviews that are taken from Internet Movie Database (IMDb). The dataset size is 50 K and contains highly polar movie reviews annotated as positive or negative review, which makes it widely used for a binary classification tasks. The average length of a document in the training set is 25 k for training and 25 k for testing. The dataset also contains an additional bag of words formats and raw texts ([3]).

2. Movie Reviews (MR) Movie Reviews (MR) is a small sized dataset[4] [87] (5 k positive and 5 k negative reviews) that contains reviews in the form of labeled sentences, which can be specified as objective or subjective. Furthermore, the selected sentences have been gathered from IMDb and Rotten Tomatoes websites ([5]), each selected sentence contains at least 10 words. The sentiments of these sentences have been classified as positive or negative.

3. Amazon Dataset (Amazon) Reviews data from iPhone wireless earphones on Amazon were collected. Overall, we collected 480 negative reviews and 480 positive ones in order to use the data to check the overall performance of the proposed model. We annotated the data with 0 or 1 where every positive review is annotated or labeled by 1 and the negative reviews were annotated by 0.

4.2.6 Results

In this section, we want to demonstrate the performance of the proposed approach. The prototype is implemented in Python. In order to gain sufficient information and prove the applicability of our approach, the following libraries have been used: Natural Language Toolkit (NLTK) ([6]) for natural language processing, Keras ([7]) for deep learning and Scikit-learn library([8]) for machine

[3] http://ai.stanford.edu/~amaas/data/sentiment/
[4] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[5] https://www.rottentomatoes.com/
[6] https://www.nltk.org/
[7] https://keras.io/
[8] https://scikit-learn.org/stable/

learning, which is mainly used for testing the performance of the other classifiers. We applied 10-fold cross-validation for performance evaluation. The neural models have been trained on a GeForce GTX 1080-NVIDIA ([9]).

In order to conduct experimental results and check the performance of the proposed approach, we tested the algorithm using the extracted features based on three datasets, namely IMDb, Movie Reviews, and Amazon reviews.

To evaluate the overall performance of the classifiers, we consider several performance metrics. In particular, we use precision, recall, f1, and accuracy, as in [91]. Regarding the evaluation scenarios, we used the trained model in Section 4.2.3. Tables 4.5, 4.6 and 4.7 present the precision, the recall, and the f-measure using IMDb, Movie Reviews, and Amazon datasets, respectively.

Classifier	**Precision**	**Recall**	**F-Measure**	**Accuracy**
Random Forest	0.73	0.73	0.73	0.73
Naive Bayes	0.64	0.59	0.56	0.59
SVM (poly)	0.24	0.29	0.33	0.49
SVM (rbf)	0.24	0.29	0.33	0.49
Proposed SNN	0.87	0.87	0.87	0.87
Proposed CNN	0.81	0.81	0.81	0.81

Table 4.5: Performance metrics for IMDb, where SVM (poly): Support Vector Machine using a polynomial kernel, SVM (rbf): Support Vector Machine using a radial basis function kernel.

Classifier	**Precision**	**Recall**	**F-Measure**	**Accuracy**
Random Forest	0.67	0.66	0.65	0.66
Naive Bayes	0.58	0.58	0.57	0.58
SVM (poly)	0.23	0.48	0.31	0.48
SVM (rbf)	0.75	0.61	0.55	0.48
Proposed SNN	0.82	0.82	0.82	0.82
Proposed CNN	0.75	0.75	0.75	0.75

Table 4.6: Performance metrics for MR, where SVM (poly): Support Vector Machine using a polynomial kernel, SVM (rbf): Support Vector Machine using a radial basis function kernel.

[9] `https://www.nvidia.com/de-de/geforce/products/10series/geforce-gtx-1080/`

In all tables, the proposed neural models show the highest performance compared to random forest, which is hereby the next best classifier. However, the support vector machine using a radial basis function kernel also performs well for the IMDb dataset, but the Naive Bayes classifier performs much better using our own Amazon dataset.

Additionally, it is remarkable to realize that the proposed shallow neural network performs better than the proposed convolutional neural network model. However, random forest and our proposed neural models show a robust behavior regarding sentiment classification.

Moreover, it should be realized that some classifiers show high-precision and low recall or vice versa where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. This reflects the complexity of this classification task and shows the robust performance of the proposed neural models.

Furthermore, the trained neural models are applied on different datasets that are not related to each other (see Table 4.8). Despite this fact, it still behaves well and, consequently, it can be extended for different applications in the research field of sentiment analysis.

Figures 4.4 and 4.5 show the 10-folds cross-validation results for the trained CNN and SNN models, respectively. In addition, they show the mean accuracy and the standard deviation for each fold. We can observe a reasonable symmetric distribution and that the mean captures the central tendency well. The cross-validation results belong to the pre-trained model, which has been applied to calculate the results in Table 4.8.

Classifier	**Precision**	**Recall**	**F-Measure**	**Accuracy**
Random Forest	0.80	0.80	0.80	0.80
Naive Bayes	0.71	0.67	0.67	0.67
SVM (poly)	0.31	0.55	0.40	0.55
SVM (rbf)	0.31	0.55	0.40	0.55
Proposed SNN	0.77	0.76	0.74	0.74
Proposed CNN	0.67	0.67	0.67	0.68

Table 4.7: Performance metrics for Amazon reviews, where SVM (poly): Support Vector Machine using a polynomial kernel, SVM (rbf): Support Vector Machine using a radial basis function kernel.

Classifier	Precision	Recall	F-Measure	Accuracy
SNN (Amazon)	0.66	0.65	0.64	0.64
CNN (Amazon))	0.65	0.64	0.64	0.64
SNN (MR)	0.82	0.82	0.82	0.82
CNN MR)	0.80	0.80	0.80	0.80

Table 4.8: Performance metrics for Amazon reviews and MR reviews using the pretrained neural models on the IMDb dataset.

4.2.7 Discussion

Based on our results, we could demonstrate the following points:

1. Summarizing the final opinion in some words might lead to the problem that the extracted features did not take that sentence as a feature of interest. For example, a reviewer might have a positive opinion about acting and the overall story of a movie, but he was not satisfied by the music in certain scenes.

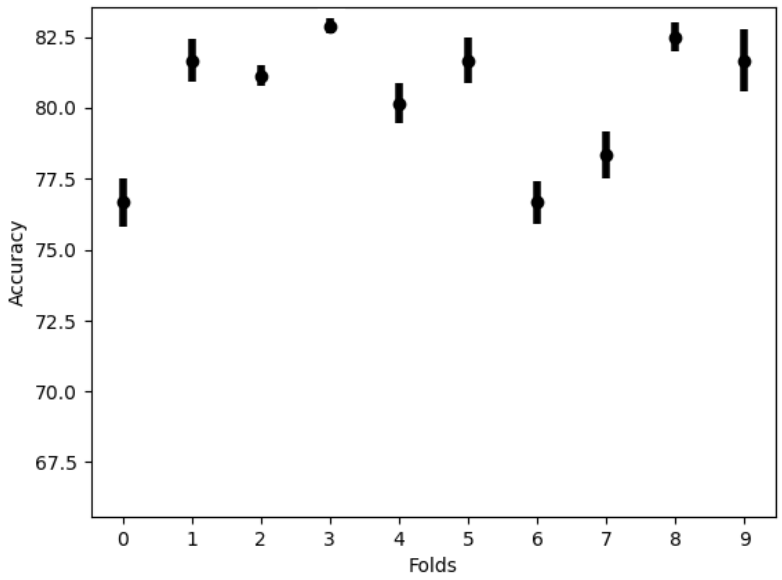

Figure 4.4: The cross-validation results for the trained CNN model using IMDb model, which has been used on MR and Amazon datasets.

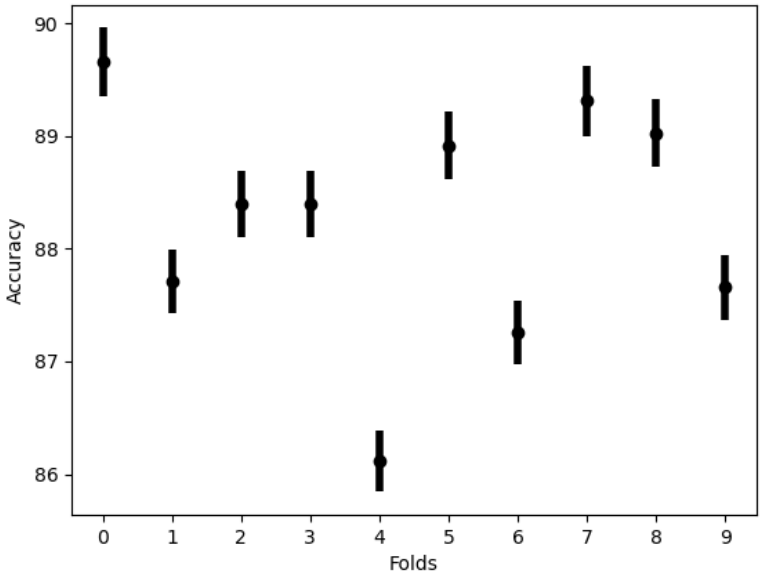

Figure 4.5: The 10-fold cross-validation results for the trained SNN model using the IMDb model, which has been used on the MR and Amazon datasets.

2. Sentiments can be expressed in different forms that might be even indirect expressions. Therefore, they require common-sense reasoning techniques to be classified. In addition, it is challenging to analyze sentiments with complex structures of sentences, especially when negations do exist.

3. Some reviewers may use expressions that have negative connotations, but at the end of the review, they summarize their overall opinion clearly. Consequently, this makes the classification a tough task.

4. Feature engineering suffers from overcoming the previous problems.

5. Sentiment analysis requires analyzing large units of individual words to capture the context in which those words appear.

6. Table 4.8 shows that SNN shows a better generalization performance using the MR dataset compared to CNN. The results of other approaches listed in Tables 4.9 and 4.10 for movie review data and IMDb, respectively. Some proposed approaches perform better; however, they do not

Paper	Classifier Used	Accuracy
[61]	CNN	0.81
[102]	Autoencoders	0.77
[123]	SVM	0.77
[57]	DCNN	0.86

Table 4.9: A summary of state-of-the-art performance metrics for MR, where DCNN: Dynamic Convolutional Neural Network, SVM: Support Vector Machine.

Paper	Classifier Used	Accuracy
[88]	SVM	0.82
[97]	Markov Model	0.80
[96]	Random Forest	0.88
[23]	statistical approach	0.87

Table 4.10: A summary of the state-of-the-art precision metrics for IMDb, where SVM: Support Vector Machine.

consider the domain-independent sentiment classification. It means that the results obtained for training and testing on the same dataset.

We can observe that this work is:

1. Able to classify binary reviews very well, especially, domain-independent reviews.

2. The first building block toward the generalization of sentiment classification, where a model can be trained once and tested on entirely new datasets that even may come from different contexts.

3. It inherits the advantages of neural models, which can nowadays classify hundreds of objects using a pre-trained model.

This is since CNN can overcome many challenges of sentiment analysis that have been highlighted previously. For example, words in a specific region are more likely to be related than words far away. Thus, CNN can automatically

and adaptively extract spatial hierarchies of features out of written reviews that may capture different writing styles of users.

4.2.8 Summary of the Approach

In this work, we proposed a sentiment analysis generalized approach that can classify the sentiments of different datasets robustly. Additionally, the proposed approach showed promising results in the context of domain-independent sentiment analysis. This is because neural models can extract robust features when reviews are converted to proper input vectors using our proposed review for vector algorithms. Furthermore, it shows a high performance regarding generalization. The proposed model has been trained once and tested on three different datasets from different domains. The model could perform very well compared to other works that used the same datasets and showed a generalization capability for sentiment classification w.r.t. in different domains. Furthermore, the paper covered a wide range of sentiment analysis approaches from the state of the art and compared the results obtained to the performance of the proposed neural models.

Additionally, in our future work, we will integrate the implemented version of the algorithm into different browsers and platforms, aiming at using the power of this approach for fact checking purposes.

"When you give false information you tend to restrict the freedom of choice to others." -Randal Marlin

CHAPTER 5

Fact-Checking as an Optimization Problem

5.1 Introduction

This chapter is based on the following paper *Fact-Checking Reasoning System for Fake Review Detection Using Answer Set Programming* [53]
Social media and e-commerce platforms have become a fundamental element of today's society [79], as a result, the data on the Web is hugely growing, but the quality of this data requires more investigation [55]. Social media platforms have become where people share the urge to stay connected and express themselves by discussing the news, giving opinions about politics, watching movies or products, but unluckily, this online data can be fast and quickly disseminated [52] and it is not so hard to manipulate with it. The fake news and misleading information became apparent nowadays, especially in moments of crisis, such as what we face today, the COVID-19 pandemic [17]. Different studies have proven that there is a need to flatten the curve of the rumors and misinformation about the virus with a view to flatten the curve of COVID-19 [2]. Besides, e-commerce websites are also an essential instance where numerous fake reviews and fake profiles exist due to the rising number of new products and the intense competition between companies [65].

Nevertheless, online reviews can be fraudulent and are yet necessary for both consumers and firms. For the consumers, it is necessary to check them to decide for the quality of the products and whether to buy it or not. In contrast, firms need online reviews to improve their products and boost the sales [54].

Thus, developing fact-checking tools that can detect fake reviews or fake users will help improve the clarity and quality of online data; otherwise, the Web will be a sea of abnormal distribution of rumors and false information.

Obviously, this is not an easy task: researchers in this domain invested a lot of effort, but it still needs to be improved. To understand the problem well, we should first recognize the characteristics of such data and the behavior patterns of the fake users.

In [64] and [29] sockpuppets are individuals with pretended profiles that incorporate wrong information and do not show their real identities; they have suspicious activities like spamming or flooding to mislead other users and hide the facts. Those activities can be exposed by their behavioral patterns and the other accounts who cooperate with them; for instance, by checking the IP addresses, it is possible to obtain information indicating spamming behaviors, mainly when the same user with the same IP address uses multiple fake accounts. Accordingly, identifying those accounts is highly important and removing them requires well-designed systems, which can identify such strange behaviors and malicious patterns.

This work is an extensive study of this kind of user's behavior that writes fake reviews and has malicious patterns to detect it. We define a knowledge base (KB) and a reasoning rules-based approach, using Answer Set Programming (ASP), where the following study [28] showcases the established usage of ASP in industry fields.

This approach is a universal approach because it includes previous mentioned approaches and integrates them in one model. It works as follows: the input is the movie reviews. Then, each review is checked using natural language processing to determine the sentiment to distinguish positive and negative criticism; next, a Levenshtein-distance algorithm [55] identifies deviations between user review data. Afterwards, the results of these algorithms and user data such as review rating and IP addresses data are grounded and

put it all in an ASP KB with predefined rules that fact check them. This way, the framework can identify reviews that are not trustworthy and label them accordingly in the output. The evaluation showed that this technique can detect fake reviews and works well with high flexibility since more algorithms can later be considered as extended rules can be for better performance.

5.2 Methodology

Why Reasoning System For Fake Review Detection is important?

According to the literature, 90 percent of customers who remembered reading online reviews claimed that positive online reviews impacted their buying considerations [36]. At the same time, 86 percent of them said negative online reviews affected their buying decisions. Additionally, around 40 percent of all online reviews on Amazon are fake [1]. Thus, the need to have a reasoning system that can detect such reviews is in high demand, specially when taking the following points in consideration:

1. Fake reviews deceive customers of being fully aware of all the information about the product they are buying.

2. Some fake reviews are written to distract customers from positive rating of the product.

3. Numerous fake reviews are written to promote or even to demote the product.

4. Pure supervised learning approaches could not be a proper choice due to the lack of training data that considers the behavioral patterns of users/reviewers. The provided approach uses supervised learning for sentiment analysis of the review, which is considered in one rule in the KB.

[1] https://www.brightlocal.com/research/local-consumer-review-survey/

On the Characteristics of Fake Reviewers and Reviews

Behind every fake review, there is a user that normally has a well-defined malicious behavior. As described in [118], certain patterns such as posting only positive or negative reviews, or posting reviews in a specific time interval, for example, every day or every week at the same period of time, can be detected as a spamming behavior. Another interesting pattern for a fake review can be the content itself. For example, there are many redundant reviews with only minor changes or spam reviews with a bad quality texting and few counts of words [81].

A further well-described pattern of malicious behavior described in both, [81] and [89], is the control of different accounts by a single spammer, who uses them for spamming and domination. These accounts are also known as bot accounts or sockpuppets and can be recognized by following their actions in connection with other similar users. Most likely, the grouping that will appear is going to be bot users.

An additional aspect for fake review recognition could be a single individual user coupling in a group, which describes the behavior of multiple users acting similarly relatively at the same time, which is often done by bot accounts [81]. Furthermore, accounts with some special "sequential" user naming are often associated with bot users.

Another way to detect fake reviews can be achieved by checking the dislike counts. Many e-commerce websites offer some functionality that allows people to vote if a review was helpful or not. Every so often the community is doing the job for us by disliking the suspicious reviews. This makes the dislike count be represented as a feature that needs to be taken into consideration.

The work in [29] discusses the anonymity factor that may need further consideration as well. For example, on Amazon, users can post a review under a nickname or as an anonymous user. Anonymous users post many fake reviews because the user may want to avoid getting tracked. In our proposed system, the requirement to post a new review is that the user initially registers thanks to the registration step; the anonymous review feature is not feasible.

Another significant factor in the recognition process is the review feature similarities of a new or updated review with all other reviews in the system. It is possible to compute these similarities by using dynamic algorithmic

approaches as discussed in [55]. Given the deviation values results of the analyzed reviews, the three reviews with the highest similarity scores are considered, and the number of high and low deviating reviews is used in the ASP knowledge base

5.3 Approach

ASP is a well-known technique in the domain of Knowledge Representation and Reasoning (KRR) [35]. It belongs to the group of so-called constraint programming languages. In ASP, the specification of the problem is given, and the specification model delivers the solution. It is also a form of declarative programming, which is mainly used to solve non-deterministic search problems. As described in [34] and [93] it got prevalent *"because of its appealing combination of a rich yet simple modeling language with high-performance solving potentials"*. In ASP, search problems are reduced to compute solid models and answer sets. It utilizes solvers that are reasoners, which outcome stable model generation [34].

Figure 5.1 presents the ASP solving process. The aim is to deliver one or several solutions to the given problem. The primary step is to model the problem into a logic program with rules, first-order predicates, and variables. Then, the grounder will reduce the variables by substituting them with ground terms in the language. As a consequence, a propositional program will only contain propositional atoms. The program is then taken by the solver, which will calculate all the stable models and interpret the solution [32]. Therefore, it is essential to understand the data and the problem itself to start building our KB.

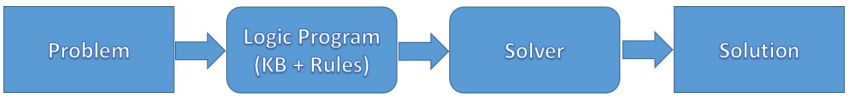

Figure 5.1: Answer Set Programming Solving Process

This allows for creating a formal model of the problem and the input data through a KB, constraints, and rules that have to be satisfied (see Sections

5.3.5 and 5.3.5). In the upcoming illustrating example, we show the approach considering only three movies just for simplicity. This simplification does not cause any limitation of the approach.

While developing the KB, with all the constraints and facts, we have used the online tool Potassco [2] to test the resulting labels and analyze the running time of the solver. Solving a KB with four reviews, including their facts, such as IP address, sentiment, star rating, and cosine similarity, took a total of 0.013s to solve, showcasing the efficiency of ASP solvers.

5.3.1 ASP Fundamentals

As previously mentioned, the KB consists of various facts and rules which are used to generate an answer set, which includes the possible solutions of the given problem. ASP programs are based on the AnsProlog (also called A-Prolog [8]) programming language and use the predefined rules and facts, instead of a classical algorithmic approach, to find a solution to a given problem. The following shows examples of defining a rule and constraint.

$$a_0 \leftarrow b_1, ..., b_n, \neg c_1, ..., \neg c_m \tag{5.1}$$

The example rule 5.1, shows a list of atoms in first order logic $b_1, ..., b_n, c_1, ..., c_m$, which is called the body of the rule, whereas a_0 is called the head of the rule. It is also important to note that rules without a body are called facts [3].

$$\leftarrow b_1, \neg c_1 \tag{5.2}$$

The example constraint 5.2, shows a list of atoms in first order logic $b_1, \neg c_1$, which is different from a rule in such a way that the head is empty. A constraint is used to remove some elements from the answer set, as in the given example constraint, the answer set will not include b_1 if c_1 is not generated.

5.3.2 Dataset

To achieve accurate results that mimic real-world scenarios, we implemented a website for evaluating movie reviews. It allowed us to create a more extensive

[2]The Potsdam Answer Set Solving Collection, https://potassco.org/clingo/run/

dataset that can help us explain the reviewers' behavior. We wanted to have complete control over the dataset so that we can get all the required attributes. In order to achieve this, the website uses a local database instance for storing all related data, including user data, review data, and movie-related data. The dataset can be used later to inspect the best or worst-rated movies, movies with many fake reviews, and similar measurements.

5.3.3 Analysis of structured and unstructured data

The approaches of analyzing fake data on the web can be categorized in two segments. The first one is based on structured data, where the content can be directly analyzed as such. As structured data is organized well, depending on the type of the data, it is much easier to generate a well-defined KB, which will yield clear answer sets, given the rules and constraints defined in the KB. On the other side, with unstructured data, the approach includes different steps of organizing the unstructured raw data, to result in processed data that can be used in the KB. These steps include behavioral based analysis steps, that can be used to analyze predefined classifications of behaviors or to generate new behavioral patterns from the given data.

Another way of interpreting the data is using natural language processing steps, to analyze, i.e. the sentiment of the data. The main idea behind these approaches is to transform the unstructured data into well-structured data, which can be used as a basis for the KB generation. These steps are crucial to understand, implement and evaluate in the early steps of generating the KB, meaning that the rules and constraints within the KB need to comply with the processed unstructured data, to establish a well-defined answer set. The final step of the process is to evaluate the inputs to the KB, and generate the answer set that recognizes the given data as, in our case, fake or genuine. These classifications can be extended and adapted to other requirements, giving the opportunity to evaluate different data inputs for other, possibly more domain-specific, datasets, based on an existing KB that can be easily extended to meet the new requirements. This is one of the aspects that illustrate the advantage of ASP approaches for problem-solving tasks.
Figure 5.2 shows the given steps of the analysis of structured and unstructured data, regarding the generation of a KB.

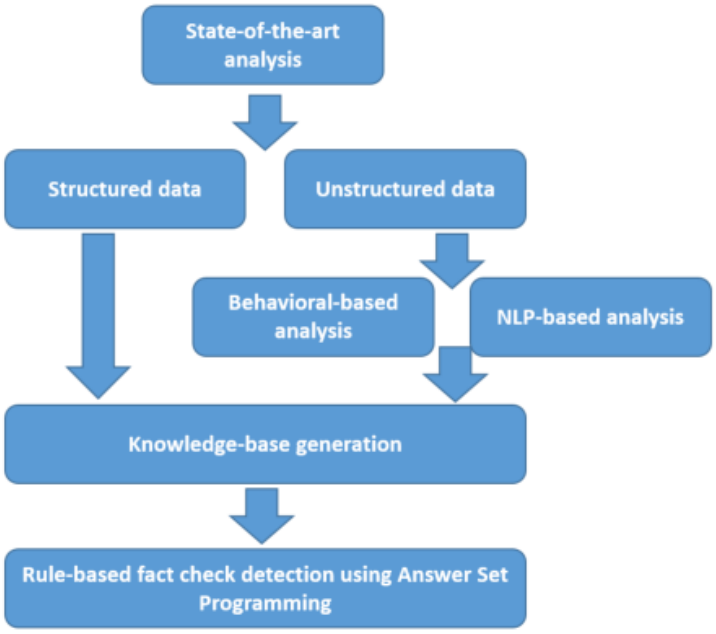

Figure 5.2: Process for structured and unstructured data

Levenshtein / Cosine Similarity

Presents a technique that helps in pinpointing contradicting and deviating pieces of information in structured data sources. It incorporates two algorithms, Levenshtein Distance (LD) and Cosine Similarity (CS). LD measures the similarity of two input strings [39]. It is also known as "edit distance", which helps in many applications like text analyzing and correcting errors [124]. In this case, the distance is the number of substitutions, deletions, and insertions required to create the target string from the source string. A high distance would indicate that the two inputs are notable and deviate from each other. Moreover, a low distance indicates higher similarity and that the strings do not deviate as much. According to [55], obtaining these results involves three succeeding steps: The first step is to edit input data to compare them. This step includes extracting unnecessary characters from strings with regular expressions and parsing dates to a standard format. The second step is to calculate pairwise deviations between the reference object and each item in a dataset, where every reference object or a single object in the compared

dataset consists of a movie title, a movie date, and the movie cast names. After normalizing the computed values, the distance matrix is then calculated. The next step of the algorithm is to specify further the Levenshtein distance analysis matrix, which was created in the second part of the algorithm and comprises ordering the results in descending or ascending order, which is done using what so-called sensitivity vectors. It allows investigating which sort of deviation to examine out of all movie features. For example, a sensitivity vector of (1, 1, 1) would look for significantly deviating data and creates a list as output, which shows movies that have a high deviation across all categories. In contrast, a vector (0, 0, 0), on the other hand, does the opposite: It orders the created list by non or most minor deviating data, which results in a list that holds low deviating movies in the front. The chosen sensitivity vector contributes to a cosine similarity function with the deviation matrix created in step 2. The output is a list that orders the deviating data by the preferences earlier defined in the sensitivity vector and later used to analyze the algorithms' calculations by initially ordering the feature list received by the function.

This algorithm is a robust tool in identifying deviating data in structured data elements. The most significant part of this algorithm is that it does not require a training phase like machine learning approaches. Another advantage is that there is no need to use sample data to determine a specific models' parameters [55]. Consolidating this algorithm into a given project is very straightforward because, as discussed earlier, it does not need sample data or training phase.

5.3.4 Demo Application Scenario

In the demo application, it is assumed that there are movies, each of them having assigned some reviews written by some users. Each review has assigned a number of (dis)likes reflecting the opinion of other users, number of stars reflecting the author opinion of the movie, a sentiment score of the review (ranging from 1 as strongly negative to 5 as strongly positive), a timestamp encoding the time the review was posted, and an IP address encoding the users' machine as the source of the review.

All information about reviews and authors, such as (dis)likes, star ratings,

sentiment scores, timestamps, and IP addresses are encoded in a KB which will be discussed in Section 5.3.5.

In the application scenario, it is assumed that reviews may be considered *fake*, *possiblyFake*, *possiblyGenuine*, *genuine* or *contradicted*. Many rules are included and will lead to detect the correct classification of the reviews, helping the users to decide whether a review on a certain movie can be trusted or not.

All these rules determine the results by labeling a review as *fake*, *possiblyFake*, *possiblyGenuine*, *genuine* or *contradicted*, determining which reviews where posted from the same IP address, and labeling reviews as candidates for spam in case they have been written within a short time interval and stemming from the same IP address (e.g. a pattern often observed in the case of sockpuppetry [64]), determining the polarity scores of the reviews between the review star rating and the review sentiment analysis score, finally analyzing the review similarity with all other reviews in the system.

Section 5.3.5 will illustrate some sample rules.

5.3.5 Knowledge Base

The system uses two knowledge bases, one for the review classification and another for the author classification.

Review Knowledge Base

In the review KB (see Listing 5.1), all constants should be declared first. The only constant value in the KB is 30 seconds, refers to the time interval between the posting time of two reviews.

Next we declare some timestamps t_i and IP addresses ip_i, followed by the declaration of movies m_i and reviews r_i. Subsequently, we declare which reviews are assigned to a movie (see *hasRev*).

Review features are being set, such as the review stars (see *stars*), sentiment score (see *sentScore*) and review Levenshtein/Cosine similarity scores (see *revLCS*).

Then the timestamps and the source IP addresses for all reviews are modeled (see *timestamp_of_Review* and *hasIp*). Finally, we declare the labels used for the classification of reviews.

```
1
2     #const time_interval = 30.
3     #const t1 = 2.
4     #const t2 = 4.
5     #const t3 = 6.
6     #const t4 = 7.
7     #const t5 = 8.
8     #const t6 = 13.
9     ...
10    #const ip1 = 858513516.
11    #const ip2 = 199188126120.
12    #const ip3 = 960233229.
13    ...
14
15    movie(m1;m2;m3; ...;mn).
16    review(r1;r2;r3;...;rn).
17    hasRev(m1,(r1;r2;r3)).
18    hasRev(m2,(r4;r5;r6)).
19    ...
20
21    stars(r1, 5).   stars(r2, 4).
22    stars(r3, 1).   stars(r4, 1).
23    ...
24    sentScore(r1, 4). sentScore(r2, 4).
25    sentScore(r3, 5). sentScore(r4, 2).
26    ...
27    revLCS(r1, 0, 3). revLCS(r2, 0, 3).
28    revLCS(r3, 3, 0).
29    revLCS(r4, 2, 1).
30    ...
31
32    timestamp_of_Review(r1, t1).
33    timestamp_of_Review(r2, t2).
34    timestamp_of_Review(r3 ,t3).
35    timestamp_of_Review(r4, t4). ...
36
37    hasIp(r1, 1010). hasIp(r2, 2020).
38    hasIp(r3, 1010). hasIp(r4, 2220). ...
39
40    reviewLabel(fakeReview). reviewLabel(
      possiblyFakeReview).
41    reviewLabel(possiblyGenuineReview).
42    reviewLabel(genuineReview).
43    reviewLabel(contradictedReview).
44
45
```

Listing 5.1: Abstracted Samples of Facts from the Review KB

Review Constraints and Rules

Based on the facts from the KB, constraints and rules satisfying conditions on the facts are declared such that they formally encode the problem that needs to be solved. In our scenario, the problem we want to solve can be stated like this: *Given some facts about reviews on movies, we want to detect fake and genuine reviews and identify candidates for spam reviews.*

In our case, a set of illustrating constraints and rules (see Listing 5.2) has been declared, which state the following:

1. Each review can be assigned as highly, moderately, or normal regarding the review polarity. A review will be of *highPolarityDiff* if the difference between the sentiment score and the star rating of the review is greater or equal to 3.

2. Analogously to the previous rule, *moderatePolarityDiff* checks if the difference between the sentiment score and the star rating of the review is greater or equal to 1 and is not believed to be of *highPolarityDiff*.

3. Analogously to the previous two rules, *normalPolarityDiff* checks if the difference between the sentiment score and the star rating of the review is not believed to be of *highPolarityDiff* and *moderatePolarityDiff*.

4. A review is considered as a unique review (*spamByLCS*) if the Levenshtein/Cosine similarity scores are higher in the number of lower deviating reviews than high deviating reviews.

5. The given rule states that if any two different reviews of a certain movie have an identical source, IP address, and both reviews were written within a short time interval (e.g., 30 seconds, i.e., $|z_2 - z_1| < 30 seconds$) it has to be derived and concluded that both reviews are spam candidates.

6. The given rule checks if the review can be concluded to be a genuine review if the review is not a spam candidate by the *spamByCS* rule, not spam by timestamp, and neither high nor moderate polar. (see *genuineReview*).

7. The given rule checks if the review can be concluded to be a possibly genuine review if all previous rules match. However, the review is considered to be of moderate polarity. (see *possiblyGenuineReview*).

8. The given rule checks if the review can be concluded to be a possibly fake review if the review is considered to be spam by the Levenshtein Cosine similarity measure, spam by timestamp, and a moderately polar review. (see *possiblyFakeReview*).

9. The given rule checks if the review can be concluded to be a fake review if the review is considered to be spam by the Levenshtein Cosine similarity measure, spam by timestamp, and a highly polar review. (see *fakeReview*).

10. If the review is not believed to be of the previous classifications, it is considered to be a contradicted review. (see *contradictedReview*).

```
1
2     #highPolarityDiff(X) :- review(X), stars(X, S1),
3           sentScore(X, S2), |S1 - S2| >= 3.
4
5     #moderatePolarityDiff(X) :- review(X), stars(X, S1),
6           sentScore(X, S2), S1 - S2| > 1,
7           not highPolarityDiff(X).
8
9     #normalPolarity(X) :- review(X),
10           not highPolarityDiff(X)
11           not moderatePolarityDiff(X).
12
13    #SpamByLCS(X) :- review(X), revCS(X,Y,K), Y > K.
14
15    #SpamByTimestamp(X;Y) :- hasRev(M1, X),
16           hasRev(M2, Y), M1=M2, X!=Y,
17           hasIp(X,I1), hasIp(Y, I2), review(X),
18           timestamp_of_Review(X, Z1), review(Y),
19           timestamo_of_Review(Y, Z2),
20           |Z2 - Z1| < time_interval.
21
22    #genuineRev(X) :- review(X), not spamByLCS(X),
23            not spamByTimestamp(X),
24            not highPolarityDiff(X),
25            not moderatePolarityDiff(X)
26
27    #possiblyGenuineRev(X) :- review(X),
28            not spamByCS(X), spamByTimestamp(X),
29            not highPolarityDiff(X).
30
31    #PossiblyFakeRev(X) :- review(X),
32            spamByLCS(X), spamByTimestamp(X),
33            moderatePolarityDiff(X).
34
35    #fakeRev(X) :- review(X), spamByLCS(X),
      spamByTimestamp(X),
36            highPolarityDiff(X)
37
38    #contradictedRev(X) :- review(x),
39            not fakeRev(X),
40            not possiblyFakeRev(X),
41            not genuineRev(X),
42            not possiblyGenuineRev(X).
43
44
45
```

Listing 5.2: Sample review constraints and rules

Author Knowledge Base

In the author KB (see Listing 5.3), the main focus lies on the number of reviews posted by the author for a given movie and how these reviews are rated by other users of the system.

The facts include the author's object, movies and reviews, the number of likes and dislikes of a review, and what review is assigned to which movie.

```
1
2     author(a).
3     movie(m1).
4     movie(m2).
5     movie(m3).
6     movie(m4).
7
8     review(r1).
9     review(r2).
10    review(r3).
11    review(r4).
12    review(r5).
13    review(r6).
14
15    reviewLikes(r1, 10).
16    reviewLikes(r2, 0).
17    reviewLikes(r3, 16).
18    reviewLikes(r4, 79).
19    reviewLikes(r5, 2).
20    reviewLikes(r6, 30).
21
22    reviewDislikes(r1, 2).
23    reviewDislikes(r2, 15).
24    reviewDislikes(r3, 3).
25    reviewDislikes(r4, 5).
26    reviewDislikes(r5, 2).
27    reviewDislikes(r6, 7).
28
29    forMovie(r1, m1).
30    forMovie(r2, m1).
31    forMovie(r3, m1).
32    forMovie(r4, m2).
33    forMovie(r5, m3).
34    forMovie(r6, m4).
35
```

Listing 5.3: Abstracted samples of facts from the author KB

Author Constraints and Rules

Based on the facts from the KB, constraints and rules satisfying conditions on the facts are declared such that they formally encode the problem that needs to be solved. In our scenario, the problem we want to solve can be declared like this: *Given some facts about reviews on movies, we want to detect fake and genuine reviews and identify candidates for spam reviews.*

In our case, a set of illustrating constraints and rules (see Listing 5.4) has been declared, which state the following:

1. Using the *sum* aggregate, the number of likes overall reviews is being calculated.

2. Using the *sum* aggregate, the number of dislikes overall reviews are being calculated.

3. The *highDiff* rule checks how many reviews have a higher number of likes compared to the number of dislikes, and returns the positive number of likes of a review. For example, if review *r1* has 15 likes and 5 dislikes, the rule will return *highDiff(r1,10)* as an answer set.

4. To get the best-rated review, the rule *bestRev*, uses the *max* aggregate and the *highDiff* rule, calculates which review has the highest number of likes compared to the number of dislikes, over all reviews.

5. Analogously to the *highDiff* rule *lowDiff*, this rule calculates the reviews that have a higher dislike count than like count.For example, if review *r1* has 3 likes and 20 dislikes, the answer set of this rule will be *lowDiff(r1,-17)*.

6. Analogously to the *bestRev* rule , *worstRev* uses the *min* aggregate and the *lowDiff* rule to calculate which review has the highest number of dislikes compared to the number of likes, over all reviews.

7. The *highCount* rule gets the number of liked reviews.

8. The *lowCount* rule gets the number of disliked reviews.

9. Using the *liked* rule, it returns if the author has more liked than disliked reviews.

10. Using the *neutral* rule, it returns if the author has the same number of liked and disliked reviews.

11. Using the *disliked* rule, the author is classified as disliked if the author is not believed to be liked or neutral.

```
1
2     likes(N) :- N = #sum{S, R : reviewLikes(R, S)}.
3
4     dislikes(N) : N = #sum{S, R : reviewDislikes(R, S)}
5
6     highDiff(R, X) :- reviewLikes(R, L),
7         reviewDislikes(R1, D), R = R1, L > D, X == L -
    D.
8
9     bestRev(R, X) :- review(R), highDiff(R, X),
10        #max {XX, 1: highDiff(Z, XX)} = X.
11
12    lowDiff(R, X) :- reviewLikes(R, L), reviewDislikes(
    R1, D),
13        R = R1, L < D, X == L - D.
14
15    WorstRev(R, X) :- review(R), lowDiff(R, X),
16        #min {XX, 1: lowDiff(Z, XX)} = X.
17
18    highConts(S) :- S = #count{R : highDiff(R, X)}.
19
20    lowConts(S) :- S = #count{R :  lowhDiff(R, X)}.
21
22    liked(X) :- highCount(S), lowCount(S2),
23        S > S2, person(X).
24
25    neutral(X) :- highCount(S), lowCount(S2),
26        S == S2, person(X).
27
28    dislike(X) :- not liked(X), not neutral(X),
29        person(X).
30
31
```

Listing 5.4: Sample author constraints and rules

5.3.6 Computed Results

The results computed by an ASP solver always satisfy all the constraints and rules. The computed results are considered as an optimal solution with the highest accuracy and precision compared to other work we previously

mentioned in Section 2, where most of them apply machine learning techniques and natural language processing for fake reviews detection.

The set of constraints and rules can be easily extended to cover much more complicated scenarios built upon a variety of behavior patterns, and not just a few characteristics the spammers or the spams may have, [108] which are well understood and studied in the literature.

5.4 The Demo Application

The major components of the proposed system have been implemented as a website. The system allows users to register for an account to be able to write reviews for movies obtained from the IMDB API[3]. When the user is logged in, it is possible to add reviews, such as review title, review content, and review star rating, insert reviews, delete their reviews, update their reviews, and down vote other reviews. All the inserted data will be investigated by the proposed rule-based reasoning approach to detect the review classification.

We strongly believe that such a tool is one of the first rule-based website systems that offer the possibility to collect datasets, annotate reviews, and use ASP as a powerful reasoning paradigm.

5.5 Results

Regarding the sentiment analysis of the review content, a trained logistic regression model is used as an adapted version of the Stanford Treebank dataset[4]. The adapted version uses the review content from the dataset. The dataset is preprocessed, and the sentiment score in the training and test datasets vary from 1 to 5, strongly negative to strongly positive, respectively. The trained sentiment analysis model, which was used as a tool in the review analysis system, has accomplished an accuracy of 41.22%, which is closely related to the accuracy of 40.7%, [103] the authors of the used dataset have achieved that. In the analysis of the model, many of the neutral sentiment reviews were assigned either a weakly positive or weakly negative sentiment, as the trained model was kept very simplistic for this approach. In the future,

[3]https://developer.imdb.com/

[4]https://nlp.stanford.edu/sentiment/treebank.html

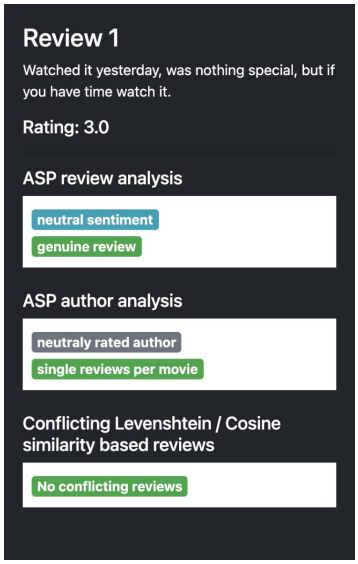

Figure 5.3: Neutral review example

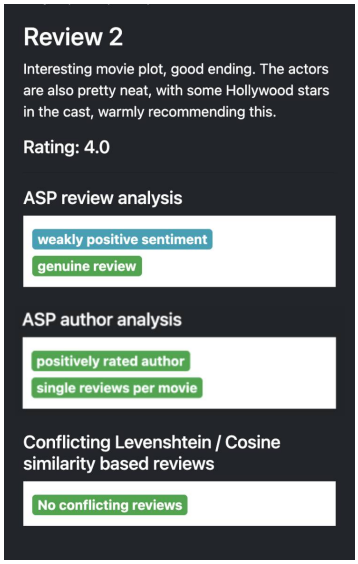

Figure 5.4: Genuine review example

the aim is to use a more sophisticated model for sentiment analysis to enhance the accuracy of the whole system, such as in [52], where Convolutional Neural Network (CNN) and Shallow Neural Network (SNN) reached 81.0% and 87.0% accuracies respectively, using the IMDB dataset.

For this paper, a simple trained model was sufficient, as the main idea was to show how ASP can be used with different tools to classify movie reviews considering the review content and its sentiment.

The following figures show examples of classified reviews on the website. They show the functionality of how a review is classified based on the review itself and the author, additionally including any reviews that have been flagged as highly similar with the current review by the Levenshtein/Cosine Similarity value.

Figure 5.3 shows a genuine review with a neutral sentiment, where the author has no ratings yet.

In contrast, Figure 5.4 shows a weakly positive genuine review where the author has reviews that other users have liked.

Figure 5.5 shows a spam review with a strongly negative sentiment, that has been posted from the same IP address.

Figure 5.6 shows a contradicted review, as it has a positive sentiment but a

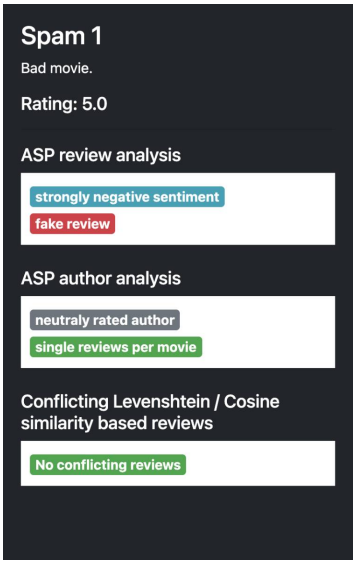

Figure 5.5: Spam review example

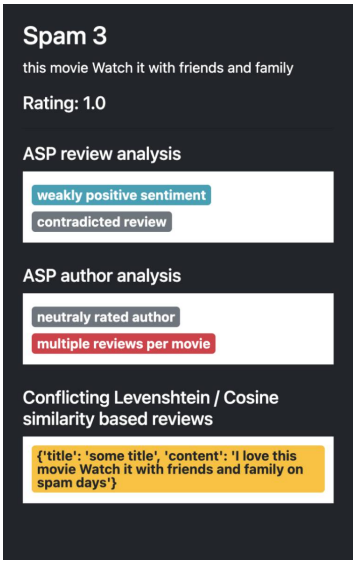

Figure 5.6: Contradicted review

low star rating, additionally it is very similar to another review in the system, which has been flagged accordingly.

It is also important to note that the review shown in Figure 5.6 has a label showing that the author posts multiple reviews for a single movie, which is considered to be negative.

5.6 Summary of the Approach

The problem of fake review detection is complicated, since fake reviews are linked to at least two sources: the review content and the reviewers' behavior. Commonly, the content is analyzed and classified by adopting natural language processing techniques. However, determining the behavior of a fake reviewer is a complex problem. Various spatial-temporal patterns of behavior can be modeled as part of an optimization problem and because of the complexity, these problems are time-consuming and required a lot of computational resources to be solved. ASP is an exciting and powerful tool to address such problems, mainly offering a declarative way to specify the problem and properties of the desired solution(s). ASP inherits the advantage of rule-based approaches, (a) flexibility which means that adding new rules or updating the system requires only a few changes compared to other paradigms, (b) it

does not require a vast amount of training data, (c) rule-based techniques are considered a white box; thus, it provides traceability and transparency for critical decisions that demand a higher degree of explanation usually provided by machine learning approaches, (d) same rules can be used for another data type like tweets or Facebook posts for fact-checking purpose, and (e) clear and understandable specification for users. Regarding threshold values used in ASP approaches, they can be calculated regularly and automatically based on newly inserted data. Additionally, adding new threshold values can help in restricting and limiting the spamming behavior.

Table 5.1 shows a list of different reviews, representing examples of labeled reviews by the ASP solver, using the proposed review-centric KB. The table consists of the review content in the first column, following the review rating, sentiment score using the trained model, the Levenshtein/Cosine review similarity score, the triggered rules, and the actual label has given the review. The last two reviews show an example of two reviews posted shortly after each other, thus resulting in the second review being labeled as a contradicted review, according to the given input parameters for the ASP solver.

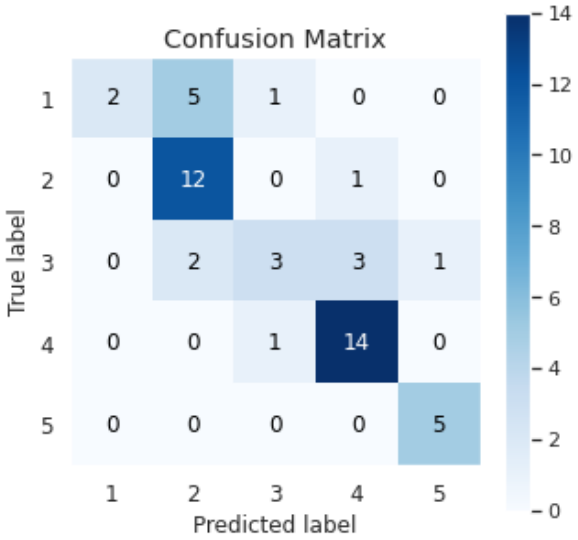

Figure 5.7: Confusion Matrix of 50 movie reviews for sentiment analysis

Figure 5.7 shows a confusion matrix of 50 movie reviews, obtained from the Rotten Tomatoes dataset, including their sentiment score from 1 to 5,

Review List					
Review-content	Review rating	Review Sentiment	Similar-reviews (LCS)	Triggered rules	Label
"bad movie"	5	1	3	highPolarityDiff, spamByLCS	fake review
"Watched it yesterday, pretty good movie, would recommend to watch!"	4	4	0	normalPolarity	genuine review
"Would say it was worth watching it, but in the end the story was too vague, and we were not happy with the ending."	2	4	0	moderate-PolarityDiff	possibly genuine review
"Great movie and awesome actors, one of our favourites!"	2	5	0	highPolarityDiff	contradicated review
"Would say it was worth watching it, but eventually the story was too vague, and we were not happy with the ending."	2	4	0	moderate-PolarityDiff	possibly genuine review
"This movie is just not for me." *(Posted shortly after the previous review from the same IP address)*	2	2	2	moderate-PolarityDiff, spamByTimestamp, spamByLCS	possibly fake review

Table 5.1: Example reviews alongside their triggered ASP rules.

that were processed by the website to analyze the accuracy of the sentiment analysis. The matrix shows that the prediction of the sentiment analysis is correct for most of the reviews, while only few reviews had a difference higher than 1, for example the one review that had a true sentiment score of 1 and the system predicted a score of 3. The highest accuracy was for reviews with the sentiment score of 4.

This approach proposes a dynamic recognition system for detecting fake reviews, spam, and spammers on the Web introducing fake reviews. After looking in-depth at the behavioral patterns of users known from the state-of-the-art literature, a set of illustrating constraints and rules has been designed. The problem could be transformed into an optimization problem that can be solved using ASP. As proven in [27], ASP is already established in the industrial segment as a state-of-the-art tool for different approaches such as search or scheduling problems and usages for engineering tasks and optimizing systems, such as caching strategies for optimizing performance in network systems.

Different components used in the reasoning system, which process movie reviews and user data, were implemented and combined under a fact-checking framework. Such a system is necessary because the problem of opinion spam has increased dramatically in past years. The output of the implemented system and other user and review data is fact-checked with declarative rules coded in ASP. The constraints and behavioral rules can be altered, edited, and deleted easily due to the flexibility of this system. Behavioral rules are used on user data to examine the trustworthiness of said reviews and users according to their overall behavior. Sentiments of user reviews are determined, which analyses the raw review content. Deviations in movie features are detected with an algorithm based on Levenshtein Distance that calculates a deviation score and then provides a list that is sorted by high or low deviation.

Such rule-based systems and the KB are in high demand to be well-designed. It allows for achieving a higher degree of accuracy and a properly working model. A proper design also leads to an extensible model, which will enable us to process much more complex scenarios at the same level of high accuracy. One of our next steps is to consider the social and personal features of the users; the reason is, based on their activities and profiles, it is likely

that fake users can be identified [9].

This work shows the high capability of ASP to solve such problems, and we believe that much scientific work can be done by further developing our proposed approach for fake review detection, a simple yet powerful and highly flexible tool for detecting opinion spam in movie reviews, although we believe that the same approach can work for different domains. Moreover, many more behavioral rules can be added to point out suspicious users or spamming patterns; it might require additional input data to work correctly.

"Unfortunately, fact-checking has become a lost art"
-Gary Hopkins

CHAPTER 6

Conclusion

6.1 Summary and Discussion

The increasing amount of data on the Web and the rage of social media made it easy for misleading, contradicted, or false data to disseminate fast and has proved to affect our societies negatively. While fact-checking communities grow up to battle this negative impact, which can be shown clearly after the COVID-19 pandemic, the continuous portion of inconsistent data is often complex to control. This thesis focuses on structured and unstructured textual data and answers essential research questions through proposed algorithms. Moreover, the developed algorithms can detect conflicting/deviating data. They can also spot and recognize suspicious behavioral patterns behind the spread of misinformation. Furthermore, considering the sentiment of end-user feedback and its essential role in detecting such inconsistent/conflicting data was a profound reason to develop a generalized sentiment analysis approach integrated into the fact-checking system. Using logic programming proposes a dynamic recognition system for detecting fake/misleading content on the Web and spotting the people behind those content by analyzing their content and studying their behavioral patterns after looking in-depth at the behavioral patterns of users known from the state-of-the-art literature. All the developed approaches have been integrated into one reasoning system. Utilizing the power of answer set programming and its flexibility helps create

a reasoning system that applies all the algorithms in a universal approach, as demonstrated in Chapter 4. According to the final results, it is certain to state that there is still ample space to extend the work. For example, by adding more algorithms that serve the aim of the fact-checking and consider the topic-independent characteristic, which means applying the approaches not just in a specific domain but also on different data from various domains.

The significant contributions of the thesis (C1, C2, C3, and C4) are shown in Figure6.1. It gives an overview of this thesis and summarizes the answers for all research questions (RQ1, RQ2, RQ3, and RQ4):

C1: Intensive state-of-the-art analysis has been performed to provide better solutions and better problem understanding related to fact-checking. Profound analysis has been made on fact-checking issues regarding textual data. Limitations and weaknesses of the existing approaches have been studied and understood in order to answer the following research questions.

C2: A flexible algorithmic approach [55] that allows the comparison between different data records and investigates them with considering the rate of deviations in their attribute values based on Levenstein distance. In the proposed approach, neither training step nor probabilistic or statistical models are needed.

C3-(a): A declarative programming algorithmic approach [54] using Answer Set Programming (ASP) has been developed to spot suspicious behavioral patterns of the people spreading fake data and reasoning over the data content. Such models encode the problem phrased "which piece of text is to be considered genuine, fake, or need to be investigated further on" and can be used to compute an optimal solution by applying ASP techniques.

C3-(b): A generalized sentiment analysis approach based on natural language processing techniques, e.g., bag of words, word to vectors, and neural models have been developed to classify the sentiment of a specific text (e.g., movie reviews). This approach has been developed to catch and categorize sentiments to help users from being easily misled.

C4: A generic approach to handle fact-checking problems using logic pro-

gramming or constraint programming (ASP) has been developed. The approach proposes a proper knowledge representation of facts on the Web, a model for precision metrics, which will allow for the definition of a robust optimization problem which minimizes and maximizes meaningful measurements considering different measurements of similarities and dissimilarities.

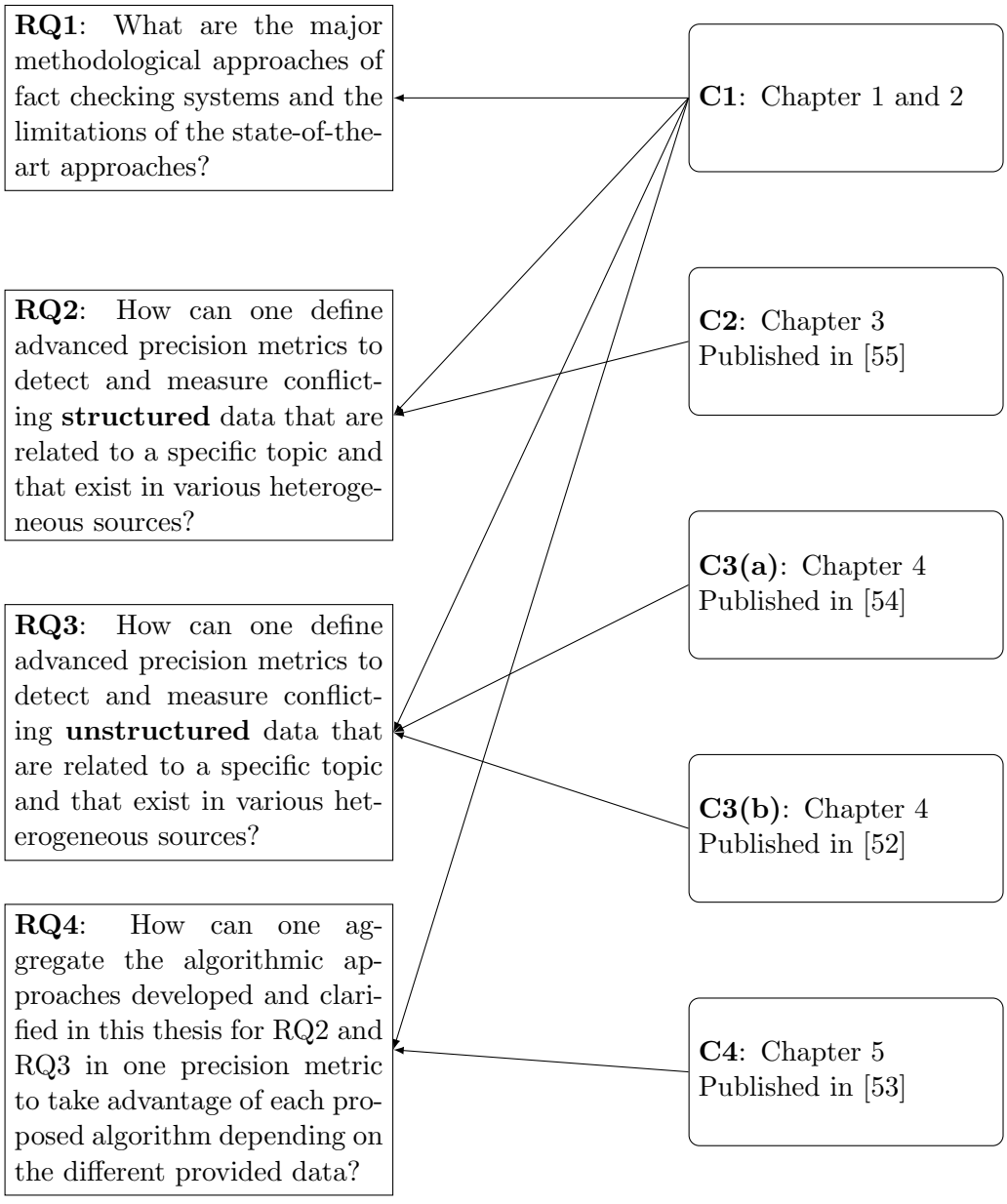

Figure 6.1: Research overview

6.2 Limitation of the work

All developed approaches presented in this thesis handle only textual data and do not handle images, audio, and videos. All the datasets that have been tested are from the movie domain, such as movie description records and movie reviews. Thus, testing datasets from different disciplines will be necessary. The rules that address the users' behavioral patterns are limited, and extensive research from different disciplines may help to extend the rules for more accurate results.

6.3 Future Work

Automated fact-checking approaches aim to understand and categorize user-generated content for any given domain. It is a challenging and ample task, especially when considering the variety of data users publish online, like texts, images, audio, and videos.

We believe that the current universal approach offers the possibility of integrating more algorithms in the current framework, yielding different, more precise results which can handle both textual data and other data types.

Moreover, the results rely on the behavioral patterns of users, where the primary challenge is to detect unanticipated/suspicious behaviors or outliers. Hence, we believe that there is a possibility of adding more rules that yield to spot and categorize the general user population's patterns in a domain; achieving this can be achieved by doing intensive study in the area of online user behavior analysis.

When designing a fact-checking approach, every measuring step one can take depends highly on the domain; for example, the evaluation measures for deviated data differ strictly in the medical scenarios from all other domains and need to be handled differently. In contrast, checking the deviation in the movie dataset records does not require a strict deviation measurement, such as in medical scenarios.

We suppose that more different features can be considered in the given text, such as selecting a lexicon and calculating the influence of each word or sentence that corresponded to all other words and sentences in the text. Integrating diverse methodologies such as machine learning methods, answer

set programming, similarity measures in one reasoning system that can be easily adapted and extended looks very promising and opens the path for further advancements. Thus, a dynamically adapting system would result in the most reasonable accuracy.

List of Figures

List of Tables

Bibliography

[1] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018.

[2] Md Shad Akhtar and Tanmoy Chakraborty. Overview of constraint 2021 shared tasks: Detecting english covid-19 fake news and hindi hostile posts. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation: First International Workshop, CONSTRAINT 2021, Collocated with AAAI 2021, Virtual Event, February 8, 2021, Revised Selected Papers*, page 42. Springer Nature, 2021.

[3] Fadi Al Machot, Heinrich Mayr, and Suneth Ranasinghe. *A Hybrid Reasoning Approach for Activity Recognition Based on Answer Set Programming and Dempster–Shafer Theory*, pages 303–318. 01 2018.

[4] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, 2003.

[6] Samir Bajaj. The pope has a new baby! fake news detection using deep learning, 2017.

[7] Rajiv Bajpai, Devamanyu Hazarika, Kunal Singh, Sruthi Gorantla, Erik Cambria, and Roger Zimmerman. Aspect-sentiment embeddings for company profiling and employee opinion mining. *arXiv preprint arXiv:1902.08342*, 2019.

[8] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[9] Rodrigo Barbado, Oscar Araque, and Carlos A Iglesias. A framework for fake review detection in online consumer electronics retailers. *Information Processing & Management*, 56(4):1234–1244, 2019.

[10] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Probabilistic models to reconcile complex data from inaccurate data sources. In *International Conference on Advanced Information Systems Engineering*, pages 83–97. Springer, 2010.

[11] Jens Bleiholder and Felix Naumann. *Conflict handling strategies in an integrated information system*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik, 2006.

[12] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Computing Surveys (CSUR)*, 41(1):1, 2009.

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] Alison M Buttenheim, Karthik Sethuraman, Saad B Omer, Alexandra L Hanlon, Michael Z Levy, and Daniel Salmon. Mmr vaccination status of children exempted from school-entry immunization mandates. *Vaccine*, 33(46):6250–6256, 2015.

[15] Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco. *A practical guide to sentiment analysis*. Springer, 2017.

[16] Carlos Carvalho, Nicholas Klagge, and Emanuel Moench. The persistent effects of a false news shock. *Journal of Empirical Finance*, 18(4):597–615, 2011.

[17] Wilson Ceron, Mathias-Felipe de Lima-Santos, and Marcos G Quiles. Fake news agenda in the era of covid-19: Identifying trends through fact-checking content. *Online Social Networks and Media*, 21:100116, 2021.

[18] Neha S Chowdhary and Anala A Pandit. Fake review detection using classification. *International Journal of Computer Applications*, 180(50):16–21, 2018.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.

[21] Rupesh Kumar Dewang and Anil Kumar Singh. State-of-art approaches for review spammer detection: a survey. *Journal of Intelligent Information Systems*, 50(2):231–264, 2018.

[22] Komal Dhingra and Sumit Kr Yadav. Spam analysis of big reviews dataset using fuzzy ranking evaluation algorithm and hadoop. *International Journal of Machine Learning and Cybernetics*, pages 1–20, 2017.

[23] Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336, 2015.

[24] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[25] Esra Erdem, Michael Gelfond, and Nicola Leone. Applications of answer set programming. *AI Magazine*, 37(3):53–68, 2016.

[26] Wael Etaiwi and Ghazi Naymat. The impact of applying different preprocessing steps on review spam detection. *Procedia computer science*, 113:273–279, 2017.

[27] Andreas Falkner, Gerhard Friedrich, Konstantin Schekotihin, Richard Taupe, and Erich Teppan. Industrial applications of answer set programming. *KI - Künstliche Intelligenz*, 32:1–12, 06 2018.

[28] Andreas Falkner, Gerhard Friedrich, Konstantin Schekotihin, Richard Taupe, and Erich C Teppan. Industrial applications of answer set programming. *KI-Künstliche Intelligenz*, 32(2-3):165–176, 2018.

[29] Tommaso Fornaciari and Massimo Poesio. Identifying fake amazon reviews as learning from crowds. 2014.

[30] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[31] D. Fuscà, Stefano Germano, J. Zangari, Francesco Calimeri, and Simona Perri. Answer set programming and declarative problem solving in game ais. *CEUR Workshop Proceedings*, 1107:81–88, 01 2013.

[32] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The potsdam answer set solving collection. *Ai Communications*, 24(2):107–124, 2011.

[33] Martin Gebser, Torsten Schaub, Sven Thiele, and Philippe Veber. Detecting inconsistencies in large biological networks with answer set programming, 2010.

[34] Martin Gebser, Torsten Schaub, Sven Thiele, and Philippe Veber. Detecting inconsistencies in large biological networks with answer set programming. *Theory and Practice of Logic Programming*, 11(2-3):323–360, 2011.

[35] Michael Gelfond and Yulia Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[36] Eleanor Vaida Gerhards. Your store is gross-how recent cases, the ftc, and state consumer protection laws can impact a franchise system's response to negative, defamatory, or fake online reviews. *Franchise LJ*, 34:503, 2014.

[37] Sherry Girgis, Eslam Amer, and Mahmoud Gadallah. Deep learning algorithms for detecting fake news in online text. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 93–97. IEEE, 2018.

[38] Zhen Hai, Gao Cong, Kuiyu Chang, Wenting Liu, and Peng Cheng. Coarse-to-fine review selection via supervised joint aspect and sentiment model. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 617–626. ACM, 2014.

[39] Rishin Haldar and Debajyoti Mukhopadhyay. Levenshtein distance technique in dictionary lookup methods: An improved approach. *arXiv preprint arXiv:1101.1232*.

[40] Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. The quest to automate fact-checking. In *Proceedings of the 2015 Computation+ Journalism Symposium*, 2015.

[41] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 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*, pages 1803–1812, 2017.

[42] Atefeh Heydari, Mohammadali Tavakoli, and Naomie Salim. Detection of fake opinions using time series. *Expert Systems with Applications*, 58:83–92, 2016.

[43] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[44] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[45] Doaa Mohey El-Din Mohamed Hussein. A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, 30(4):330–338, 2018.

[46] David Isaac and Christopher Lynnes. Automated data quality assessment in the intelligent archive. *White Paper prepared for the Intelligent Data Understanding program*, 17, 2003.

[47] Hisao Ishibuchi, Tomoharu Nakashima, and Tadahiko Murata. Three-objective genetics-based machine learning for linguistic rule extraction. *Information sciences*, 136(1-4):109–133, 2001.

[48] Xiao Jiang, Chengkai Li, Ping Luo, Min Wang, and Yong Yu. Prominent streak discovery in sequence data. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1280–1288. ACM, 2011.

[49] Zhengrui Jiang. A decision-theoretic framework for numerical attribute value reconciliation. *IEEE Transactions on Knowledge and Data Engineering*, 24(7):1153–1169, 2012.

[50] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 international conference on web search and data mining*, pages 219–230, 2008.

[51] Nitin Jindal, Bing Liu, and Ee-Peng Lim. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1549–1552, 2010.

[52] Nour Jnoub, Fadi Al Machot, and Wolfgang Klas. A domain-independent classification model for sentiment analysis using neural models. *Applied Sciences*, 10(18), 2020.

[53] Nour Jnoub, Admir Brankovic, and Wolfgang Klas. Fact-checking reasoning system for fake review detection using answer set programming. *Algorithms*, 14(7):190, 2021.

[54] Nour Jnoub and Wolfgang Klas. Declarative programming approach for fake review detection. In *2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA*, pages 1–7. IEEE, 2020.

[55] Nour Jnoub, Wolfgang Klas, Peter Kalchgruber, and Elaheh Momeni. A flexible algorithmic approach for identifying conflicting/deviating data on the web. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE, 2018.

[56] Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1), March 2019.

[57] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[58] Peter Kalchgruber, Wolfgang Klas, Nour Jnoub, and Elaheh Momeni. Factcheck-identify and fix conflicting data on the web. In *International Conference on Web Engineering*, pages 312–320. Springer, 2018.

[59] Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. Fully automated fact checking using external sources. *arXiv preprint arXiv:1710.00341*, 2017.

[60] Anna Kata. Anti-vaccine activists, web 2.0, and the postmodern paradigm–an overview of tactics and tropes used online by the anti-vaccination movement. *Vaccine*, 30(25):3778–3789, 2012.

[61] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[63] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[64] Srijan Kumar, Justin Cheng, Jure Leskovec, and VS Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 857–866, 2017.

[65] Theodoros Lappas, Gaurav Sabnis, and Georgios Valkanas. The impact of fake reviews on online visibility: A vulnerability assessment of the hotel industry. *Information Systems Research*, 27(4):940–961, 2016.

[66] Tom Lauricella, Christopher S Stewart, and SHIRA Ovide. Twitter hoax sparks swift stock swoon. *The Wall Street Journal*, 23:14, 2013.

[67] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[68] Stephan Leitner, Bartosz Gula, Dietmar Jannach, Ulrike Krieg-Holz, and Friederike Wall. Understanding the dynamics emerging from infodemics: a call to action for interdisciplinary research. *SN Business & Economics*, 1(1):1–18, 2021.

[69] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. Spotting fake reviews via collective positive-unlabeled learning. In *2014 IEEE international conference on data mining*, pages 899–904. IEEE, 2014.

[70] Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *ninth international AAAI conference on web and social Media*, 2015.

[71] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948, 2010.

[72] Yuming Lin, Tao Zhu, Hao Wu, Jingwei Zhang, Xiaoling Wang, and Aoying Zhou. Towards online anti-opinion spam: Spotting fake reviews from the review sequence. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 261–264. IEEE, 2014.

[73] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[74] Bing Liu et al. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010):627–666, 2010.

[75] Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870, 2015.

[76] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[77] Diego Marcheggiani, Oscar Täckström, Andrea Esuli, and Fabrizio Sebastiani. Hierarchical multi-label conditional random fields for aspect-oriented opinion mining. In *European Conference on Information Retrieval*, pages 273–285. Springer, 2014.

[78] Delia Mocanu, Luca Rossi, Qian Zhang, Marton Karsai, and Walter Quattrociocchi. Collective attention in the age of (mis) information. *Computers in Human Behavior*, 51:1198–1204, 2015.

[79] Rachel R Mourão and Craig T Robertson. Fake news as discursive integration: An analysis of sites that publish false, misleading, hyperpartisan and sensational information. *Journalism Studies*, 20(14):2077–2095, 2019.

[80] Arjun Mukherjee and Bing Liu. Modeling review comments. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–329, 2012.

[81] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM, 2012.

[82] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. What yelp fake review filter might be doing? In *Seventh international AAAI conference on weblogs and social media*, 2013.

[83] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, Natalie Glance, et al. Fake review detection: Classification and analysis of real and pseudo reviews. *Technical Report UIC-CS-2013–03, University of Illinois at Chicago, Tech. Rep.*, 2013.

[84] Salman Bin Naeem and Rubina Bhatti. The covid-19 'infodemic': a new front for information professionals. *Health Information & Libraries Journal*, 37(3):233–239, 2020.

[85] Brendan Nyhan, Jason Reifler, and Peter A Ubel. The hazards of correcting myths about health care reform. *Medical care*, pages 127–132, 2013.

[86] Viral Panchal, Shailesh Pillai, and Ashutosh Singh. Truth finder algorithm for multiple conflicting information providers on the web. *International Journal of Computer Applications*, 5:1–4, 2012.

[87] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 2004.

[88] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[89] Manali S Patil and AM Bagade. Online review spam detection using language model and feature selection. *International Journal of Computer Applications*, 59(7), 2012.

[90] Parth Patwa, Shivam Sharma, Srinivas Pykl, Vineeth Guptha, Gitanjali Kumari, Md Shad Akhtar, Asif Ekbal, Amitava Das, and Tanmoy Chakraborty. Fighting an infodemic: Covid-19 fake news dataset. In *International Workshop on Combating On line Ho st ile Posts in Regional Languages dur ing Emerge ncy Si tuation*, pages 21–29. Springer, 2021.

[91] David Martin Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2:37–63, 01 2011.

[92] Cristina M Pulido, Beatriz Villarejo-Carballido, Gisela Redondo-Sama, and Aitor Gómez. Covid-19 infodemic: More retweets for science-based information on coronavirus than for false information. *International Sociology*, 35(4):377–392, 2020.

[93] Carroline Dewi Puspa Kencana Ramli. Detecting incompleteness, conflicting and unreachability xacml policies using answer set programming. *arXiv preprint arXiv:1503.02732*.

[94] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, 2017.

[95] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *Proceedings of the 33. AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 6940–6948. AAAI, 07 2019.

[96] Tirath Prasad Sahu and Sanjeev Ahuja. Sentiment analysis of movie reviews: A study on feature selection & classification algorithms. In *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, pages 1–6. IEEE, 2016.

[97] Franco Salvetti, Stephen Lewis, and Christoph Reichenbach. Automatic opinion polarity classification of movie reviews. *Colorado Research in Linguistics*, 17(1), 01 2004.

[98] Kin Meng Sam and Chris Chatwin. Ontology-based sentiment analysis model of customer reviews for electronic products. In *Encyclopedia of Information Science and Technology, Third Edition*, pages 892–904. IGI Global, 2015.

[99] Sunil Saumya and Jyoti Prakash Singh. Detection of spam reviews: a sentiment analysis approach. *Csi Transactions on ICT*, 6(2):137–148, 2018.

[100] Christina Sauper and Regina Barzilay. Automatic aggregation by joint modeling of aspects and values. *Journal of Artificial Intelligence Research*, 46:89–127, 2013.

[101] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[102] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

[103] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[104] Matla Suhasini and Badugu Srinivasu. Emotion detection framework for twitter data using supervised classifiers. In *Data Engineering and Communication Technology*, pages 565–576. Springer, 2020.

[105] Afroza Sultana, Naeemul Hassan, Chengkai Li, Jun Yang, and Cong Yu. Incremental discovery of prominent situational facts. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 112–123. IEEE, 2014.

[106] Karthik Sundararajan and Anandhakumar Palanisamy. Multi-rule based ensemble feature selection model for sarcasm type detection in twitter. *Computational Intelligence and Neuroscience*, 2020, 2020.

[107] Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca de Alfaro. Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*, 2017.

[108] Enhua Tan, Lei Guo, Songqing Chen, Xiaodong Zhang, and Yihong Zhao. Spammer behavior analysis and detection in user generated content on social networks. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 305–314. IEEE, 2012.

[109] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[110] Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath. Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, 57:117–126, 2016.

[111] Kimitaka Tsutsumi, Kazutaka Shimada, and Tsutomu Endo. Movie review classification based on a multiple classifier. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation*, pages 481–488, 2007.

[112] Özlem Uzuner, Xiaoran Zhang, and Tawanda Sibanda. Machine learning and rule-based approaches to assertion classification. *Journal of the American Medical Informatics Association*, 16(1):109–115, 2009.

[113] Levenshtein V. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[114] Brett Walenz, Y Wu, S Song, Emre Sonmez, Eric Wu, Kevin Wu, Pankaj K Agarwal, Jun Yang, Naeemul Hassan, Afroza Sultana, et al. Finding, monitoring, and checking claims computationally based on structured data. In *Computation+ Journalism Symposium*, 2014.

[115] Hongning Wang, Yue Lu, and ChengXiang Zhai. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–626. ACM, 2011.

[116] Hongwei Wang and K. Guo. The impact of online reviews on exhibitor behaviour: evidence from movie industry. *Enterprise Information Systems*, 11:1–17, 09 2016.

[117] William Yang Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[118] Xinyue Wang, Xianguo Zhang, Chengzhi Jiang, and Haihang Liu. Identification of fake reviews using semantic and behavioral features. In *2018 4th International Conference on Information Management (ICIM)*, pages 92–97. IEEE, 2018.

[119] Zhaoxia Wang, Chee Seng Chong, Landy Lan, Yinping Yang, Seng Beng Ho, and Joo Chuan Tong. Fine-grained sentiment analysis of social media with emotion sensing. In *2016 Future Technologies Conference (FTC)*, pages 1361–1364. IEEE, 2016.

[120] Geoffrey I. Webb. Naïve bayes. In *Encyclopedia of Machine Learning and Data Mining*, pages 895–896. Springer, 2017.

[121] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. On one of the few objects. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1487–1495. ACM, 2012.

[122] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831, 2012.

[123] Ainur Yessenalina, Yisong Yue, and Claire Cardie. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1046–1056. Association for Computational Linguistics, 2010.

[124] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

[125] Zuraini Zainol, Mohd TH Jaymes, and Puteri NE Nohuddin. Visualurtext: a text analytics tool for unstructured textual data. In *Journal of Physics: Conference Series*, volume 1018, page 012011. IOP Publishing, 2018.

[126] Nurulhuda Zainuddin, Ali Selamat, and Roliana Ibrahim. Discovering hate sentiment within twitter data through aspect-based sentiment analysis. In *Journal of Physics: Conference Series*, volume 1447, page 012056. IOP Publishing, 2020.

[127] Dongsong Zhang, Lina Zhou, Juan Luo Kehoe, and Isil Yakut Kilic. What online reviewer behaviors really matter? effects of verbal and nonverbal behaviors on detection of fake online reviews. *Journal of Management Information Systems*, 33(2):456–481, 2016.

[128] Xianchao Zhang, Zhaoxing Li, Shaoping Zhu, and Wenxin Liang. Detecting spam and promoting campaigns in twitter. *ACM Transactions on the Web (TWEB)*, 10(1):1–28, 2016.

[129] Yanyan Zhao, Bing Qin, and Ting Liu. Clustering product aspects using two effective aspect relations for opinion mining. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 120–130. Springer, 2014.

[130] Cäcilia Zirn, Mathias Niepert, Heiner Stuckenschmidt, and Michael Strube. Fine-grained sentiment analysis with structural features. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 336–344, 2011.

Curriculum Vitae

Personal Data

Name	Nour Jnoub
Date of Birth	March 24th, 1990
Nationality	Austrian
Mail	nour.jnoub@wu.ac.at
Web	`https://www.wu.ac.at/en/deco/team/dipling-nour-jnoub-bsc`
ORCiD	`https://orcid.org/0000-0002-7769-5176`

Education

03/2017 – current	University of Vienna (Universität Wien), Vienna, Austria Dr. techn. (PhD), Computer Science
2014 – 2016	University of Klagenfurt, Austria Master of Science, Information Technology
2007 – 2013	Arab Academy for Science, Technology and Maritime Transport Bachelor of Science (BSc), Computer Engineering

Experience

05/2021 – current	Vienna University of Business and Economics (WU Wien), Vienna, Austria Research and Teaching Assistant, Research Unit of Digital Ecosystems
03/2017 – 03/2021	University of Vienna (Universität Wien), Vienna, Austria University Assistant (Prae-Doc), Faculty of Computer Science Research Group Multimedia Information Systems

Publications

– Fact-Checking Reasoning System for Fake Review Detection Using Answer Set Programming (2021)

– A Domain-Independent Classification Model for Sentiment Analysis Using Neural Models (2020)

– Declarative Programming Approach for Fake Review Detection (2020)

– Detection of Tampered Images Using Blockchain Technology (2019)

– A Flexible Algorithmic Approach for Identifying Conflicting/Deviating Data on the Web (2018)

– FactCheck - Identify and Fix Conflicting Data on the Web (2018)