UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

Episode detection and mining in real world applications

Pietro Dell'Oglio

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Episode detection and mining in real world applications

Pietro Dell'Oglio

Advisors:

Prof. Francesco Marcelloni, prof. Alessandro Lenci

Head of the PhD Program:

Prof. Stefano Berretti

Evaluation Committee:
Prof. Bernardo Magnini, *Fondazione bruno kessler*
Prof. Giuseppe Fenza, *University of Salerno*

XXXVI ciclo — February 2024

Acknowledgments

Throughout my PhD and the process of writing this thesis, I received a great amount of support and guidance, for which I am deeply grateful.

First, I extend my sincere appreciation to my advisor, Professor Francesco Marcelloni, whose invaluable assistance and guidance have been crucial throughout my doctoral journey. His insights and support have enabled me to grow as a researcher.

I am truly thankful to Professor Alessandro Lenci. His teachings, as well as his invaluable advice, have significantly contributed to my academic development. I am also grateful to Alessandro Bondielli and Lucia Passaro for both being helpful and encouraging (and for putting me back in line when needed) during this journey.

I would like to acknowledge also my Supervisory Committee, formed by Professors Pietro Ducange, Francesco Marcelloni, Alessandro Lenci and Alessio Bechini. Their annual input and feedback have proven immensely valuable in continuing my research in the right direction.

I thank my Evaluation Committee, formed by Bernardo Magnini and Giuseppe Fenza, for their helpful suggestions and the review of my work.

I must express my gratitude to all my collaborators and colleagues from both the information engineering and computer science departments and the CoLingLab.

Part of the present work was developed during my involvement in the project AUTOMIA, funded by Tuscany Region in the framework of the regional program “POS FESR Toscana 2014-2020”. I would like to thank all the partners of the project who enabled me to see my research work applied to an industrial context.

Lastly, I would like to extend my thanks to my friends and family, as well as all the other essential people in my life. To my parents, who have always been there, unshakable in their support and belief in me. To my friends, who have stood by me and shared in both the best and most challenging moments. A special mention to the theatrical improvisation group "ADA", which evolved into a circle of friends. To all the other vital people in my life, whether they are still present or have moved on, I offer my most heartfelt gratitude.

Abstract

In data mining, an *episode* is defined as a sub-sequence of events or symbols extracted from a single sequence of events or symbols. Frequent Episode Mining is the field of data mining that searches for episodes that occur frequently in a sequence over time, following some specific temporal order.

In this thesis, we also consider episodes referring to the domains of Natural Language Processing (NLP) and Text Mining. In particular, we deal with *text episodes*, defined as text sub-sequences in an extended text sequence, and *interesting text episodes*, which are text episodes meeting particular conditions relevant to the application domain. We propose different approaches for mining interesting episodes in three distinct case studies. For all the studies, we first present the solutions and then discuss the obtained results.

The first case study is the News Collector, a system developed to assist readers in automatically gathering comprehensive information about a news event from various sources. We have proposed a methodology based on transformer neural models, text summarization models, and decision rules to highlight information divergence between texts. In this context, the "text sequence" is represented by several articles on the same topic of a given reference article already read by the user. The "interesting text episode" to mine consists of a piece of information represented by a set of sentences that form a sub-sequence of non-redundant information within the larger sequence of articles. This subsequence is summarized and returned to the user.

The second case study consists of a framework for mining frequent subsequences of actions from logs that report the interaction of operators with specific applications. The system aims to automatically spot repetitive subsequences of actions that are suitable for possible automation. This framework has been developed within the AUTOMIA project and it has also been used to investigate how techniques of Episode Mining can be employed in conjunction with a specific similarity metric.

In the third case study, we investigate the issue of unverified information and its dissemination on the web. Here, an "interesting text episode" is defined as an article or a social media post classified as fake within a sequence of articles or posts. We introduce a methodology for collecting and labelling both authentic and fake news, while also establishing ground truth for real-world events. This methodology is then applied to two specific events: the 2019 Notre Dame fire and the more recent Ukraine-Russian war. We employ the concept of *information divergence* to detect fake news: whenever the information begins to diverge from the one obtained by trustable sources, the information is likely fake.

Contents

Chapter 1

Introduction

In recent years, social media and online newspapers have increasingly established themselves as the main source of dissemination and spread of information. The COVID-19 pandemic and the subsequent lockdowns accelerated the widespread use of these platforms. Consequently, it is not surprising that we are inundated with reliable and less reliable content, often filtered under different personal, ideological, and political perspectives.

In this context, language technologies have become a crucial point of reference in the analysis, management, and development of resources and tools that utilize textual data, in particular text streams extracted from the web.

The earliest NLP and Text Mining methods relied on feature engineering. Salient features were defined and extracted from raw data, and models were provided with an appropriate inductive bias to learn from these data. With the emergence of deep learning, the focus shifted from feature engineering to architecture engineering, enabling models to automatically learn features during their training (Liu et al., 2023a). The introduction of advanced architectures, such as transformer-based ones (Devlin et al., 2018), gave rise to a new language modelling era, in which a Language Model (LM) is pre-trained to predict the probability of observed textual data, and subsequently fine-tuned introducing additional parameters to adapt it to different downstream tasks.

The bigger the models become, the more their transfer learning abilities increase. With the advent of Large Language Models, and their applications (i.e., chatGPT), instead of adapting pre-trained LMs to downstream tasks, the tasks themselves are reformulated and directly requested to a frozen LLM, without any fine-tuning, typically with the help of a textual prompt. The advantage of this method is that, with a suitable set of prompts, a single LM trained in an entirely unsupervised manner can be applied to a wide range of tasks. However, there is a caveat: this method introduces the need for prompt engineering, that is finding the most effective prompt to enable an LM to solve a specific task. Recent techniques, such as Prompt Tun-

ing, face the problem but require expensive computational resources (Lester et al., 2021).

Several advancements have been seen in fields such as Natural Language Processing (NLP) and Text Mining, and the recent introduction of advanced Artificial intelligence (AI) systems based on Large Language Models (LLMs) has raised important questions on a global and multidisciplinary scale. These advancements, and the introduction of such large models with exceptional transfer learning capabilities, have presented new challenges from both NLP and Data Mining perspectives.

Indeed, social media platforms have become popular as hubs for news, and in general content consumption and sharing. Both online newspapers and users regularly post a lot of content on these platforms. However, the information flood risks making it confusing and difficult to find useful information or distinguish between real and constructive content from deceitful ones. The problem can be approached from three different perspectives.

Firstly, newspapers differ from each other in the way they deal with specific topics. It can be difficult for a user to obtain clear and complete information. Different sources could treat the same event in different ways, and some content information might be contradictory. Thus, each piece of news can contain in practice different details on a specific topic. Users desire to acquire comprehensive information on a particular subject to form their personal opinions. However, they would like to avoid reading all the related news, which often contains redundant information, and would prefer to focus only on the unique content.

Secondly, due to the unstructured nature of social media settings, it can be challenging to pinpoint the exact piece of content desired by the user. In this context, the identification of specific *episodes* within the information stream lies in the field of Sequential Pattern Mining, particularly through a multidisciplinary approach that connects Frequent Episode Mining (FEM) and Text Mining. FEM involves the identification of sub-sequences within a single sequence, where instances of these subsequences occur frequently and follow a specific temporal order. From an NLP and Text Mining perspective, a text episode represents an interesting segment of text (i.e., a sub-sequence) extracted from the information stream itself (i.e., the sequence). This approach is better explained in Chapter 2.

Finally, certain newspapers or ordinary web users may disseminate misleading information, or even fake news (Bondielli and Marcelloni, 2019). The problem of unverified information and its propagation on the web is of particular interest for the present work and also for the research community in general. Over the last few years, the problem of Fake News Detection has gained significant attention within the NLP research community. The proliferation of fake news, as well as disinformation and misinformation in general, is closely linked to social media. These platforms have indeed become one of the most fertile grounds for the spread of dis-

information (Zubiaga et al., 2018b). The majority of current approaches to Fake News Detection treat the problem as a standard binary or multi-class classification task, by leveraging content features (i.e., the presence of specific lexical, syntactic, or morphosyntactic markers) or context features (i.e., how the news spreads on social media, who propagates it, who are their followers, etc.), or a combination of both. However, the problem remains much more complex than that. Often, fake news can be challenging to identify because they are presented credibly and relate to plausible events. Real-world events can be distorted by altering several key facts that may lead public opinion toward the direction pushed by the fake news authors, who have often political or social motivations.

An analysis that focuses on the meaning of words and sentences, while leveraging the relationships between the content of real and fake news to distinguish between them, may offer a more effective modelling approach for the problem. This approach can also be extended to real-world scenarios. Moreover, an approach based on information divergence between the news to be debunked and the real and verified story could prove to be an interesting method. Such an approach could not only aid in identifying the most deceptive fake news but also provide explicit information about the extent of the actual divergence.

1.1 Goals and contributions

The present work focuses on the fundamental concept of *episode* derived from Data Mining and applies it to various real-world scenarios. These scenarios encompass mining sequences of log actions, and extending the concept to two significant areas within Natural Language Processing (NLP): semantic similarity, viewed from an information divergence perspective, and fake news detection.

This work has three primary objectives and each of them correspond to a case study: the first one is aimed at investigating if it is possible to identify and extract novel and non-redundant episodic information from multiple newspaper articles concerning a reference article, exploiting both transformer neural models and decision rules. The second case study intends to evaluate how, in a traditional data mining context, an approach based on Frequent Episode Mining, combined with similarity metrics, can be utilized to discover interesting patterns within longer sequences. Subsequently, we explore how this approach can be adapted to an NLP scenario. Lastly, the third case study involves evaluating the extent to which semantics and meaning can affect and improve the performance in automatic Fake News Detection: in this work, attention is particularly focused on the development of a strategy for collecting and labelling datasets based on the meaning of news articles in relation to verified and trustworthy ground truth. In the future, our efforts will be extended to implement a fake news detection model based on an information

divergence approach.

Information Divergence The first main goal is to describe a framework for extracting information divergence from one (or more) target texts in comparison to a single reference text. This is accomplished by leveraging sentence similarity and employing decision rules. To demonstrate the practical application of this framework, we have developed a user-friendly system called News Collector. When provided with a news article (referred to as the *reference article*), which has already been read by a user, and a set of *target articles* related to the same topic or event, the system assesses similarities and differences at the sentence level among the target articles and in comparison to the reference article. It generates an abstractive summary of the unique content, and outputs this summary to the users, allowing them to save time and effort. We evaluated the News Collector by conducting a two-step analysis. Initially, we assessed its efficacy in discerning content differences between the reference article and associated articles by leveraging human judgments obtained through crowdsourcing as the ground truth. We obtained an average F1 score of 0.772 against average F1 scores of 0.797 and 0.676 achieved by two state-of-the-art approaches based, respectively, on model tuning and prompt tuning. Notably, these approaches necessitate an appropriate tuning phase and, therefore, greater computational effort. Second, we asked a selected group of people to assess how well the system-generated summaries represent the information that is not present in the article initially read by the user. The obtained results are promising.

Episode Mining The second main goal of this work is to explore the field of data mining, specifically Sequential Pattern Mining, with a focus on the problem of Frequent Episode Mining. In this context, a two-step framework has been designed to extract sequences of actions that have the potential to be automated. This framework has been developed within the context of the AUTOMIA project. We assessed the effectiveness of the approach using a benchmark dataset, supplemented by the presentation of its functioning on a real-world dataset of activity logs generated in the context of the AUTOMIA project.

Fake News Detection The third main goal of this work is to address the current challenges in the field of fake news detection and to propose solutions, encompassing both resource creation and problem modelling aspects. In the first case, we present a methodology for collecting and labelling real and fake news, along with establishing a ground truth for real-world events. This methodology is then applied to two specific events, namely the Notre Dame fire in 2019, and the recent Ukraine-Russian war. Finally, we introduce the concept of information divergence within a demo framework, which is based on the one used for the AUTOMIA project. We

evaluated the framework on the Notre Dame fire dataset, obtaining promising performances. This serves as a primary foundational step for future directions in this area.

Contributions The contributions of the present work can be summarized as follows:

- Investigation and evaluation of data mining approaches in a Natural Language Processing perspective.
- Development of a sentence-level-based methodology to highlight unique and different content (i.e., information divergence) in one or more texts with respect to a reference text.
- Development of a system that, given a newspaper article already read by the user, collects and summarizes new information in similarly new articles.
- Development of a two-step methodology to mine interesting sequences of actions of log in a dataset.
- Development of a methodology for data collection in the realm of fake news.
- Proposal of an approach for fake news detection based on information divergence and text episode mining.

1.2 Structure of the thesis

This thesis is organized as follows. In Chapter 2 a preliminary discussion of the methodologies and the philosophy followed in the present work is reported. In particular, a dissertation on the concept of *text episode* is presented. In Chapter 3 a thorough literature review is conducted on the three main fields investigated in this work: distributional semantic models (Section 3.1), Sequential Pattern mining (Section 3.2), with a particular focus on Frequent Episode Mining, and fake news detection (Section 3.3). Chapter 4 presents the first case study examined, namely the News Collector, an in-development system aimed at assisting readers in automatically gathering comprehensive and summarized information about an event from various sources, based on an approach focused on information divergence. Chapter 5 reports the two-step framework designed to mine sequences of actions in the context of the AUTOMIA project. Chapter 6 tackles the problem of fake news detection. Finally, Chapter 7 provides an overview of the findings and insights obtained from the previous chapters. Some conclusions and possible future directions are drawn.

Chapter 2

Methodologies

In the last few years, Natural Language Processing (NLP) and Text Mining have seen significant advancements in the development of increasingly accurate models and frameworks. Traditionally, task-specific models were trained on datasets targeted for the task. However, due to the challenges in obtaining high-quality resources for such models, attention has shifted towards feature engineering. Between 2017 and 2019, the learning of NLP models experienced a significant shift toward the *pre-train and fine-tune paradigm*. A Language Model (LM) with a fixed architecture is initially pre-trained to predict the probability of observed textual data. Then, LMs are adapted to different downstream tasks by introducing new parameters, fine-tuning them with task-specific objective functions, and developing reliable techniques to collect and label small but high-quality datasets, such as the In-context annotation of topic-oriented datasets in the case of fake news detection task (Passaro et al., 2022), which is described in depth in Chapter 6.

In the last two years, thanks to the development of Large Language Models (LLMs), the *pre-train and fine-tune paradigm* has been replaced by the *pre-train, prompt, and predict procedure* (Liu et al., 2023a). The idea is that, instead of adapting pre-trained LMs to downstream tasks, these tasks are reformulated to be solved with the help of a textual prompt. According to this paradigm, a large number of different tasks, not necessarily just closely related to that of NLP, can be solved by paying attention to prompt engineering. The data used for many of these tasks are sequential.

In fact, many of these tasks involve the classification of particular pieces of texts in two or more classes (such as, among the others, fake news detection, fact checking, sentiment analysis, hate-speech detection), or the generation of new texts that contain a subset of the information contained in one (or more) longer text(s) (text summarization), or even the generation of new texts just collecting the information from elsewhere. The common thread between all these tasks is the aim of mining a final text sequence that can be considered as a sub-sequence of a longer one. We

name these sub-sequences *episodes*

The idea of *episode* comes from Frequent Episode Mining (FEM). It is a field related to Sequential Pattern Mining (SPM), a branch of data mining that deals with finding statistically relevant regularities among data samples whose values are expressed through ordered sequences (Bechini et al., 2023).

Frequent Episode Mining is aimed at discovering sub-sequences in a single sequence where instances occur frequently, each following a specific temporal order (Bechini et al., 2023). This problem holds significant relevance across various application domains, particularly in the context of system log analysis and, we can argue, in Natural Language Processing as well. Examples of applications are the monitoring of network traffic for detecting attacks, the monitoring of telecommunication alarm signals, or the analysis of usage data for recurring patterns.

As detailed in Chapter 3, Frequent Episode Mining is an approach within Process Mining. The objective of a process mining algorithm is to identify specific sequences within a dataset that occur a statistically significant number of times. This process aims to verify particular, potentially interesting recurrences (Han et al., 2007). The problem of the identification of such sequences has been explored in the literature under different lights and different domains, leading to the classic research problems known as Frequent Itemset Mining (FIM) and Sequential Pattern Mining (SPM). FIM was first developed as a tool for market basket analysis to study the behavior of customers by considering what products are frequently bought together (Agrawal et al., 1993). Any single purchasing event, or transaction, can be modeled as a set of items, usually called *itemset*. They are our starting point.

Let D be a transactional database consisting of a set of transactions $\{T_1, T_2, ..., T_i\}$. We can see a single transaction as the basic entry in the database. Each one is typically identified by a Transaction ID (TID) and corresponds to a set of items (objects or symbols) out of a set of possible items $I = \{i_1, i_2, ..., i_n\}$. For example, regarding the sales in a shop, each transaction represents a receipt, and thus the items purchased by a particular customer. In this case, I is the set of all the items sold by the shop. A subset $X \subseteq I$ is called an itemset. An itemset that contains k items is denoted as k-itemset. For instance, the set $\{a, b\}$ is a 2-itemset.

The availability of databases containing transactions led to the development of Association Rule Mining, which consists of finding association rules in a transaction database (Agrawal et al., 1993). An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, $A \neq \emptyset$, and $B \neq \emptyset$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$. That is,

$$support\,(A \Rightarrow B) = P\,(A \cup B) \tag{2.1}$$

The percentage of transactions in D containing A that also contain B is named

confidence c, and is formalized as,

$$confidence\,(\mathrm{A} \Rightarrow \mathrm{B}) = P\,(\mathrm{B}|\mathrm{A}) \tag{2.2}$$

Rules that satisfy both a minimum support threshold (min sup) and a minimum confidence threshold (min conf) are called *strong*. The occurrence frequency of an itemset is the number of transactions that contain the itemset. Whenever the temporal ordering of transactions is relevant, we can handle temporally ordered sequences of itemsets.

Let $SeqD = \{S_1, S_2, ..., S_j\}$ be a sequence database consisting of j sequences. A sequence is identified by a Sequence ID (SID) and consists of an ordered list of elements. In the most general case, an element is an itemset (usually, a non-empty one). Usually, the ordered elements of the sequence (i.e., the itemsets) are referred to as *events*. Given a sequence $S = \langle E_1, E_2, ..., E_n \rangle$, the corresponding length $|S|$ can be defined as the number of events in S. On the basis of the total number of objects it contains, a sequence is referred to as k-sequence, with $k = \sum_{i=1}^{n} |E_i|$.

An event is always associated with a timestamp, and it may contain one or more items out of a given set. The frequent sub-sequences of events that we are interested in, occurring within specified time intervals in a given partial order, are usually called *episodes*.

To apply this concept to other fields, in particular to Natural Language Processing and Text Mining, it is necessary to broaden the definition of episode. We can take as an example the analysis of protein strings in bioinformatics. A substring can be obtained by removing both the head and the tail from the original string. For instance, if we have the original string "abcdefghi", removing the head "abc" and the tail "ghi" would result in the substring "def": "abc**def**ghi". It is essential to note that obtaining a substring is not limited to this single method. By deleting different substrings from the original string, we can acquire a variety of substrings. For example, to obtain the substring "acdgh" we need to delete "a," "cd," and "gh" from the original string: "**a**b**cd**ef**gh**i"

In the most general case, when describing an event, a similar concept can be applied. According to the Frequent Episode Mining theory, some events can occur concurrently, i.e. may have the same timestamp. To highlight this aspect, sequences that involve potentially concurrent events are referred to as *complex sequences*.

A complex sequence S of events, each with one or more items out of I, is represented by a temporally ordered list of tuples of type $(SE_{t_i} t_i)$, where SE_{t_i} is the event occurring at timestamp t_i. An episode is thus a non-empty, totally-ordered set of events of the form $\langle E_1, E_2, ..., E_p \rangle$, where E_i is a subset of I and E_i appears before E_j for all integers i and j, with $i < j$, in the interval $[1, p]$. The occurrence of an episode is given by a time interval $[t_s, t_e]$ where t_s and t_e are the start and end timestamps for the episode, respectively. The support of an episode is given by the number of

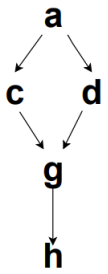

Figure 2.1: An example of complex sequence

its occurrences within the sequence. Figure 2.1 is an example of complex sequence described as $[(a,1)\ (c,2)\ (d,2)\ (g,3)\ (h,4)]$. Both the sub-sequences "acg" and "adg" can be considered valid episodes.

In this case, the order is partial. In fact, we can consider the partial order as a generalization of the total order.

In past years, authors in (Ahonen et al., 1997) have explored the idea that general data mining methods can apply to text analysis tasks under certain conditions. To do this, they formulate a definition of *text episode* as a pair $\alpha = (V, \leq)$, where V is a collection of features vectors, and $\leq$ is a partial order on V. Given a text sequence S, a text episode $\alpha = (V, \leq)$ *occurs* within S if there is a way of satisfying the feature vectors in V using the feature vectors in S so that the partial order $\leq$ is respected. This means that the feature vectors of V can be found within S in an order that satisfies the partial order $\leq$.

According to this paradigm, a sub-sequence is considered an episode where all the feature vectors of the sub-sequence occur within a certain proximity in S. The limit of the closeness is determined by a window size W, which sets the limit within which the episode must occur.

The authors of this formulation considered a feature vector to be an ordered set of features. In this context, a feature can be defined as a word, a grammatical feature (such as part of speech, case, or number), a punctuation mark, a special character, or a structure tag. Nowadays, this formulation can be extended by incorporating word and sentence embeddings generated from Neural Language Models as feature vectors. These embeddings represent the features based on the contextual properties of the word or sentence in a vector space.

Another useful distinction is the classification of partial orders into two types: total and trivial partial order (Ahonen et al., 1997). In the case of a total partial order, the feature vectors of each episode follow a fixed order, and such episodes are referred to as *serial*. On the other hand, in the case of a trivial partial order, the order is not significant, and these episodes are known as *parallel*. For instance, a serial text

episode can be exemplified by a sentence, which consists of a sequence of related words conveying a specific meaning. Similarly, a piece of text can be considered a text episode, comprising a sequence of related sentences or multiple pieces of text (i.e., a sequence of paragraphs in a newspaper article or chapters in a novel). Conversely, a parallel text episode can be represented by a collection of co-occurring terms that collectively describe the content of a document more effectively than any individual term.

According to the author's definition, a text episode is considered interesting when it is frequent. In other words, a frequent text episode α is a sub-sequence of S with a support in S above a given support threshold. The support of α in S, with respect to a given window size W is defined as the number of minimal occurrences of α in S.

When dealing with feature vectors that incorporate context, such as sentence embeddings extracted from Language Models, it becomes necessary to compute support in order to identify all similar sequences. This is typically done using a chosen similarity measure, such as cosine similarity. It is important to note that achieving perfect matching becomes impossible in this context.

However, we use the concept of *interesting text episode*, which includes the definition of *frequent text episode* as a particular and specific case of it. According to the definition provided by the Cambridge Dictionary[1], an episode is a single event or group of related events. It represents a distinct and separate occurrence within a larger series. In our context, we define an *interesting text episode* as a sub-sequence of S that fulfills the conditions specific to the task at hand. For example, in the context of Fake News Detection, the sequence S represents the collection of all the examined articles or social media posts. An episode is represented by an individual article or social media post, comprising a sequence of sentences conveying information. An *interesting text episode* refers to a specific newspaper article or social media post that is classified as Fake News.

In the rest of this work, we present three case studies along with a literature review for each aspect. In these case studies, we investigate traditional approaches and use the concept of 'interesting text episodes' as a link to these three scenarios.

The first case study we examine is the News Collector, a system developed to help readers automatically gather comprehensive and summarized information about an event from various sources. The core of this system lies in a methodology designed to highlight information divergence between texts. In this scenario, the initial sequence is represented by a set of multiple newspaper articles (referred to as targets) all related to the same topic. The text episode we aim to mine consists of a piece of information in the form of a set of sentences, which constitutes a sub-sequence within the larger sequence of articles. This sub-sequence does not appear in an external

[1]https://dictionary.cambridge.org/dictionary/english/episode

newspaper article (referred to as a reference); its information diverges from the reference. For a more comprehensive explanation of this case study, refer to Chapter 4.

The second case study involves a two-step framework designed to extract sequences of actions with the potential for automation. This framework utilizes log data generated from the interactions between a human operator and specific software applications. While this problem domain primarily falls within the realm of Data Mining rather than Natural Language Processing, this case study is valuable in illustrating that techniques derived from Frequent Episode Mining can be effectively employed in conjunction with a similarity metric. This approach helps uncover archetypal patterns, although they may be somewhat coarse for application to textual data. For a more comprehensive explanation of this case study, refer to Chapter 5.

The last case study revolves around the Fake News Detection task. In this scenario, we work with a collection of articles or social media posts that encompass both real and fake content. Here, an interesting text episode is defined as an article or a social media post that has been classified as fake. In this context, we introduce a novel methodology for collecting and labelling both fake and real news. Furthermore, we take the initial steps in applying the concept of information divergence, as proposed in the first case study, within a framework inspired by the one described in the second case study. We hope that, in the future, this line of research could facilitate the detection of fake news. For a more detailed explanation of this case study, refer to Chapter 6.

Chapter 3

Literature Review

3.1 Textual and Semantic Similarity

Natural Language Processing (NLP) is a field of artificial intelligence and computational linguistics that focuses on the interaction between computers and human language. It involves the development and application of algorithms, models, and techniques to empower computers to comprehend, interpret, and generate human language in a meaningful manner.

NLP encompasses a wide range of tasks and applications, including but not limited to:

1. Text Parsing and Tokenization: breaking down text into smaller units such as words or sentences for further analysis.

2. Language Understanding: extracting meaning and context from text, including tasks such as part-of-speech tagging, named entity recognition, and syntactic analysis.

3. Sentiment Analysis: determining the emotional tone or sentiment expressed in a piece of text, often used in social media monitoring or customer feedback analysis.

4. Machine Translation: automatically translating text from one language to another, enabling communication across language barriers.

5. Information Extraction: identifying and extracting structured information from unstructured text, such as extracting names, dates, or locations from news articles.

6. Question Answering: building systems that can understand and respond to natural language questions, often using techniques like information retrieval and text summarization.

7. Text Generation: creating coherent and contextually relevant text, such as generating product descriptions or writing news articles.

NLP relies on various techniques and methodologies, including statistical models, machine learning algorithms, deep learning, and linguistic rules. It combines knowledge from linguistics, computer science, and cognitive science to bridge the gap between human language and machine understanding, enabling computers to process, analyze, and generate human language effectively.

The definition of Natural Language Processing just given is obtained from ChatGPT using a simple prompt: "Give me the definition of Natural Language Processing". The research in NLP, particularly in Natural Language Understanding (NLU) within AI, has been brief but intense, characterized by a series of ups and downs (Lenci, 2023). The most recent generations of language models have indeed taken a significant leap forward, as evident from reading the definition of NLP provided by chatGPT. The mastery of tasks such as text generation, language understanding, and text summarization (among others) is truly impressive and currently has no competitors.

In this subsection, we will describe the state-of-the-art of language modelling, from the definition of distributional semantics to the description of the current standard Language Models, Large Language Models, and the prompting era.

From distributional semantics to Language Models

Distributional Semantics (DS) is a subfield of Applied and Computational Linguistics that focuses on developing theories and methods for representing and acquiring the semantic properties of linguistic items based on their distributional properties in text corpora (Lenci, 2018). The theoretical assumption of DS has become known as the Distributional Hypothesis (DH). It states that words with similar linguistic contexts tend to have similar meanings (Lenci, 2008). The idea comes from the insights of American structural linguists, particularly from the work of Harris, who stated that "difference of meaning correlates with difference of distribution" (Harris, 1954).

Distributional models have also been explored in cognitive science and psychology. Notably, an important figure in this field is Miller, who utilized Harris's analysis to establish an empirical foundation for the notion of similarity in semantics (Miller, 1971). Miller & Charles defined the semantic similarity in distributional terms as a "function of contexts in which words are used" (Miller and Charles, 1991). According to this paradigm, words are distributed in a vectorial space, and their semantic similarity is determined by the distance between their distributional representations (Lenci, 2018). We usually refer to distributional representations of words as vectors or *word embeddings* (Lenci, 2008).

A significant contribution of distributional semantics in the field of Natural Language Processing and Computational Linguistics stems from its application in Information Retrieval. The Vector Space Model (Salton et al., 1975) was originally developed to address the task of automatically indexing textual documents. The authors employed an approach based on space density computations. In particular, they represented a collection of documents with a matrix, where its rows are vectors of words (or some kind of lexical items), and the columns are vectors corresponding to documents. Each value of the matrix represents the number of occurrences of a word in a particular document. In this way, it becomes easy to identify semantically associated words by measuring the similarity of their corresponding vectors (Lenci, 2018).

The Latent Semantic Analysis model was proposed for the first time in 1997 by Landauer & Dumais (Landauer and Dumais, 1997). It is a count model for distributional semantics that uses a word-document co-occurrence matrix. It assumes that if two words recur in similar documents, they tend to be semantically related. Singular Value Decomposition is then employed to reduce the dimensionality of the matrix while preserving the similarity structure among the columns.

Over the years, several other distributional models emerged from both computational linguistics and applied linguistics perspectives.

The Hyperspace Analogue of Language (HAL) model utilizes a global cooccurrence learning algorithm to encode the context in which words occur (Burgess, 1998). Thanks to this encoding, researchers can form meaningful representations in a high-dimensional context space. Latent Relational Analysis (LRA) is a method for measuring relational similarity (Turney, 2006), which extends the Vector Space Model. Authors distinguish relational similarity from attribute similarity. The first one detects analogous, while the second one detects synonyms (Turney, 2006). Sebastian Padó and Mirella Lapata introduce a framework for constructing semantic spaces using syntactic dependency relations (Padó and Lapata, 2007). Distributional Memory (DM) frameworks were also proposed. The corpus-based framework described by Baroni and Lenci (Baroni and Lenci, 2010) extracts distributional information from the corpus in the form of a set of weighted word-link-word tuples fixed into a third-order tensor (Baroni and Lenci, 2010). Different matrices could then be generated from the tensor, with rows and columns that constitute natural spaces dealing with different semantic tasks (Baroni and Lenci, 2010). High-dimensional Explorer (HiDEx) is a generalization of HAL that increases the range of parameter settings (Shaoul and Westbury, 2010).

From the perspective of NLP and text mining, many applications related to text similarity are based on *language models*. A *language model* defines a probability distribution over a sequence of tokens (Goodfellow et al., 2016). These are usually an approximation of the concept of word, and always discrete entities. N-grams are a

kind of language model based on models of sequences with a fixed length. We could define an *n*-gram as a sequence of *n* tokens. For example, an *1*-gram (unigram) is a n-gram with $n = 1$, a *2*-gram (bigram) is a n-gram with $n = 2$, a *3*-gram (trigram) is a n-gram with $n = 3$, and so on. N-gram-based models define the conditional probability of the *n*-th token given the previous $n - 1$ tokens. The probability distribution over longer sequences is then given by the products of these conditional distributions:

$$P(x_1, ..., x_\tau) = P(x_1, ..., x_n - 1) \prod_{t=n}^{\tau} P(x_t | x_t - n + 1, ..., x_t - 1) \tag{3.1}$$

Usually we train both an *n*-gram model and a *n*-1 gram model simultaneously:

$$P(x_t | x_t - n + 1, ..., x_t - 1) = \frac{P_n(x_t - n + 1, ..., x_t)}{P_n - 1(x_t - n + 1, ..., x_t - 1)} \tag{3.2}$$

The n-gram-based models have been used principally in the context of statistical language modelling for many decades, among others, by Jelinek (1980), Katz (1987), and Chen and Goodman (1999). These models were enhanced over time with the application of *smoothing* and *back-off methods* (Chen and Goodman, 1999). Furthermore, to enhance the statistical efficiency of these models, *class-based language models* were introduced (Brown et al., 1992).

Neural Networks

The Neural Network Language Model (NNLM) was one of the earliest neural architectures proposed able to generate distributional representations of words (Bengio et al., 2000). One crucial aspect of the NNLM is the utilization of distributed word representations. The NNLM is trained to maximize the probability of the next word in the training data given the preceding context. This is done by minimizing the negative log-likelihood loss function, also known as cross-entropy loss. Instead of representing words as discrete symbols, the model represents them as continuous-valued vectors. This enables the model to capture semantic and syntactic similarities between words and generalize better to unseen word combinations. The architecture of NNLM is a three-layer neural network and consists of an input layer, a hidden layer, and an output layer. The input layer represents the word sequence, commonly encoded as a fixed-length vector using techniques such as one-hot encoding or word embeddings. The hidden layers contain nonlinear activation functions, such as sigmoid or hyperbolic tangent, which enable the network to learn complex patterns in the data. The output layer provides the predicted probability distribution over the vocabulary.

However, one of the most popular models for learning word embeddings is *word2vec* (Mikolov et al., 2013a,b). The model is implemented in two different algorithms: CBOW (Continuous Bag of Words) and SGNS (Skip-Gram with Negative Samplings). The general model is described by the Equation 3.3

$$p(b|a) = \frac{exp(\mathbf{b} \cdot \mathbf{a})}{\sum_{b' \in C}^{C} exp(\mathbf{b'} \cdot \mathbf{a'})} \tag{3.3}$$

where C is the set of context words, and $\mathbf{a}$ and $\mathbf{b}$ are the embeddings for the target and the context words.

CBOW aims to predict a target word based on its surrounding context words within a given window. SGNS faces the problem from an opposite perspective and aims to predict the context words given a target word.

The architecture of CBOW uses a shallow neural network with an input layer, a hidden layer, and an output layer, similar to the NNLM one (Bengio et al., 2000). The CBOW algorithm learns word representations by optimizing the prediction of a target word from its context. The input layer takes n-dimensional embeddings for the words in the context. Then, the hidden layer (also named the projection layer) calculates the average of the input word vectors. This layer sums up all the input vectors and divides the result by the window size, yielding a single vector representation of the context. Such representation is used to predict the target word. The model is trained using backpropagation, minimizing the cross-entropy loss between the predicted target word and the true target word.

The SGNS algorithm starts from the target word, and its goal is to maximize the probability of predicting the context words. It learns distributed word representations (word embeddings) in a similar manner but with a different training approach. The target word vector is fed into the hidden layer, and the output layer predicts the context words using a softmax activation function. Again, the model is trained using backpropagation. The weights of the hidden and output layers are updated iteratively through gradient descent.

A visual representation of both architectures is shown in Figure 3.1.

One of the primary limitations of Word2Vec algorithms is that the embeddings are solely trained on the local context of words, neglecting the global corpus statistics during training. A proposed solution is Global Vectors for Word Representation (GloVe) (Pennington et al., 2014). Instead of relying only on local context windows, GloVe considers the overall word co-occurrence patterns across the entire corpus.

GloVe begins by creating a word co-occurrence matrix from the input corpus. This matrix represents the frequency of word co-occurrences, capturing how frequently two words appear together in a given context window. The co-occurrence matrix offers a measure of the statistical relationship between words. GloVe subsequently establishes a ratio of word probabilities based on the co-occurrence ma-

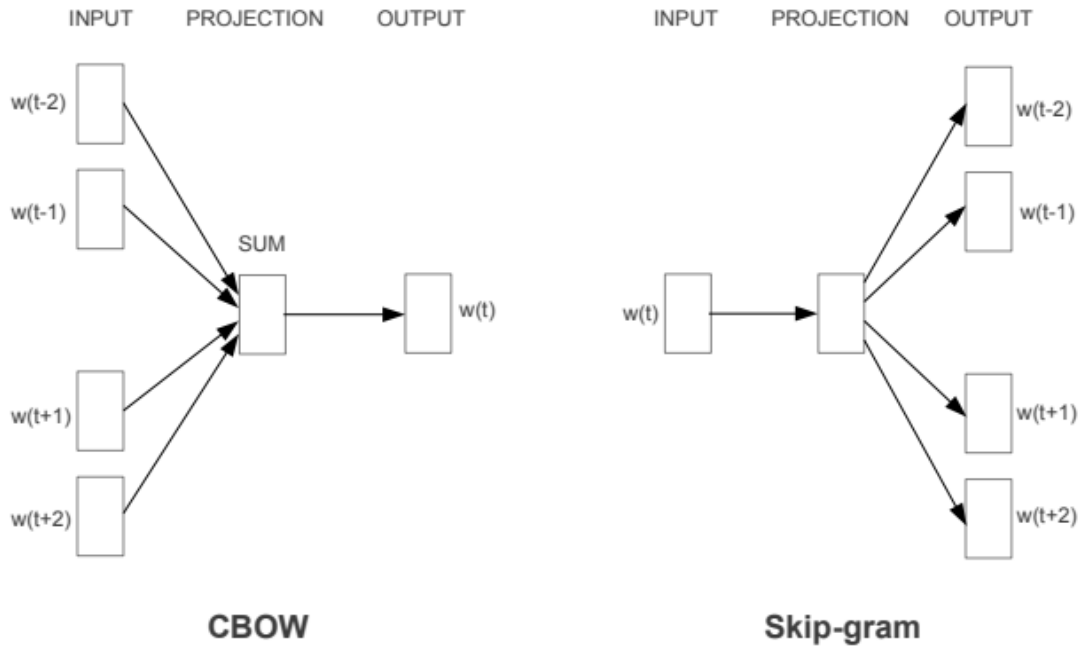

Figure 3.1: CBOW and SGNS architectures as proposed in Mikolov et al. (2013a)

trix. The relationship between two words is established as the ratio between the co-occurrence of such words with a set of probe words.

Given two words i and j, and a probe word k co-occurring with both of the words, the ratio of $\frac{P_{ik}}{P_{jk}}$ is particularly small if k is related to j but not to i; instead it is particularly high if k is related to i but not to j; finally it is close to 1 if k is related or unrelated to both. Authors propose a log-linear function to approximate the relationship between words:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_j = log\left(X_{ij}\right) \tag{3.4}$$

where X_{ij} is the co-occurrence frequency of i and j, w and $\tilde{w}$ are word vectors and context vectors respectively, and the two b are bias terms.

Finally, a weighted least square regression model with a weighting function $f(X_{ij})$ to learn the embeddings is proposed, such that:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_k + b_i + \tilde{b}_j - log\left(X_{ij}\right)\right)^2 \tag{3.5}$$

with V as the vocabulary. If we consider x_{max} the maximum number of co-occurrences for a ij pair, the empirically chosen weighting function states as follows:

$$f(x) = \begin{cases} (x/x_{max})^{0.75} & if\ x < x_{max} \\ 1 & otherwise \end{cases} \tag{3.6}$$

GloVe embeddings have been widely used in various natural language processing tasks. They have shown effectiveness in capturing both syntactic and semantic information, performing better than CBOW and SGNS, and have been influential in advancing the field of word representation learning.

Models like GloVe and the two versions of Word2Vec present several issues. They rely on a fixed vocabulary determined during the training phase. Words that are not present in the training data are treated as out-of-vocabulary (OOV) words. The most popular model presented to address this issue is fastText (Bojanowski et al., 2017). Authors propose the use of sub-word information for learning embeddings in a Skip-gram model.

Furthermore, such models struggle to capture the multiple senses of polysemous words (words with multiple meanings) and the distinction between homonyms (words with the same form but different meanings). Word embeddings tend to aggregate the various senses into a single representation, limiting their ability to capture fine-grained semantic differences. Also, they operate at the word level and do not naturally capture the meaning of phrases or named entities. While word embeddings can provide contextually similar representations for individual words, they may not effectively capture the compositional semantics of longer phrases or entities. Finally, these models generate static word embeddings that do not capture changes in word meaning or usage over time. Language is dynamic, and word meanings evolve. Therefore, these models may not adequately capture the temporal dynamics and semantic shifts that occur in language.

Context aware Language models

The final output of all the previously mentioned models and algorithms is a set of pre-trained word embeddings based on the model hyper-parameters and the training data, or the trained network itself. These embeddings can be subsequently used to obtain representations for words. However, the relationship between embeddings and words (considered as types) is biunivocal: for each word type, one and only one embedding is available, and vice versa.

Context-aware models consider the broader context, including preceding sentences, paragraphs, or even entire documents. This is also crucial because most NLP downstream tasks actually benefit from the understanding of both words and their contexts (Wang et al., 2020). By leveraging this contextual understanding, these models can generate more accurate and contextually appropriate language, leading to improved performance in tasks such as language translation, sentiment analysis,

question answering, and text generation. Through the use of architectures such as Transformer-based (Vaswani et al., 2017) models like GPT (Generative Pre-trained Transformer) (OpenAI, 2023) or BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018), context-aware language models have gained significant attention in recent years.

Often, these architectures are primarily designed for downstream tasks rather than language modelling itself. It is worth noting that such models have also facilitated the adoption of the pre-training and fine-tuning approach for NLP tasks, commonly referred to as transfer learning. In transfer learning, a broad language model trained on unsupervised language comprehension tasks is subsequently fine-tuned to tackle specific tasks. The concept is to preserve the general language understanding acquired during pre-training and effectively transfer it to the specific tasks by further training and adjusting the network's weights and parameters to address the specific objective.

One of the earliest models proposed in literature was ELMo, which stands for "Embeddings from Language Models". It is a context-aware language representation model introduced by Peters et al. (2018). Unlike traditional word embeddings that capture only static word representations, ELMo generates dynamic word representations that take into account the surrounding context (i.e. a function of the entire input sequence). In its architecture, ELMo utilizes a bidirectional language model, specifically two BiLSTM (Bidirectional Long Short-Term Memory) layers, to capture context-dependent word embeddings. One layer reads the sequence from beginning to end, and the other one reads it backward.

LSTM (Sak et al., 2014) stands for Long Short-Term Memory and is a type of recurrent neural network (RNN) architecture designed to capture long-term dependencies in sequential data. It consists of memory cells that allow the network to store and access information over long sequences. LSTMs exploit gating mechanisms to regulate the flow of information through memory cells. They are able to capture and maintain relevant information while discarding unnecessary or irrelevant one.

A Bidirectional Long Short-Term Memory (BiLSTM) is an extension of the LSTM architecture that incorporates both forward and backward information flow to capture context from both preceding and succeeding elements in a sequence. It consists of two LSTM layers. The first one processes the sequence in the forward direction and the second one does it in the backward direction. Each layer independently updates the hidden states based on the input sequence. The output is then obtained by applying some combination mechanism, such as for example by concatenating the forward and backward hidden states.

ELMo incorporates two BiLSTM layers in its architecture. Authors call the model BiLM.

After the input is passed through the two LSTM layers, the contextual word rep-

resentations are obtained by combining the hidden states from the two BiLSTM layers. This fusion of forward and backward representations captures the context from both preceding and succeeding words, providing a rich context-aware representation for each word in the input sequence. Authors report state-of-the-art performances on six supervised NLP tasks by using pre-trained ELMo models, such as sentiment analysis, named entity recognition, and question answering.

Transformer architectures

The introduction of the Transformer architecture (Vaswani et al., 2017) has revolutionized various NLP tasks. Transformers are a type of neural network for sequence transduction that relies on a self-attention mechanism. They allow a model to selectively focus on different parts of the input sequence while generating contextual representations. The attention mechanism calculates attention scores between each element in the input sequence, enabling the model to assign different weights or importance to different elements based on their relevance. This attention-based weighting mechanism allows the model to capture dependencies and relationships between words or tokens, regardless of their relative positions in the sequence.

In Vaswani et al. (2017), authors propose a model that relies solely on the attention mechanism without any recurrent or convolutional components, achieving state-of-the-art results on various language translation benchmarks. The model follows an encoder-decoder architecture. The input sequence is passed through an encoder to obtain contextualized representations, and these representations are then fed into a decoder to generate the output sequence. The core is the self-attention mechanism. It allows each position in the sequence to attend to all other positions, capturing the dependencies and relationships between them. Self-attention is calculated using query, key, and value vectors obtained from the previous layer's representations. The model employs multiple parallel attention layers, known as *attention heads* (i.e. the model implements a multi-head self-attention mechanism to map input and output values). Each head attends to a different subspace of the input representations, enabling the model to capture different types of information. The outputs from different attention heads are concatenated and linearly transformed to obtain the final attention output.

A visual representation of the transformer-model architecture as described in Vaswani et al. (2017) is shown in Figure 3.2.

The Transformer model achieved impressive results on machine translation tasks, overcoming the performance of previous recurrent and convolutional architectures. It demonstrated the power of self-attention and inspired a vast number of researchers to focus on other aspects of language modelling. The success of the so-called *pre-train and fine-tune paradigm* comes from the introduction of GPT (Generative Pre-

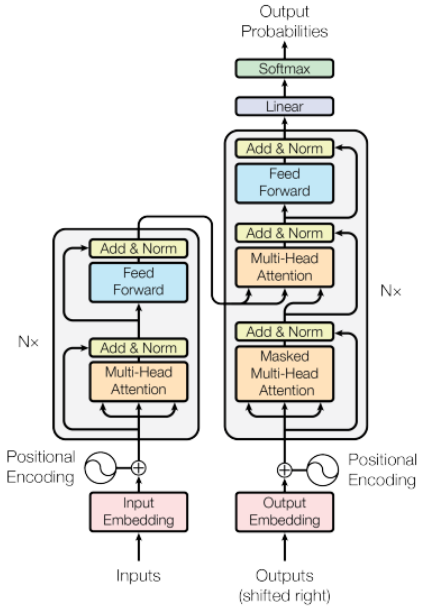

Figure 3.2: The Transformer - model architecture in Vaswani et al. (2017)

trained Transformer) (Radford et al., 2018), an architecture based on the decoder part only of the Transformer architecture (Liu et al., 2018).

GPT employs a pre-training phase where it is trained on a large corpus of unlabeled text data. This *unsupervised* pre-training helps the model to learn general language features. The objective is to predict the next word in a sentence given the preceding context, as in standard language modeling. After pre-training, GPT undergoes a fine-tuning phase where it is further trained on specific downstream tasks. By fine-tuning task-specific labelled data, GPT can adapt its learned representations to perform well on various NLP tasks, such as text classification, sentiment analysis, and question answering. In simple terms, the fine-tuning phase consists of adding a linear output layer on the top of the transformer and updating the weights of the entire network based on a *supervised* learning task.

The next step on the pre-train and fine-tune paradigm was BERT (Bidirectional Encoder Representation for Transformers) (Devlin et al., 2018), which is the starting point of the *BERT Era* in NLP. In fact, GPT's attention is limited to previous tokens in the sequence, which can restrict its performance in sequence-level tasks. On the other hand, BERT employs bidirectional representation learning, utilizing the Masked Language Model and next-sentence prediction tasks to gain a comprehensive understanding of the context. In the Masked Language Model (MLM) task, a certain percentage of the input tokens are randomly masked or replaced with a [MASK] token. The model is then trained to predict the original masked tokens based on the surrounding context. This task encourages the model to learn bidi-

rectional representations as it needs to rely on both the left and right contexts to accurately fill in the masked tokens (Devlin et al., 2018). The Next Sentence Prediction task (NSP) is designed to help the model understand relationships between sentences. In this task, pairs of sentences are created, and the model is trained to predict whether the second sentence follows the first sentence or not. The model learns to capture the relationships and coherence between sentences, which is crucial for tasks like question answering or natural language inference (Devlin et al., 2018). The authors proposed two versions of the architecture, the bert-base, and the bert-large. They differ in the number of layers, parameters, and resulting hidden representations.

The success of BERT has also influenced subsequent models, leading to novel advancements in language understanding and improving the state-of-the-art in many NLP applications. Several models were presented based on distillation (Sanh et al., 2019), a technique for compressing the BERT knowledge in smaller and less expensive models. Examples are DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019). Some other models have been proposed to modify some aspects of BERT to improve its performance while not increasing the computational cost, for example, RoBERTa (Liu et al., 2019). Also, a whole line of research is dedicated to exploring the performance of increasingly bigger models, with orders of magnitude more parameters than the original BERT, such as Transformer-XL (Yang et al., 2019) and XLNet (Dai et al., 2019). Transformer-XL addresses the limitation of the vanilla Transformer architecture in handling long-range dependencies. By introducing a segment-level recurrence mechanism, Transformer-XL enables the model to capture longer-term dependencies more efficiently. This model is particularly effective in tasks that require an understanding of longer context, such as document-level language modeling and machine translation. XLNet is a model that takes the idea of bidirectional context from BERT and combines it with the autoregressive approach of language modelling. Unlike traditional autoregressive models, XLNet considers all possible permutations of the input sequence, allowing each position to attend to both the left and the right context. This helps capture dependencies more comprehensively and improves the quality of the generated text.

Several generations of GPT (Radford et al., 2018, 2019; Brown et al., 2020) show how, with a substantial increase in the number of parameters, pre-trained Transformer models can achieve few-shot and even one-shot learning capabilities without fine-tuning.

Sentences and documents

The advancements in language modelling and word embedding models have marked a significant milestone in the field of NLP, and have expanded the encoding capabilities beyond singular words to include more extensive textual units such as sentences,

paragraphs, and even entire documents. Exploring this direction has the potential to provide significant benefits across various tasks, including text clustering, information retrieval, information extraction, and unsupervised techniques that rely on the semantics of entire text sequences. The aim is parallel to that of word embeddings: representing text units of varying lengths in an n-dimensional space.

The earliest approaches to documents were initially introduced in the information retrieval literature, preceding the discussion about words (Harris, 1954; Salton et al., 1975). At first instance, documents are represented as bag-of-words.

Given a pre-defined vocabulary consisting of n words, documents are represented as n-dimensional vectors. Each dimension corresponds to a word in the vocabulary and signifies the presence or absence of the word in the document. Initially, only the presence of the word is considered. Alternative approaches have suggested the use of raw frequencies, but, similar to word embeddings, raw frequencies have shown to be unreliable measures for document-wise vectors. Several term weighting techniques have been introduced, such as the tf-idf (term frequency-inverse document frequency), a metric used to assess the significance of a word within a document across a collection or corpus (Salton and Buckley, 1988).

With the advent of artificial intelligence and the emergence of neural network language models (Bengio et al., 2000) like word2vec (Mikolov et al., 2013b), the processing of sentences and entire documents has shifted towards the development of similar models capable of encoding documents. The concept behind word2vec was extended to n-gram embeddings, including bi-grams and tri-grams, to capture more contextual information (Mikolov et al., 2013a). However, this approach had limitations in terms of generalization to unseen sentences and the representations of sequences longer than two or three tokens.

The initial effort to extend the word2vec algorithms to longer sequences is illustrated by doc2vec (Le and Mikolov, 2014). Similar to word2vec, doc2vec utilizes neural networks for training. The authors introduce two algorithms, PV-DM (Paragraph Vector - Distributed Memory) and PV-DBOW (Paragraph Vector - Distributed Bag of Words), which aim to incorporate document information into the Skip-gram and CBOW algorithms, respectively. In the PV-DM model, the document vectors are trained to predict the target words within the context of the document. The document vector is combined with the word vectors during the prediction phase. On the other hand, the PV-DBOW model treats the document as a context and predicts the words within the document directly. Figure 3.3 shows the architecture of Doc2Vec as described in Le and Mikolov (2014).

By incorporating document-level information, Doc2Vec enables the representation of documents as dense vectors, allowing for various downstream tasks such as document classification, document similarity, and information retrieval (Le and Mikolov, 2014).

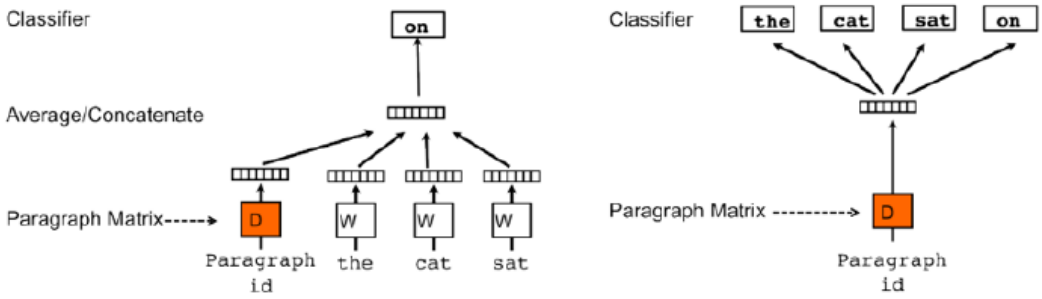

Figure 3.3: The Doc2Vec architecture in Le and Mikolov (2014)

Deep learning-based models have then emerged as robust methods in tasks related to sentence representation and similarity, frequently achieving state-of-the-art performance levels. These models aim to acquire representations from both annotated and unannotated data, enabling them to capture the semantic essence of sentences. The idea of leveraging labeled and parallel corpora to learn sentence embeddings was initially proposed in the context of machine translation. Several works (Cho et al., 2014; Sutskever et al., 2014) employed encoder-decoder architectures to learn sentence embeddings from labelled parallel corpora for translation tasks. Then, the approach has also been extended to monolingual tasks, such as learning paraphrases (Wieting et al., 2015), question answering (Das et al., 2016), and other Natural Language Inference tasks (Conneau et al., 2017; Nicosia and Moschitti, 2017).

The utilization of Transformers has also found extensive application in these tasks and has delivered state-of-the-art outcomes. Transformers can be utilized for downstream tasks and feature extraction, generating word embeddings for each element in the input sequence. Usually, the representation of the entire sequence is encoded with special tokens placed at either the beginning of the sequence or the end. However, it's worth recognizing that sequence representations alone might not be tailored to encapsulate semantically significant information about the sequence itself, and may not be suitable for sentence-pair similarity tasks (Devlin et al., 2018). To overcome these limitations Sentence-BERT (Reimers and Gurevych, 2019) was proposed. It is one of the most widely exploited architectures for learning sentence-wise representation. Sentence-BERT is a modification of the pre-trained BERT network with siamese and triplet network structures. It can produce semantically meaningful sentence embeddings and that can be compared using a similarity measure (for example, cosine similarity or Manhattan/Euclidean distance). The effort for finding the most similar pair is reduced from 65 hours with BERT/RoBERTa to about 5 seconds with Sentence-BERT while maintaining the accuracy achieved by BERT (Reimers and Gurevych, 2019). Efforts have been recently spent to build

an unsupervised contrastive learning method that converts pre-trained language models into universal text encoders, such as with Mirror-BERT (Liu et al., 2021b) and subsequent models (Liu et al., 2021a).

Prompting era

If the introduction of Transformer architectures could be regarded as an initial revolutionary shift in the landscape of the learning of NLP models, the current era, starting in 2021, marks a second significant transformation. In this phase, the *pre-train and fine-tune approach* is going to be replaced or parallelized by the *pre-train, prompt and predict paradigm* (Liu et al., 2023a).

In this paradigm, the process involves not adjusting pre-trained Language Models to suit downstream tasks through specific objectives, but rather reformulating the downstream tasks to look more like those solved during the initial LM training, with the help of textual prompts. For instance, when recognizing the emotion of a social media post, "I missed the bus today," we may follow with a prompt such as "I felt so " and ask the LM to complete the sentence with an emotion-associated term. Alternatively, by employing the prompt "English: I missed the bus today. French: ", an LM could be guided to fill the blank with a French translation (Liu et al., 2023a).

In this way, by choosing suitable prompts we can manipulate the model's behavior so that the pre-trained LM itself can be used to predict the desired output, without requiring any additional task-specific training. The strength of this approach lies in the fact that, given a suite of appropriate prompts, a single LM trained in a fully unsupervised manner, can be used to solve a great number of tasks. Nonetheless, this method introduces the necessity for prompt engineering, finding the most appropriate prompt to allow an LM to solve the task at hand.

The most recent language models successfully apply the latest and most advanced prompting engineering techniques. Models include OpenAI's GPT (OpenAI, 2023), Google's LLaMA (Touvron et al., 2023), and HuggingFace's BLOOM (Scao et al., 2022). In general, large language models typically outperform smaller counterparts, and their zero-shot learning capabilities and emergent new abilities are recognized. However, it's essential to note two significant limitations. Firstly, many of these models are controlled by private companies and are only accessible via APIs. Secondly, the computational demands of such models often pose challenges for running them on standard commercial hardware without resorting to parameter selection and/or distillation techniques.

The following is a brief dissertation on the most recent advanced prompting engineering techniques:

Zero-shot prompting pertains to the approach of employing a prompt to guide a language model in producing intended outputs for tasks which it has not been

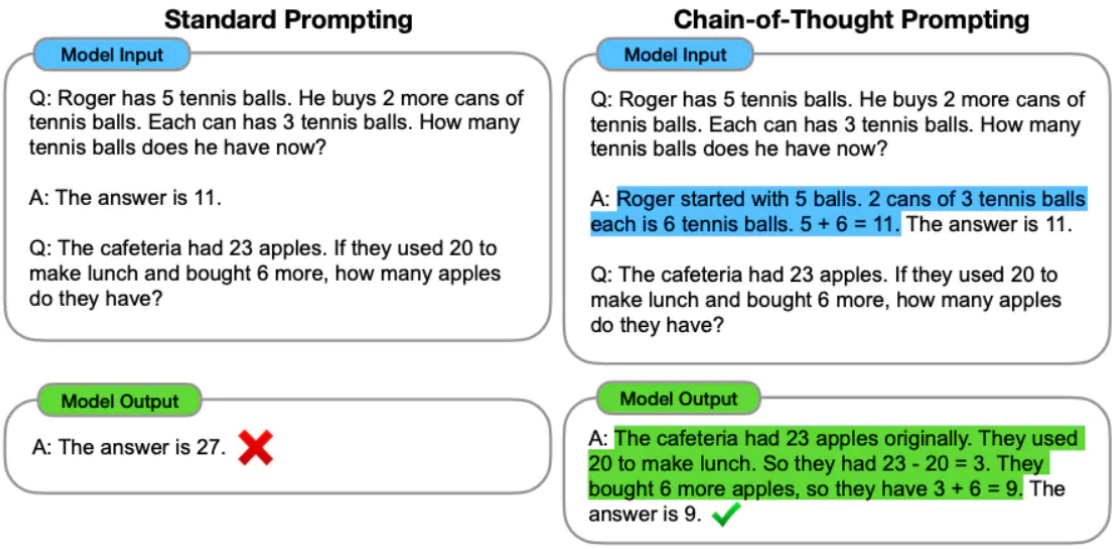

Figure 3.4: A comparison between an example of zero-shot prompt and an example of chain-of-thought prompt, from Wei et al. (2022)

directly trained for. (Wei et al., 2021). Instruction tuning has been shown to improve zero-shot learning (Wei et al., 2021).

Few-shot prompting refers to the technique of using a limited amount of labeled examples or prompts to fine-tune a language model for specific tasks, enabling it to perform well with minimal training data (Brown et al., 2020). It is useful because although large language models exhibit impressive zero-shot capabilities, they may struggle with more complex tasks under the zero-shot setting. Few-shot prompting facilitates in-context learning by incorporating prompt demonstrations to guide the model toward improved performance. While standard few-shot prompting is generally effective for a wide range of tasks, it still has some limitations, particularly when it comes to handling more complex reasoning tasks (Brown et al., 2020).

Chain-of-thought (CoT) prompting refers to a technique where a series of related prompts or instructions are provided to a language model to guide it through a coherent and connected line of thinking, allowing it to generate contextually consistent and coherent responses. By combining it with few-shot prompting, the performance could be enhanced on more complex tasks that demand reasoning before generating responses, thereby achieving improved results (Wei et al., 2022).

Figure 3.4 shows a comparison between an example of a zero-shot prompt and an example of a chain-of-thought prompt, from Wei et al. (2022).

Several improvements of the CoT prompting have been proposed, such as zero-shot CoT (Kojima et al., 2022), and Auto-CoT (Zhang et al., 2022)

Self-consistency is an advanced technique of prompt engineering (Wang et al.,

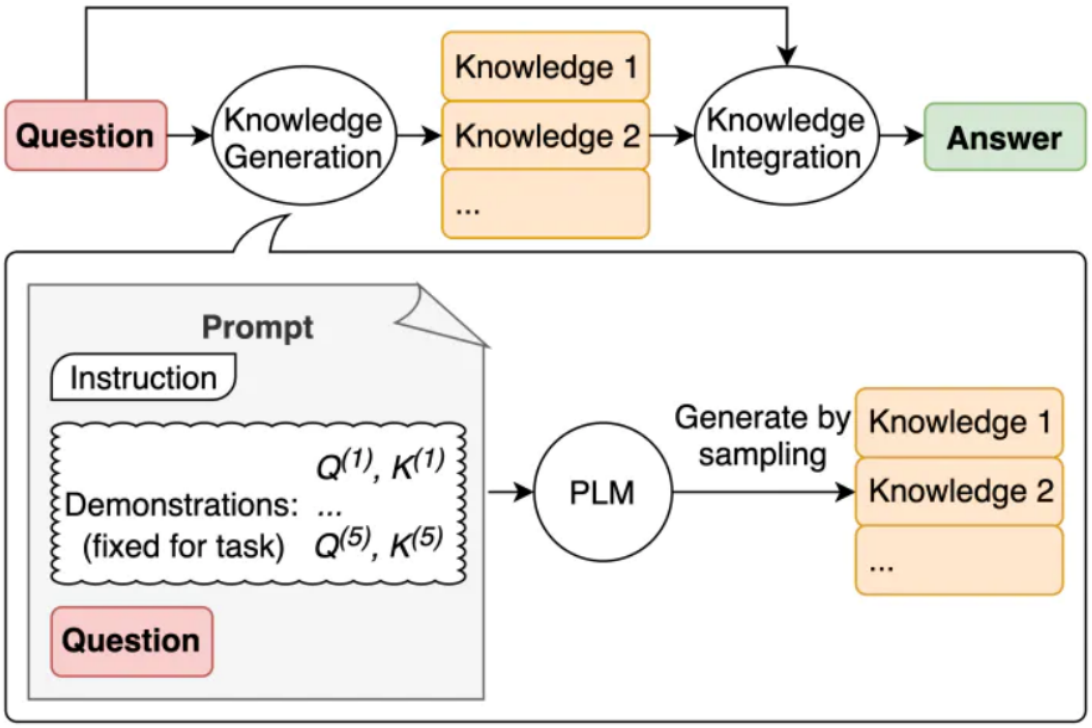

Figure 3.5: A visual representation of the Generate Knowledge Prompting process, from Liu et al. (2021c)

2022). Instead of the simplistic greedy decoding employed in chain-of-thought prompting, self-consistency focuses on sampling diverse reasoning paths through few-shot CoT. The generated responses are then used to select the most consistent answer. This helps to boost the performance of CoT prompting on tasks involving arithmetic and commonsense reasoning.

An interesting and advanced prompting technique is the so-called **Generate Knowledge Prompting** (Liu et al., 2021c). The model is used to generate knowledge before making a prediction, as part of the prompt itself.

A visual representation of this idea is shown in Figure 3.5.

Tree of Thoughts (Yao et al., 2023) is a structured framework where prompts are organized in a hierarchical tree-like structure. The framework is shown in Figure 3.6. Each node in the tree represents a specific prompt or instruction, guiding the language model's generation process. This technique allows for more structured and coherent responses by enabling the model to navigate and expand upon different branches of thought within the tree.

For complex and knowledge-intensive tasks, a language model-based system that incorporates external knowledge sources can be built. This approach improves factual consistency, and enhances the reliability of generated responses.

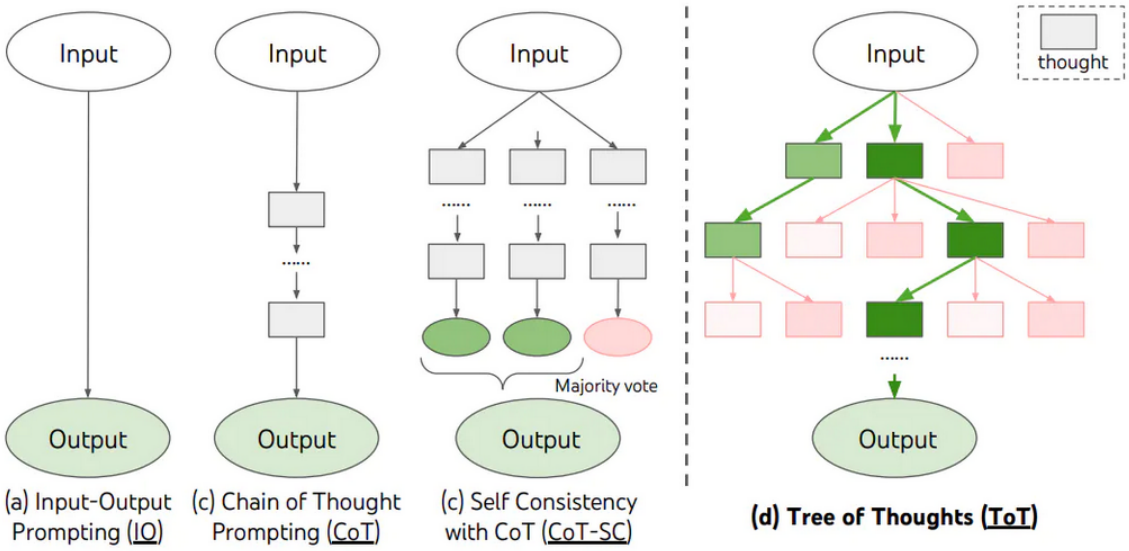

Figure 3.6: The Tree of Thoughts framework, from Yao et al. (2023)

To tackle knowledge-intensive tasks, meta AI researchers introduced **Retrieval Augmented Generation** (RAG) (Lewis et al., 2020). RAG combines an information retrieval component with a text generator model. It allows for fine-tuning the model and modification of its internal knowledge efficiently, without the need for retraining the entire model.

The RAG procedure is exemplified in Figure 3.7.

RAG takes an input and retrieves a set of relevant/supporting documents from a source such as Wikipedia. These documents are concatenated as context with the original prompt and fed into the text generator, producing the final output. This adaptability enables RAG to handle situations where facts may evolve over time, which is particularly valuable as the parametric knowledge of language models is static. By leveraging retrieval-based generation, RAG allows language models to access the latest information without requiring retraining, ensuring reliable outputs.

A general-purpose fine-tuning recipe for RAG has been proposed (Lewis et al., 2020). The approach utilizes a pre-trained seq2seq model as the parametric memory and a dense vector index of Wikipedia as the non-parametric memory, accessed through a neural pre-trained retriever.

RAG is demonstrated to perform strongly on several benchmarks such as Natural Questions, WebQuestions, and CuratedTrec (Lewis et al., 2020).

The combination of CoT prompting and interleaved tools has proven to be a powerful and reliable approach for tackling various tasks using LLMs. These approaches often involve creating task-specific demonstrations and strategically interleaving the model's generated outputs with the utilization of tools.

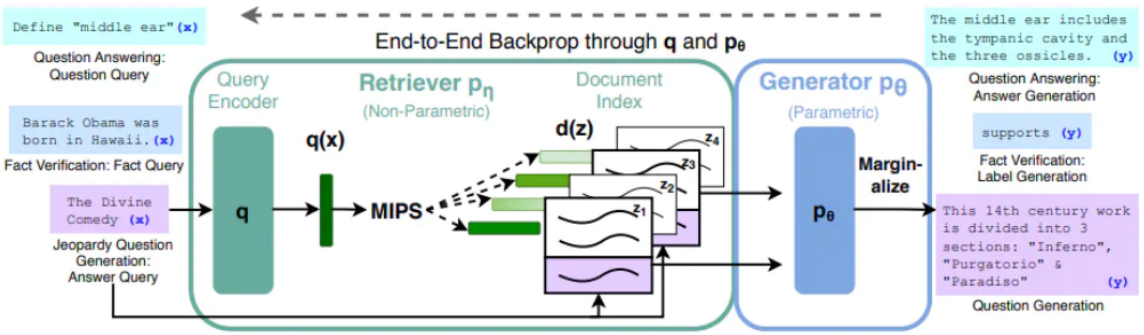

Figure 3.7: Overview of the RAG procedure, from Lewis et al. (2020)

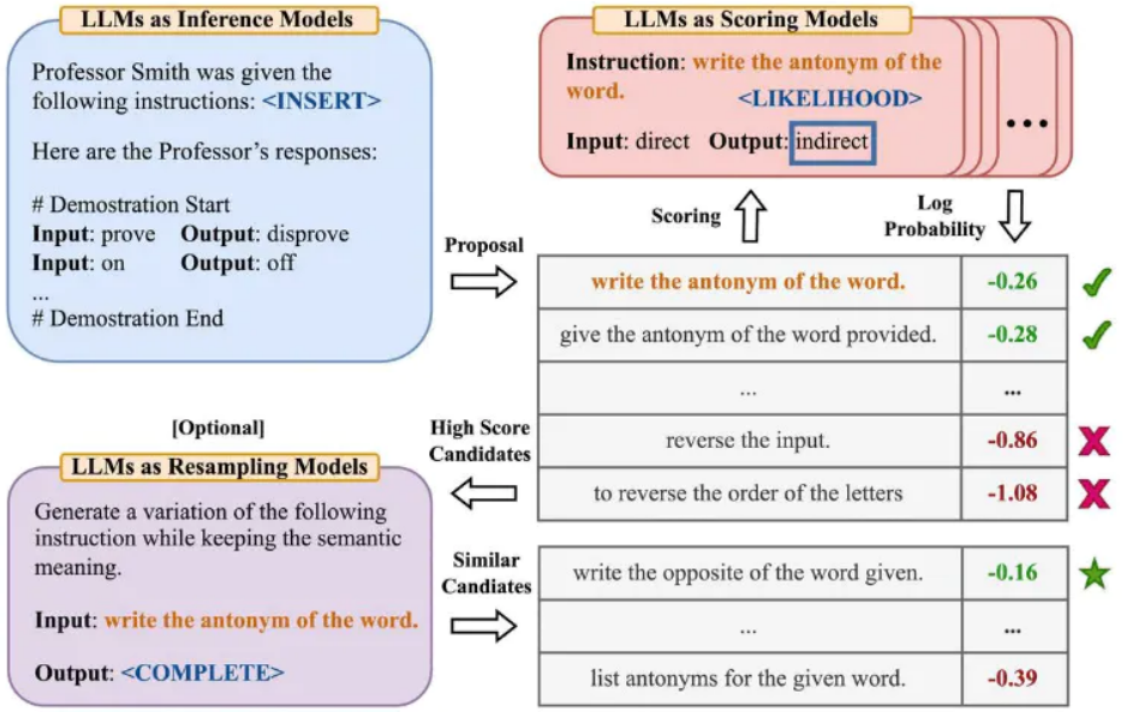

Figure 3.8: Overview of the APE procedure, from Zhou et al. (2022)

A framework that uses a frozen LLM to automatically generate intermediate reasoning steps, called **Automatic Reasoning and Tool-use** is proposed in Paranjape et al. (2023).

Automatic Prompt Engineer (APE) is, instead, a framework for automatic instruction generation and selection (Zhou et al., 2022). An overview of how it works is given in Figure 3.8

The initial stage entails utilizing a large language model as an inference model, which receives output demonstrations to generate potential instructions for a given task. These instruction candidates serve as guides for the subsequent search process.

The instructions are executed by a target model, and based on computed evaluation scores, the most suitable instruction is selected.

ReAct is a framework where LLMs are used to generate both reasoning traces and task-specific actions in an interleaved manner (Yao et al., 2022).

By generating reasoning traces, the model gains the ability to deduce, monitor, and update action plans, as well as handle exceptional cases. The action step facilitates interaction with external sources like knowledge bases or environments to gather relevant information.

The ReAct framework empowers Large Language Models (LLMs) to engage with external tools for retrieving additional information, resulting in more reliable and factually accurate responses.

Experimental results demonstrate that ReAct surpasses several state-of-the-art baselines in language and decision-making tasks (Yao et al., 2022). Furthermore, ReAct enhances the interpretability and trustworthiness of LLMs from a human perspective. The authors conclude that the optimal approach combines ReAct with the chain-of-thought (CoT) technique, enabling the utilization of both internal knowledge and external information acquired during reasoning (Yao et al., 2022).

There are several other advanced techniques, and the field is in growth. Other examples are the Active-Prompt approach (Diao et al., 2023), Directional Stimulus Prompting (Li et al., 2023), Multimodal CoT Prompting (Zhang et al., 2023), and Graph Prompting (Liu et al., 2023b).

A special mention should be given to the approach known as **Prompt Tuning** (Lester et al., 2021). It is a technique designed to improve the performance and versatility of LLMs on various downstream tasks. Prompt Tuning involves training a separate vector of tokens called "soft prompts" to guide the frozen language model's behavior and output generation. One of the key advantages of Prompt Tuning is its flexibility and adaptability. By modifying the prompts, researchers can easily customize the model's behavior for different tasks without the need for extensive retraining. This allows for quick task adaptation and the ability to address a wide range of NLP tasks with the same pre-trained model. Prompt Tuning has been shown to be effective in improving the performance of language models across various domains and tasks. It provides a mechanism to inject task-specific knowledge into the model, enabling a better understanding of context and more accurate response generation. Overall, Prompt Tuning enhances the versatility and performance of large language models by leveraging prompts as external guidance, resulting in improved task-specific capabilities and more tailored outputs.

Challenges in the present work

In the present work, we address the problems of textual and semantic similarity from several perspectives.

In the first case study, we focus on newspaper articles, specifically at the sentence level. Sentence-BERT is employed to acquire appropriate representations of the sentences, facilitating a comprehensive comparison between sentences extracted from multiple newspaper articles to highlight their differences. Our approach involves leveraging semantic sentence similarity and association rules, introducing a novel perspective denoted as *information divergence*.

The challenges inherent in semantic and textual similarity, as well as those associated with Large Language Models, are further explored in our third case study, where we delve into the realm of Fake News Detection problem. We both exploit Language Models and incorporate modern techniques such as prompt tuning. Notably, we also seek to integrate the information divergence methodology previously developed in the first case study.

3.2 A brief introduction to Sequential Pattern Mining and Frequent Episode Mining

The ever-increasing availability of data and new data sources is at the basis of the growing popularity of fields such as data mining. In this context, *process mining* is a family of techniques that emerged in recent years for the analysis of processes based on event logs (Van Der Aalst, 2012). Process mining aims to generate, out of the observation of process logs, factual knowledge possibly useful in several application contexts, supporting performance monitoring tasks, improved exploitation of the available resources, and, in particular, the automation of processes themselves.

The input data of a process mining application is called an *event log*, (i.e., a collection of chronologically ordered records of events produced by the execution of a process). Each event in the log refers to: i) a specific process instance, ii) an activity in such a process, along with related information, and ii) a timestamp. Event logs can be pre-processed to have the essential information as formal sequences of events (Marin-Castro and Tello-Leal, 2021).

From a practical standpoint, and a more general perspective, the goal of a process mining algorithm is the identification, within a dataset (e.g. a number of event logs), of particular sequences that show up a statistically significant number of times, to verify particular potentially interesting recurrences (Han et al., 2007). The problem of the identification of such sequences has been explored in the literature under different lights and different domains, leading to the classic research problems known as Frequent Itemset Mining (FIM) and Sequential Pattern Mining (SPM), along with various other derived sub-problems, such as the High Utility Sequential Pattern Mining (HUSPM) and Frequent Episode Mining (FEM). The latter is particularly interesting for the purpose of this work.

A pattern mining approach can be applied for data analysis in a large number of applications, especially in data explorations that cannot take advantage of prior knowledge of the target problem and, we can argue, could be an interesting way to analyze from a different perspective natural language and textual data.

Foundamentals of Sequential Pattern Mining

SPM is a branch of data mining that deals with discovering statistically significant patterns among data samples where the values are presented in ordered sequences. SPM is based on FIM, which was initially introduced in Market Basket Analysis to identify sets of products that are frequently purchased together (Agrawal et al., 1993). FIM does not explicitly consider the temporal ordering of data to be analyzed. It is defined in the following.

Consider a *transactional* database D, which consists of a set of transactions $\{t_i\}$. A single transaction is a basic record in the database. Any transaction is typically identified by a Transaction ID (TID) and corresponds to a set of objects out of a set of possible items $I = \{i_1, i_2, ..., i_n\}$. For example, using as an example the sales in a shop, each transaction is a receipt, relative to the items purchased by a particular customer. In this case, I is the set of all the items that the shop sold. A subset $X \subseteq I$, containing k items, is called an *itemset*. It should be noted that no constraint applies to the data types of the elements in the set I, which can be also heterogeneous. In a practical way, several algorithms ask to operate over I and to scan the relative itemsets according to a total order over their members. The support of an itemset X in the dataset D is determined by the proportion of transactions in D that contain that itemset, often denoted as $supp(X, D)$ (or simply $supp(X)$ if D is understood from context). To be considered as frequent in D and be of interest, an itemset X must have a support higher than a predefined minimum support threshold denoted as *minsupp*. Therefore, an itemset X is considered frequent if $supp(X) \geq minsupp$ in the dataset D. The support value, and consequently the minimum support threshold, typically ranges between 0 and 1. In FIM algorithms, users can specify the actual *minsupp* value, allowing them to include more itemsets (with a lower *minsupp*) or fewer itemsets (with a higher *minsupp*) as frequent. The main objective of FIM is to identify all itemsets in a given transaction database that meet or exceed the minimum support threshold *minsupp*.

Therefore, an itemset X is considered frequent if $supp(X) \geq minsupp$ in the dataset D. The support value, and consequently the minimum support threshold, typically ranges between 0 and 1. In frequent itemset mining (FIM) algorithms, users can specify the actual *minsupp* value, allowing them to include more itemsets (with a lower *minsupp*) or fewer itemsets (with a higher *minsupp*) as frequent. The main objective of FIM is to identify all itemsets in a given transaction database that meet or exceed the minimum support threshold *minsupp*.

Whenever the temporal ordering of transactions is relevant to the analysis goals, different approaches must be devised. SPM has been proposed as an extension of FIM to handle temporally ordered sequences of itemsets. The problem studied by SPM is defined as follows. A *sequence* database $SeqD = \{S_1, S_2, ..., S_j\}$ is made of *j sequences*, and each of them is supposed to have a different origin. A sequence is identified by a Sequence ID (SID) and it consists of an *ordered list* of elements: in the general case, an element is an itemset (usually, a non-empty one).

Usually, the ordered elements of the sequence (i.e., the itemsets) are referred to as *events*. Given a sequence $S = \langle E_1, E_2, ..., E_n \rangle$, the corresponding length $|S|$ can be defined as the number of events in S. On the basis of the total number of objects it contains, a sequence is referred to as a k-sequence, with $k = \sum_{i=1}^{n} |E_i|$. Consider the example sequence $s = \langle \{paper, pencil\}, \{pencil, sharpener\}, \{ruler\} \rangle$. The sequence contains three events. Thus the sequence length is $|S| = 3$, and it is a 5-sequence containing $k = 2 + 2 + 1 = 5$ items.

The concept of *support* is extended to sequences by the introduction of additional definitions. A sequence s_2 is said to be a *sub-sequence* of the sequence s_1 if it can be derived from s_1 by deleting some objects without changing the order of the remaining objects (Fournier-Viger et al., 2017). Formally, $s_2 = \langle B_1, B_2, ..., B_n \rangle$ is a sub-sequence of $s_1 = \langle A_1, A_2, ..., A_m \rangle$ if and only if there exist n integers $\{h_i\}_{i=1...n}$ with $1 \leq h_1 < h_2 < \cdots < h_n \leq m$ such that $B_1 \subseteq A_{h_1}, B_2 \subseteq A_{h_2}, ...B_n \subseteq A_{h_n}$. This is indicated by the notation $s_2 \sqsubseteq s_1$, and we can say that s_1 contains s_2 or, likewise, s_1 is a *super-sequence* of s_2. This is important because the definition of sub-sequences is helpful to generalize the concept of "support": the support of a sequence s within the database $SeqD$ is the proportion, over the size of $SeqD$, of those sequences that contain s, (i.e. $supp(s, SeqD) = |\{s' \mid s' \in SeqD \wedge s \sqsubseteq s'\}| \ / \ |SeqD|$). In some cases, in the literature, the support is defined as the plain count of super-sequences of S in $SeqD$.

Given a minimum support threshold *minsupp*, a sequence s in a sequential database $SeqD$ is called a *frequent sequential pattern* if $supp(s) \geq minsupp$. The objective of SPM is therefore the identification of the sequential patterns within a given database of sequences (Agrawal and Srikant, 1995).

SPM is a indeed complex and computationally demanding problem. Each k-sequence potentially has 2^k sub-sequences, leading to a massive search space that can be challenging to explore efficiently. SPM algorithms typically address this problem by generating candidate sequences and then identifying their support within the database. Candidate generation is typically done in two main ways: sequence extension (*s-extension*), and itemset extension (*i-extension*). In an s-extension, an existing sequence S_p is extended by adding a new event containing a single item. This approach focuses on extending the sequence by adding new events while keeping the items within the events the same. In an i-extension, a new item is added to the

last event of the existing sequence S_p.

Based on the search approach, there are typically two categories of algorithms for SPM, namely *depth-first* and *breadth-first*. With breadth-first algorithms, the search starts with shorter sequence patterns and progressively explores longer ones. The process begins with identifying sequential patterns of length 1, then moves to patterns of length 2, and so on, up to a predefined maximum length, denoted as k. In depth-first algorithms, the search space uses a prefix tree structure (often called a "trie"). The root of the tree represents single events. The search starts at the root and progressively explores deeper levels of the tree by generating candidate sequences through s-extensions and i-extensions. It continues to expand the search space until no more candidates can be generated or until predefined constraints are met. Finally, the support of all the generated sequences is computed concerning the actual data to detect the frequent sequential patterns (Bechini et al., 2023).

Main algorithms for Sequential Pattern Mining

The literature offers a large number of algorithms and implementations for SPM. Gan et al. (Gan et al., 2019) categorize these algorithms into four distinct groups: *apriori-based*, *pattern-growth*, *hybrid*, and *constraint-based*.

Apriori-based algorithms take advantage of the antimonotonicity of the apriori nature to prune infrequent sequences effectively. Pattern growth algorithms, on the other hand, typically operate by exploiting a projected database constructed based on prefixes, allowing them to count only occurrences of actual patterns within the database. Hybrid algorithms combine elements from both apriori-based and pattern-growth approaches, capitalizing on their respective strengths while mitigating their weaknesses. Finally, constraint-based algorithms focus on solving a more specific problem, which involves identifying frequent patterns that meet specific constraints, regardless of the algorithmic method employed. A visual representation of these algorithm types and their relationships is illustrated in Figure 3.9, extracted from Bechini et al. (2023).

Apriori-based algorithms are based on a general property of sequences known as the *apriori property*, which asserts that “all non-empty subsets of a frequent itemset must also be frequent." In other words, if an itemset is found to be infrequent, all of its supersets (i.e., itemsets containing the mentioned itemset) will also be infrequent. This property offers a valuable opportunity to efficiently constrain the exploration of the search space. Apriori-based algorithms typically employ a breadth-first search strategy to identify sequences within the database.

Apriori-based algorithms can make use of two database formats: the *horizontal* and *vertical* representations, both of which convey the same information in a different way. The horizontal format is the conventional database representation. The

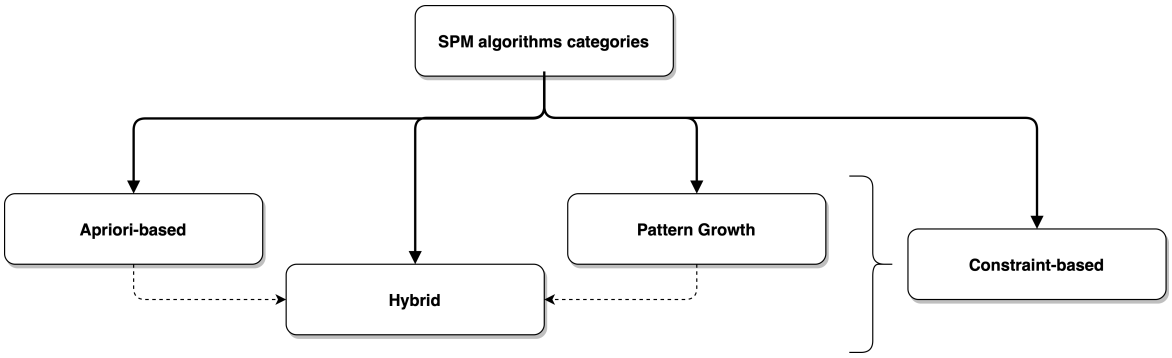

Figure 3.9: Types of algorithms for SPM and their relationships, as shown in Bechini et al. (2023).

vertical representation of a sequence database consists of the *IDList*s for individual items: each IDList corresponds to a specific item and provides information about the sequences in which that item appears and, within those sequences, in which events (itemsets) it is present.

The first apriori algorithms are *AprioriAll*, *AprioriSome*, and *DynamicSome*, which are variants of the approach based on the apriori property proposed by Agrawal and Srikant (Agrawal and Srikant, 1995). The *Generalized Sequential Pattern* algorithm (or GSP (Srikant and Agrawal, 1996)) exploits the apriori property as well, addressing performance improvements over AprioriAll and introducing also three novelties for better identifying interesting frequent sequences: time constraints (minimum and maximum gap between transactions), sliding windows, and taxonomies. The *SPADE* algorithm (Zaki, 2001b) is one of the most commonly employed SPM algorithms that exploit the apriori property. It is designed to minimize the need for multiple scans of the database. SPADE faces this issue by decomposing the problem into sub-problems through a set of combinatorial properties. The ultimate solution can be obtained with no more than three scans through the database. An algorithm closely resembling SPADE in its depth-first approach is the *SPAM* algorithm (Ayres et al., 2002). The LAPIN-SPAM algorithm (Yang and Kitsuregawa, 2005) introduces several improvements to the approach initially presented in SPAM (Ayres et al., 2002). A few years later, the LAPIN algorithm (Yang et al., 2007) was introduced, employing a technique known as *last position induction.* This approach leverages the final position of an object i within the database to determine whether or not a k-sequence can be extended using i.

Pattern Growth algorithms. Most apriori-based techniques often involve identifying patterns through the insertion and deletion of new objects or events from the

shortest patterns and then determining their frequency in the database. This process can be computationally expensive, as it requires generating candidates and counting them throughout the dataset at each iteration. To address this limitation, pattern-growth algorithms were introduced. These algorithms aim to build patterns present in the database incrementally. However, the recursive nature of this approach can also be costly. To mitigate this, algorithms based on *projected databases* have been proposed, which is a compressed representation of the original database containing projected itemsets (Han et al., 2001). The core idea is to assess the frequency of an itemset X by examining the X-projected database, which includes only transactions where X appears. Based on the assumption that if an itemset X is infrequent, then any sequence with a projected itemset that is a superset of X cannot be a sequential pattern, this approach allows for pruning large portions of the database and focusing on the projected sub-databases for further searches.

Pattern growth algorithms typically use a depth-first search of sequences within the database.

The idea of using a projected sequence database was first employed by the *FreeSpan* algorithm (Han et al., 2000). One of the most used pattern-growth algorithms that utilize this concept is *PrefixSPAN* (Han et al., 2001). An improved variant of PrefixSPAN is *PrefixSPAN-x*, which retains the core structure of PrefixSPAN while introducing additional improvements (Fei et al., 2016). For mining frequent biological sequences, such as DNA, the *Depth-First SPelling* (DFSP) algorithm was specifically designed and has demonstrated better performance compared to PrefixSPAN in this context (Liao and Chen, 2014). *FS-Miner* is an algorithm developed for analyzing web logs, which are a relevant target for frequent sequence mining. Similar to FS-Miner, *WAP-Tree* employs a tree structure and requires only two scans of the database. However, during the first scan, it calculates the support only for sequences of length one. Various improvements over WAP-Tree have been proposed in the literature, including PLWAP, a version that eliminates the need for recursive scanning of the tree (Ezeife et al., 2005).

Hybrid algorithms. In addition to Apriori-based and pattern growth algorithms, there are hybrid approaches aimed to combine the advantages of both Apriori-based and pattern growth techniques while mitigating their drawbacks. While pattern-growth algorithms have made improvements in addressing certain limitations that affected simpler Apriori-like approaches, they have also introduced their own challenges. For instance, some algorithms, such as PrefixSpan, can be computationally expensive due to the method used to construct the projected database of sequences. Research on "hybrid algorithms" aims to identify and exploit the strengths of both Apriori and pattern-growth algorithms to optimize the pattern search process.

The HVSM algorithm (first-Horizontal-last-Vertical scanning database Sequen-

tial pattern Mining algorithm) (Song et al., 2005) follows a similar approach to the SPAM algorithm. Experimental results have demonstrated that the HVSM algorithm can search for frequent sequences faster than the SPAM algorithm, particularly when dealing with very large sequence and transaction databases. Disc-All (DIrect Sequence Comparison) (Chiu et al., 2004) enables the mining of frequent sequences without the need to calculate the support of less frequent sequences. UD-DAG (UpDown Directed Acyclic Graph) (Chen, 2009) leverages bidirectional pattern growth by considering both suffixes and prefixes to extract frequent sequences, resulting in improved performance compared to other pattern growth-based algorithms, especially in terms of scalability when dealing with larger values of minsupp and longer patterns.

Constraint-based algorithms. The literature has shown also significant interest in specific problems that require the introduction of constraints for the identification of frequent patterns. Constraint-based Sequential Pattern Mining algorithms (CSPM) are designed to find sets of sequence patterns that satisfy a given constraint, denoted as C. Formally, a constraint C for the SPM task is a predicate, represented as $C(\cdot)$, which determines whether a sequence in the pattern set can be considered acceptable or not. The retrieval process aims to select only those sequences that satisfy the constraint, resulting in a more concise output that ideally contains the most interesting sequences.

Constraints in Sequential Pattern Mining can take various forms, and one common approach is to use regular expressions, as seen in algorithms like SPIRIT (Sequential Pattern Mining with Regular Expression Constraints) (Garofalakis et al., 2002). However, different types of constraints have also been explored (Mooney and Roddick, 2013; Pei et al., 2002). These include constraints on the item type, constraints involving aggregate functions of items, constraints on pattern length, and model-based constraints, such as specifying sub-patterns or super-patterns of a given pattern. In scenarios where events are associated with timestamps, constraints can involve time-related aspects like time span, time differences between adjacent events, and more. Furthermore, specific constraints known as *gap constraints* have been studied. Gap constraints involve specifying the allowed number of events (within a given range) between two adjacent, predefined events in the target pattern (Wu et al., 2017).

In many cases, it is required to meet specific criteria on the number of occurrences a pattern appears within a sequence, while considering any timing constraints that have been imposed. There exist five distinct counting techniques that can be categorized into three groups, as outlined by the authors in (Mooney and Roddick, 2013). The first group consists of the CEVT technique (count event), which involves searching for a specific sequence throughout the entire timeline of the sequence data. The

second group consists of the CWIN (count windows) and CMINWIN (count minimum windows) techniques. They deal with counting the number of windows in which a given sequence occurs. The last group consists of the CDIST (count distinct) and CDIST_O (count distinct with the possibility of overlap) (Mooney and Roddick, 2013).

CSPM algorithms often utilize specific characteristics associated with the imposed constraints. Two common properties related to constraints are *monotonicity* (if $C(S)$ is false, then the same result applies to all sub-sequences of S) or *anti-monotonicity* (if $C(S)$ is false, the same holds for all super-sequences of S). When neither of these properties is satisfied, designing CSPM algorithms becomes more challenging, as discussed in Hosseininasab et al. (2019). CSPM algorithms can be further categorized into several subgroups based on their specific approaches and the constraints they handle.

Closed and Maximal SPM. Frequently, not all sequences in an SPM (Sequential Pattern Mining) set are relevant to the user. It would be reasonable to focus only on a significant subset, considering that if a sequence is frequent, then also its sub-sequences are frequent. One of the main issues of the algorithms presented so far is that they tend to generate redundant patterns, especially with low *minsupp* values or in the presence of pattern-enriched databases (Zhang et al., 2015). To address this issue, researchers have introduced the concepts of Closed Sequential Patterns (CSPs) and Maximal Sequential Patterns (MSPs). A CSP set, denoted as S_{closed}, consists of sequences from the SPM set S_{freq} that have no super-sequences with the same support within it. In mathematical terms (Yan et al., 2003), $S_{closed} = s; |; s \in S_{freq} \wedge \nexists; s' \in S_{freq}; \text{s.t.}; s \sqsubseteq s' \wedge supp(s) = supp(s')$. The CSP set is contained in the SPM set. Similarly, the MSP set, denoted as S_{maxf}, comprises all the *maximally frequent sequences* from the SPM set. These are sequences with no super-sequences within the SPM set (Luo and Chung, 2005). Formally, $S_{maxf} = \{s \mid s \in S_{freq} \wedge \nexists\, s' \in S_{freq} \text{ s.t. } s \sqsubseteq s'\}$. The MPS set is contained in the CPS set.

Yan et al. (Yan et al., 2003) not only introduced the definition of CSP sets but also proposed the *CloSPAN* (Closed Sequential PAtterN mining) algorithm to identify closed sequential patterns. By leveraging the foundational ideas of SPADE (Ayres et al., 2002) and CloSPAN (Yan et al., 2003), the *CLaSP* algorithm (Gomariz et al., 2013) enhances its efficiency by using a vertical data representation and pruning strategies. Similarly, *CloFAST* (Fumarola et al., 2016) adopts a vertical database layout and sparse id-lists to discover closed sequential patterns. The *BIDE* algorithm (BI-Directional Extension based frequent closed sequence mining) (Wang and Han, 2004) effectively prunes the search space while keeping memory requirements low. To address redundancy in results, *CCSpan* (Closed Contiguous Sequential pattern mining) (Zhang et al., 2015) aims to obtain a more compact pattern set without los-

ing any information. More recently, the utilization of a NetTree data structure led to the development of the *NetNCSP* (Nettree for Nonoverlapping Closed Sequential Pattern) algorithm (Wu et al., 2020).

The motivation behind identifying an MSP set is to effectively reduce the result set, making it more comprehensible for end users improving task performance in terms of execution time and memory usage, and facilitating the interpretation of the algorithm outcome. Several specialized algorithms for retrieving maximal sequential patterns have been proposed. For instance, *MSPX* (Luo and Chung, 2005) employs multiple samples to exclude less frequent sequences. *MaxSP* (Maximal Sequential Pattern mining) (Fournier-Viger et al., 2013) computes all maximal sequential patterns without the need to store intermediate candidate sequences in main memory. *VMSP* (Vertical mining of Maximal Sequential Patterns) (Fournier-Viger et al., 2014) is the first vertical algorithm designed for mining maximal sequential patterns, offering state-of-the-art execution time performance.

In addition to closed and maximal sequence patterns, researchers have also explored *generator sequential patterns*, often referred to as GSP patterns (Lo et al., 2008). The elements of the SPM set supported exactly by the same sequences in the database can be considered an equivalence class. All the sequences of such an equivalence class have the same support, and the "$\sqsubseteq$" relationship is a partial order over them. According to the "$\sqsubseteq$" ordering, the set of maximal and minimal patterns within an equivalence class are called *closed patterns* and *generator patterns*, respectively. Therefore, the set of generator sequential patterns, denoted as S_{gen}, comprises sequences from the SPM set that have no subsequence with the same support. Formally, this set is defined as $S_{gen} = s; |; s \in S_{freq} \wedge \nexists; s' \in S_{freq}; \text{s.t.}; s' \sqsubseteq s \wedge supp(s) = supp(s')$. Researchers have proposed algorithms to efficiently discover these generator sequential patterns, recognizing their potential utility in various applications (Lo et al., 2008; Pham, 2015; Le et al., 2017).

Top-k SPM. In some specific cases, where determining an appropriate minimum support threshold in advance is challenging or when such a choice significantly impacts the results, top-k sequential pattern mining methods have been introduced. Instead of providing a final result containing all sequential patterns that meet specific length and minimum support criteria, these algorithms focus on returning only the k most frequent sequential patterns from the dataset, ranked by their relative support (Fournier-Viger et al., 2017). Numerous variations of this approach have been developed over the years. It is important to mention an algorithm based on constrained prefix-projected pattern growth that employs an innovative interestingness measure. This measure calculates the proportion of pattern occurrences that are cohesive (Feremans et al., 2018). The Top-k approach is frequently employed in combination with other techniques for various sequential pattern mining tasks,

including frequent episode mining (Fournier-Viger et al., 2020) and High-Utility mining (Rathore et al., 2016).

Quantitative Sequences. In various applications, it's common to enrich each item within the events of sequences with a *quantitative attribute*. This attribute represents the actual quantity associated with the transaction, allowing for value comparisons. This additional information can be highly valuable for end users. For instance, in sales transactions, having information about the quantities of items and identifying patterns that incorporate quantitative data can be beneficial for designing marketing campaigns and making informed decisions. Extending sequential pattern mining (SPM) algorithms to handle sequences with quantitative attributes is a complex task, requiring proper solutions to obtain more comprehensive and potentially more informative results. Two main variants of traditional algorithms adapted for quantitative SPM include Apriori-QSP *Apriori-QSP*, which relies on Apriori-like algorithms, and Han et al. (2001), which is based on the PrefixSpan algorithm. Additionally, in order to address quantitative sequences in the context of Big Data, *Q-VIPER* algorithm has been developed (Czubryt et al., 2022).

Parallelism in Sequential Pattern Mining

Recent years have witnessed growing accessibility in the availability of extensive and complex datasets, as well as the ability to leverage computational resources across diverse devices. This trend has spurred an increased interest in Parallel Sequential Pattern Mining (PSPM) research (Gan et al., 2019). The need to apply SPM techniques to big data has prompted researchers to explore methods that can analyze and uncover sequential patterns in a parallel fashion. The MapReduce programming model, initially introduced in Hadoop, has emerged as a pivotal approach for examining massive datasets on distributed platforms (Tsai et al., 2016; Segatori et al., 2018). This model has played a central role in various parallel SPM initiatives. On the other hand, the Apache Spark framework, which provides implicit parallelism for big data analysis, has not gained extensive usage in SPM to the same extent it has in other domains of data mining. (Barsacchi et al., 2021).

In the early stages of PSPM research, several algorithms were developed based on data partitioning strategies, as outlined in (Bechini et al., 2023), along with the development of parallelized versions of traditional SPM algorithms. Within the domain of Apriori-based algorithms, there has been an extensive exploration of MapReduce implementations (Luna et al., 2018). Notable among these early proposals are pSPADE (Zaki, 2001a) and webSPADE (Demiriz, 2002), both of which were designed as parallel adaptations of the SPADE algorithm (Zaki, 2001b). In pSPADE, the search space is partitioned into smaller categories based on sequence

suffixes, which can then be solved easily using search techniques and join operations. On the other hand, webSPADE is a parallelized approach specifically tailored for analyzing click streams within website logs. Recognizing the growing significance of web log analysis, recent efforts have focused on creating proper solutions to dissect large volumes of such data through MapReduce methodologies (Sowmya et al., 2022).

Parallel and distributed adaptations have also been proposed for the GSP algorithm and its variant PSP. DGSP (Yu et al., 2015) and DPSP (Huang et al., 2010) are solutions designed within the MapReduce programming model. In contrast, PartSpan (Qiao et al., 2008) is a distributed and parallel algorithm tailored for identifying trajectory patterns. GridGSP (Wu et al., 2012) leverages a grid computing environment to parallelize the GSP algorithm effectively. Variants of the SPAM algorithm include SPAMC (Chen et al., 2013), which employs the MapReduce framework for distributing the computations, and SPAMC-UDLT (Chen et al., 2017). SPAMC-UDLT further improves the performance of SPAMC by introducing solutions to address scalability issues, enabling it to handle extremely large databases. Several concurrent versions have also emerged for pattern growth-based algorithms. Notably, Par-ASP (Cong et al., 2005) and Sequence-Growth (Liang and Wu, 2015) are noteworthy as parallelized adaptations of PrefixSPAN.

Finally, the literature has also focused on parallelized hybrid algorithms, including MG-FSM (Miliaraki et al., 2013) and MG-FSM+ (Beedkar and Gemulla, 2015). LASH (Beedkar and Gemulla, 2015) represents another hybrid parallel algorithm that leverages hierarchies. Other parallelized hybrid algorithms encompass Distributed SPM (Ge and Xia, 2016), which exploits dynamic programming and an extended prefix-tree to store intermediate results; ISM (Interesting Sequence Miner) (Fowkes and Sutton, 2016), a novel algorithm based on a probabilistic model and exploiting machine learning methodologies, and ACME (Sahli et al., 2013), which is a combinatorial method for extracting patterns from a long single sequence generally used in bioinformatics (Bechini et al., 2023).

Other approaches and related techniques

In recent years, there has been a significant interest within the research community in various areas closely related to Sequential Pattern Mining. This interest has given rise to several distinct lines of investigation. One such area is for example *Periodic Pattern Mining*, which involves the identification of patterns that display frequent and recurring occurrences within a unique longer sequence. The periodic nature of these patterns is assessed by considering their period lengths (Ravikumar et al., 2021). Additionally, exploring different event orderings has led to the utilization of graph databases. These databases are particularly suitable for representing itemsets alongside broader relationships among them. In such contexts, the SPM problem

can be reframed as Sub-graph Mining, where the goal is to discover frequent sub-graphs either within a database of graphs or within a single extensive graph (Bechini et al., 2023).

Several specific problems, closely related to SPM, have recently gained increasing importance. They are summarized in the following.

Weighted and High-Utility SPM. We already discussed events and objects within sequences, assuming an equal level of importance for all. This may not accurately reflect several real-world scenarios. To account for this variation, we can reformulate the SPM problem by explicitly considering different degrees of importance. Weighted SPM and, subsequently, High-Utility Mining are the two most popular approaches of this kind in the literature. In the context of Weighted SPM, each item in the sequence database is typically assigned a weight in the range $[0, 1]$, denoting its relevance. It is necessary to adapt classical SPM algorithms to handle weights as well. An example is *WSPAN* (Yun and Leggett, 2006), where the authors introduce a weight range and employ pruning techniques to eliminate sequential patterns with low weights. This approach results in the generation of fewer sequential patterns with higher overall weights.

High-Utility SPM represents an extension of Weighted SPM, where objects in the sequence database can have varying degrees of importance. Instead of assigning a weight to each object, High-Utility SPM considers that each object may appear zero, one, or multiple times within a transaction, and they are weighted based on their relative significance within the dataset. In addition to the minimum support threshold required to detect frequent sequential patterns, High-Utility SPM introduces another parameter known as the *utility threshold*. To be classified as frequent, a pattern must meet two conditions: its support should be greater than or equal to *minsupp*, and it must surpass the utility threshold. This threshold can be relative to the utility of each item or the total utility of items in a transaction. One prominent High-Utility Pattern Mining algorithm is USPAN (Yin et al., 2012). USPAN employs a lexicographic quantitative sequence tree to represent sequences and a set of concatenation mechanisms to identify High-Utility patterns. Recent research has also addressed High-Utility SPM for Big Data, developing algorithms that utilize the MapReduce distributed programming model (Lin et al., 2022). Additionally, efforts have been made to efficiently mine High-Utility Sequences with constraints (Truong et al., 2021).

Multi-dimensional SPM. Traditional SPM techniques enable the discovery of general patterns across the entire database, but they lack the capacity to address specific contextual issues (Pinto et al., 2001). For instance, a pattern identified as frequent in a sequence database of customer transactions might indeed be globally frequent. However, when analyzing a specific customer segment (such as the “over 55" age group), the same conclusion may not be true. To tackle this challenge, se-

quence databases are enhanced with annotations that capture distinctive attributes of the sequences. Frequent sequences are identified also along specific dimensions, some sequences might be frequent for certain values of those dimensions but not for others (Songram et al., 2006). An illustrative instance of a Multi-dimensional SPM algorithm is SeqDIM (Pinto et al., 2001). It can be seen as a meta-algorithm since it relies on a sequential pattern mining algorithm to uncover sequential patterns and an itemset mining algorithm to handle the dimensions.

Stream SPM. Methods known as "Stream SPM" have gained recognition in recent times to extract real-time knowledge from data streams, where a substantial amount of information becomes available in real-time. These algorithms are typically extensions of incremental methods, including the previously mentioned SPADE. The primary challenge for Stream SPM algorithms is their capacity to discover sequential patterns when the complete sequence cannot be read more than once, given its nature as an evolving data stream. Some well-known algorithms for Stream SPM include eISeq (Chang and Lee, 2005), IncSPAM (Ho et al., 2006), SPEED (Raissi et al., 2006), and Seqstream (Chang et al., 2008).

Uncertain and Fuzzy SPM. Data collected in real-world settings can frequently be subject to uncertainties and noise, and their analysis necessitates considering these undesirable characteristics. In the context of SPM, uncertainty can manifest in three distinct areas: the correct assignment of an event to the appropriate sequence (or "source"), the event itself, and the event timestamp (Muzammal and Raman, 2015). Temporal uncertainty may arise from various sources, including conflicting or missing event timestamps, network latency, granularity mismatches, synchronization issues, limitations in device precision, and data aggregation. A specific challenging case involves uncertainties related to the temporal arrangement of events (Ge et al., 2017). To address the challenges posed by inexact values or noisy datasets, specialized approaches like Uncertain SPM and Fuzzy SPM have been developed. These approaches often rely on probabilistic techniques and solutions from fuzzy logic. For instance, an adapted version of PrefixSpan was proposed (Muzammal and Raman, 2015), which leverages dynamic programming to compute the expected support of a sequential pattern.

Efficiency and scalability are challenging issues in the context of Sequential Pattern Mining (SPM) when applied to uncertain databases. One strategy devised to tackle these issues involves the use of dynamic programming for extracting probabilistic frequent sequential patterns within the distributed Spark platform (Ge and Xia, 2016). In response to the impact of uncertain data on High-Utility sequential patterns, a framework for mining high average-utility sequential patterns has been introduced (Lin et al., 2020). This framework aims to identify a collection of potentially high average-utility sequential patterns while considering the sequence size. However, a drawback of several recent efforts relies on efficiency, particularly in re-

ducing the number of false-positive patterns generated during the mining process. To address this issue, it has been demonstrated that leveraging a hierarchical index structure can yield promising results (Roy et al., 2021).

The mining of closed sequences in uncertain data has proved to be very challenging. The PFCSM-CF and PFCSM-CC algorithms (You et al., 2022) were developed to reduce the search space and simplify the candidate sequence database, leading to valuable computational advantages.

Fuzzy set theory has proven to be useful in the presence of uncertainties in SPM. For instance, in a study by (Zabihi et al., 2010), an algorithm that incorporates a sliding window constraint was introduced, enabling an element to be regarded as part of various transactions within a user-defined window. Additionally, fuzziness has been employed to handle High-Utility Fuzzy Sequential Patterns, as demonstrated in a recent study by (Ritika and Gupta, 2022).

Frequent Episode Mining

For the purposes and an in-depth understanding of this work, it is worth referring to *Frequent Episode Mining* (FEM), another important related pattern mining problem that received particular attention in recent years. FEM involves the identification of sub-sequences within a single sequence, where instances of these sub-sequences occur frequently, each following a specific temporal order. This challenge holds great importance in various application domains, particularly within the realm of system log analysis (Zhu et al., 2010). Noteworthy examples of applications in such domains are the monitoring of network traffic for detecting attacks, the monitoring of telecommunication alarm signals, or the analysis of usage data for recurring patterns.

In this setting, an event is always associated with a timestamp, and it may contain one or multiple items from a predefined set. The frequent sub-sequences of events are commonly referred to as *episodes*.

The Frequent Episode Mining (FEM) problem was initially introduced by Mannila et al. (Mannila et al., 1997). In their research, the authors define an episode as a collection of events that occur within time intervals of a predetermined size in a specified partial order (Mannila et al., 1997). Consequently, given a sequence of events, each with a timestamp, and a predefined time window, the primary goal of FEM is to identify sets of events, or subsequences, known as *episodes*, that occur frequently together within that time window while adhering to a particular partial ordering. In the most general case, certain events can occur concurrently, meaning they share the same timestamp. To highlight this characteristic, sequences that allow for concurrent events have been called *complex sequences*. Let us consider a finite set of items $I = \{i_1, i_2, ..., i_m\}$.

A complex sequence S of events, each with one or more items from I, consists of a temporally ordered list of tuples of the type (SE_{t_i}, t_i), where SE_{t_i} is the event occurring at timestamp t_i. An episode is a non-empty, totally-ordered set of events of the form $\langle E_1, E_2, ..., E_p \rangle$, where E_i is a subset of I and E_i appears before E_j for all integers i and j, with $i < j$, in the interval $[1, p]$. FEM aims to identify all frequent episodes in a sequence S. The occurrence of an episode is given by a time interval $[t_s, t_e]$ where t_s and t_e are the start and end timestamps for the episode, respectively. The support of an episode is given by the number of its occurrences within the sequence. Various methods have been suggested for calculating support. Older approaches proposed to calculate it based on the minimum number of occurrences of an episode (Mannila et al., 1997), while more recent methods often rely on the frequency of the episode head. The frequency of the episode head has been experimentally proven to be the best measure of support for this problem (Huang and Chang, 2008). In this context, support is determined by considering a user-defined window length (*winlen*) and a minimum support threshold (*minsupp*). The objective is to identify episodes that occur at least *minsupp* times within a *winlen* window.

The FEM problem has been addressed by various algorithms in the literature. Initially, (Mannila et al., 1997) proposed the WINEPI and MINEPI algorithms. These algorithms follow a similar procedure but differ in how they calculate frequency and support. WINEPI considers an episode frequent only when all its sub-episodes are also frequent. In contrast, MINEPI uses minimum occurrences to determine episode frequency, leading to the generation of rules with specific time limits. The authors claim that MINEPI can outperform WINEPI, especially in the later iterations of the algorithm. In another work, (Huang and Chang, 2008) EMMA and MINEPI+ were introduced. These algorithms improve upon WINEPI and MINEPI by utilizing head frequency counting to calculate sequence support. High-Utility episode mining algorithms have also been proposed, aiming to identify High-Utility episodes in complex event sequences, such as transaction databases. Recently, there has been a focus on mining serial episode rules from complex event sequences. The Forward and Backward Search Algorithm (FBSA) was developed (Kumar et al., 2023) to detect minimal occurrences of frequent peak episodes and reduce frequent sequence scans and redundant event sets. A novel technique known as Mining Serial Episode Rules (MSER) has been proposed, based on episode correlations and the generation of parameter selections where the occurrence time of an event is specified in the consequent (Poongodi and Kumar, 2022). Efforts have also been made to mine Frequent Serial Episodes over data streams (Guyet et al., 2022).

Challenges in the present work

In the present work, we address the Sequential Pattern Mining domain with a particular emphasis on its Frequent Episode Mining iteration.

In the second case study, in particular, classical Frequent Episode Mining algorithms were employed in the initial phase of a two-step framework designed for the extraction sequences of automated sequences of activity logs. This framework is developed within the context of the AUTOMIA project.

Moreover, the definition of episode has been broadened to extend its applicability beyond the domain of Sequential Pattern Mining. Notably, we introduce the concept of *text episode*, enabling us to treat segments of text (in the first and in the third case studies) as sequences from which we extract *interesting text episodes*. A detailed explanation of this approach is provided in Chapter 2.

3.3 Fake News Detection

Another important aspect pertaining to the present work is the fake news detection problem. In fact, one of the main goals of the present work is to address the problem of fake news detection by exploiting language modelling and considering fake news as text episodes, as defined in Chapter 2. However, before proceeding with an in-depth review of the literature on fake news detection, it is crucial to address various key aspects and definitions related to fake news and rumor detection, along with the approaches proposed to solve them.

In recent years, the proliferation of fake news and rumors on social media has become increasingly prevalent. The reason comes from the fact that social media have become relevant as a tool for spreading and consumption of news (Newman et al., 2013). Journalists utilize social media platforms to reach public opinion on breaking news stories and even uncover potential new stories. Meanwhile, users can track the progress of breaking news and events through verified social media accounts, or by following updates from their personal network. Indeed, social networks have demonstrated their exceptional utility, particularly in times of crisis, due to their innate capacity to disseminate breaking news faster than traditional media (Vieweg, 2010).

The distorted use of social media has become particularly pronounced in recent years, notably highlighted by the emergence of the first "infodemic" during the COVID-19 pandemic (Patwa et al., 2021), and then during the Ukraine-Russian war. This period has been characterized by numerous authors as a Post-Truth Era (Lewandowsky et al., 2017), where emotions and pseudo-facts dominate the landscape (Alam et al., 2021). This trend has further escalated with the onset of the Russian war against Ukraine. As observed in all conflicts, disinformation has emerged as a potent strategic weapon.

Nonetheless, the lack of control and fact-checking mechanisms on social media platforms creates a fertile environment for the dissemination of unverified and/or false information, which can significantly impact public opinion on critical matters

(Zubiaga et al., 2018a). A sensitive example is the 2016 USA presidential election campaign (Allcott and Gentzkow, 2017)

Fake news can manifest in various forms and formats within the social media landscape, including rumors and clickbait articles, making their efficient detection and mitigation a challenging problem, both through manual and automated means.

These issues have led over the years to the creation of numerous initiatives for independent fact-checking and fake news detection, and the topic has increased its relevance in the research community. The literature on issues related to fake news detection, disinformation, and fact-checking, is constantly growing despite the inherent challenges and multifaceted nature of the problem.

The research interest in fake news and rumor detection has significantly intensified in recent years, as it is demonstrated by the multitude of scientific publications dedicated to this topic. In the last few years, research efforts in fake news detection have been going in the direction of the multimodal setting.

Terminology and definitions

Examining the terminology surrounding fake news can offer valuable insights into this issue. Misinformation and false information have become widely associated with the term *Fake News*, which has become commonly used to refer to the dissemination of inaccurate information in mainstream media. However, several categorizations of false information have been proposed in the literature, mostly depending on the source and type of data used for analysis. Often, the boundaries between a rumor and a piece of fake news could be hard to define. A useful categorization is the one given by A. Bondielli and F. Marcelloni in their survey (Bondielli and Marcelloni, 2019). They divide the false information on the web into three subcategories: Fake News, Rumours, and others.

Fake News. There is no precise definition of Fake News. Often, this term is used to encompass a wide range of false information that is spread on the web (and beyond). Following the Allcott and Gentzkow definition, a piece of Fake News is "a news article that is intentionally and verifiably false" (Allcott and Gentzkow, 2017). *Intention* and *verifiability* are therefore two necessary properties to identify Fake News. They are in fact articles that are intentionally false, can be verified as such, and consequently, can mislead the reader. Several recent studies have adopted this definition. Fake News has also been distinguished according to different aspects in *serious fabrications*, *large-scale hoaxes*, and *humorous fakes* (Rubin et al., 2015).

To summarize, we can identify three key aspects concerning fake news: i) its form, as a piece of news article; ii) its intent, which can be either satirical or malicious; and iii) the verifiability of its content as completely or partially false.

Rumors. If the attention on Fake News has widely increased only in recent years, the term *Rumours* has been extensively used and studied in scientific literature.

Rumors refer to information that has not been confirmed by official sources yet and is spread mostly by users on social media platforms. However, in literature, there is no unique definition of Rumours, each of them differs from one another. DiFonzo and Bordia refer to rumors as unverified (and potentially unverifiable) information that emerges in contexts of ambiguity, danger, or potential threat, with the purpose of helping make sense of a particular situation and manage the danger/risk involved (DiFonzo and Bordia, 2007). From this perspective, rumors are defined by a *context*, a *function* that justifies them, and evidently a *content*.

Another definition of rumor is the one adopted by Zubiaga et al. in their survey (Zubiaga et al., 2018a): "A piece of information whose truthfulness has yet to be verified at the time it was published." From a more practical perspective, (Zubiaga et al., 2018a) split rumors into two categories: *long-standing rumors*, that represent unverified information circulating for long periods of time (e.g. conspiracy theories), and *breaking news rumors*, that often appear in connection with breaking news stories, and could either be the product of unintentional misinformation or intentional deception.

Other kinds of false information. In addition to Fake News and Rumors, it is interesting to provide further examples of false information that can be distinguished from the previous ones. We can refer to *clickbait*, for instance, to describe the headlines of articles (or even social media posts or streaming platform videos) whose main objective is to entice users to click on the article or video in question solely to generate a view (which often serves as the primary source of revenue on the web). It is also noteworthy how the phenomenon of clickbait has contributed in its own way to the spread of Fake News on the web (Silverman, 2015). Another type of false information disseminated on the web falls within the domain of *social spammers*, a phenomenon that is often associated with phishing (Shu et al., 2017).

Most common approaches

The Fake News Detection problem is often treated as a binary classification task. The objective is to predict whether an article (or any other online text) can be classified as fake news or not. For this reason, in most cases and in recent years, researchers have proposed approaches based on the implementation of machine learning, and specifically deep learning, strategies.

Another line of research exploits different methodologies, such as data mining techniques like time series analysis, and has utilized external resources such as knowledge bases to predict document classifications or events, as well as assess their credibility. It is worth noting that fact-checking has also played a significant role among these alternative approaches. Nonetheless, most of the presented approaches and techniques are suited for a uni-modal setting. Multimodality has re-

ceived relatively less attention over the years in this context (Alam et al., 2021), but in the latest years, this has rapidly changed.

Machine Learning and Deep learning approaches

An important differentiation among approaches should be established based on the features that are considered relevant for detecting fake news and/or rumors. They have been categorized as content features and context features (Shu et al., 2017). In the former case, these features pertain to information that can be directly extracted from the text itself. Regarding texts such as articles, some examples of features in this category include the source of the article, the headline (a short text designed to grab attention and entice the reader to continue), stylistic and/or linguistic information, and audio/video-related information. Depending on the type of content we want to utilize and the type of representations we want to construct, we can differentiate between *linguistic-based* features and *visual-based* features.

The linguistic-based features are extracted in a way that allows us to isolate specific linguistic patterns that may be involved in identifying a fake news article. For example, frequent use of exaggerations and negations, or excessive use of punctuation marks are some patterns that can be considered. Linguistic features are further subdivided into subcategories that refer to *syntactic*, *morpho-syntactic*, *lexical*, and *semantic* traits.

Syntactic and lexical features include the presence and frequency of specific words and patterns. They have been utilized as indicators for identifying fake news by leveraging the linguistic properties of texts in several works (Qazvinian et al., 2011; Zubiaga et al., 2016b; Chua and Banerjee, 2016; Hardalov et al., 2016; Feng and Hirst, 2013; Potthast et al., 2016). Morpho-syntactic features refer to the frequency of specific patterns of Part of Speech or punctuation. Semantic features use semantic information such as topics and word embeddings. Recently several techniques that exploit semantic features have been effectively applied to the detection of fake news and rumors, particularly in machine learning and deep learning approaches (Jin et al., 2016; Ma et al., 2016; Ruchansky et al., 2017; Zubiaga et al., 2016b).

In addition to linguistic features, there are also visual features that are extracted from visual elements such as videos or images. Some examples of these features include clarity score, coherence score, similarity distribution histogram, diversity score, and clustering score (Shu et al., 2017). Additionally, there are statistical visual-based features such as image ratio, multi-image ratio, hot image ratio, and long image ratio (Shu et al., 2017).

On the other hand, contextual features refer to information that relates to the environment and context in which a particular news item is found. These features are classified based on three aspects of the context that one intends to represent

(Shu et al., 2017): the social media users (user-based features), the posts generated by users (post-based features), and the networks of users (network-based features).

User-based features model user profiles and their interactions with news within social media. They can be classified at individual and group levels. In the first case, the features extracted pertain to the credibility of each individual user. In the second case, they refer to groups of users who are somehow connected to the same news (Yang et al., 2012).

Post-based features focus on identifying information that can be extracted from different elements of social media posts (Shu et al., 2017). These features can also be categorized at multiple levels: the *post-level*, the *group-level*, and the *time-level*. In the first case, we refer to unique features for each post, whereas in the second, the objective is to aggregate specific features that pertain to groups of posts, which might be related to particular articles or news. Time features, on the other hand, pertain to variations over time in the values of the features identified at the post level (Ma et al., 2015).

Network-based features are extracted using networks of users who have published certain social media posts (Shu et al., 2017). It's possible to build different types of networks that, for example, can indicate the following/followee structure of social media users (such as Twitter) to trace the circle of those affected by the propagation of certain information. Some studies, however, are limited to the use of statistics on the spread of specific patterns, such as the number of retweets and propagation times (Kwon et al., 2017).

In the literature, it is noticeable that the majority of studies currently focus on the utilization of a specific category of features, but there exist some hybrid models. Many fake news detection methods primarily rely on content-based features. Rumors, on the other hand, are more likely to be approached by utilizing a combination of both content-based and context-based features for analysis. In many cases, in fact, the primary goal is to identify rumors within streams of social media posts.

The problem of fake news and rumor detection has been mostly considered a standard classification problem, and their approaches focused mainly on the implementation of machine learning strategies to solve it. The earliest approaches focused on the application of traditional machine learning algorithms, such as Support Vector Machines (SVM) (Afroz et al., 2012; Briscoe et al., 2014; Pérez-Rosas and Mihalcea, 2015; Rubin et al., 2016; Zhang et al., 2012; Qin et al., 2016; Wu et al., 2015; Horne and Adali, 2017), Decision Tree or Random Forest (Aker et al., 2017; Castillo et al., 2011; Giasemidis et al., 2016; Zhao et al., 2015), Conditional Random Field (CRF) (Zubiaga et al., 2016b, 2017) and Hidden Markov Models (HMM) (Vosoughi, 2015; Vosoughi et al., 2017).

Feature extraction for these approaches is time-consuming and may produce biased features (Ma et al., 2016), a critical issue to fake news and rumor detection

tasks.

Many modern approaches use deep neural network architectures. They can learn hidden representations from input both in context and content variations (Ma et al., 2016). The challenge is shifted to model the network such that it can solve the task efficiently. In the fake news and rumor domain Recurrent Neural Networks and Convolutional Neural Networks are very common. Ensemble hybrid approaches have been also adopted and have obtained competitive performances (Ma et al., 2016; Ruchansky et al., 2017; Chen et al., 2017; Wang, 2017; Zubiaga et al., 2018b).

It is important to notice that the Transformer architecture and its transfer learning abilities have been applied also to the fake news domain, obtaining state-of-the-art performances (Slovikovskaya, 2019; Qazi et al., 2020).

Some studies have also been done recently in the Large Language Modelling and prompting domains (Whitehouse et al., 2022)

Fact checking

It is important to acknowledge the body of literature dedicated to the investigation of fake news and rumors within the context of computational-oriented fact-checking (Shu et al., 2017). Fake news and rumor detection, as well as fact-checking, are undeniably interrelated. The primary goal is to facilitate automatic fact-checking.

Numerous approaches have been proposed in the field. Magdy and Wanas (2010) automated web-based fact-checking by comparing facts extracted from a document with facts obtained from related URLs. Wu et al. (2014) introduced automated fact-checking and presented algorithms for generating queries to determine the truthfulness of statements. Knowledge graphs have emerged as a widely used technique, employed in several studies (Ciampaglia et al., 2015; Shi and Weninger, 2016). One of them proposed utilizing Wikipedia infoboxes to create a knowledge graph and developed a semantic proximity measure using a transitive closure algorithm to verify claims against the graph (Ciampaglia et al., 2015). Shi and Weninger (2016) treated the problem as a link prediction task in a knowledge graph, leveraging meta-paths to reduce the search space for statement verification.

It is significant that various competitions, such as Barrón-Cedeno et al. (2020) and Nakov et al. (2022), have been organized to assess computational fact-checking approaches. Notably, there exists an interesting fact-checking system based on a combination of Information Extraction and Deep Learning strategies designed for the task named "Verified Claim Retrieval" (Task 2) during the CheckThat! 2020 evaluation campaign (Passaro et al., 2020). The system achieved a MAP@5 score of 0.91 on the test set.

Multimodal fake news detection

All the approaches and techniques discussed in the previous sections have been proposed for fake news detection and rumors in a uni-modal setting. Most of the proposed approaches use either the actual content of the news (i.e., the text itself), its context (e.g., social network structures, temporal information), or a combination of both (Bozarth and Budak, 2020). Most modern systems typically leverage transformer models with additional information (Passaro et al., 2020).

The simplest method for disseminating misinformation is through textual content. Nonetheless, digital platforms and social media offer additional avenues for this purpose. For instance, images can be employed within the realm of disinformation and false news in various ways. Firstly, incorporating images into deceptive content can increase the credibility of the accompanying text, enhancing the spread of false news. Secondly, images can be described in a manner that distorts their original context, potentially leading readers astray and perpetuating disinformation. Lastly, they can be utilized to increase the appeal of a post and encourage widespread sharing among social media users, thereby amplifying the reach of fake news.

The examination of social media data in a multimodal context may be considered closer to real-world scenarios. However, it's worth noting that multimodality has received comparatively less attention over the years (Alam et al., 2022).

Fortunately, this situation is rapidly evolving, with several international multimodal shared tasks emerging in areas such as fake news and propaganda detection, fact-checking, and related fields. Some notable examples include the SemEval (Da San Martino et al., 2020), HatefulMemes (Kiela et al., 2020), and SemEval again (Dimitrov et al., 2021) competitions. Despite these advancements, it's important to highlight that the development of models capable of effectively combining multiple modalities to detect fake news remains a significant open challenge in the academic literature. Additionally, there is a need for datasets that encompass various modalities and different sources of fake news (Alam et al., 2022).

Challenges as MULTI-Fake-DetectiVE, represent some steps in this direction. The challenge is part of the EVALITA 2023 Evaluation campaign (Lai et al., 2023), and it is a task aimed at addressing both the textual and visual aspects of fake news on social media and online news outlets, from two key perspectives: the focus is on fake news detection from a multimodal perspective, to explore how images and texts interact and influence each other in the context of real and fake news. MULTI-Fake-DetectiVE task is part of the present work and is discussed in detail in Chapter 6.

Social media database collection

When tackling a task like Fake News Detection or Rumor Detection, one of the main challenges is data collection and the subsequent construction of a proper dataset. The problem is not easy from various perspectives. Firstly, data collection can be expensive and complicated. The difficulties lie not only in gathering a large amount of data but also in annotating it with the appropriate truthfulness evaluation. A dataset or corpus that can be used for Fake News Detection should ideally contain articles, tweets, or any other type of information that can be collected online, annotated as fake news or real news, or assigned a score (e.g., from 1 to 7) indicating the level of text reliability. This task can be approached using various strategies. One of these strategies is called Expert-oriented fact-checking, which involves the assistance of experts to assess the reliability of information. Typically, this type of strategy is employed by certain websites that position themselves as comprehensive "archives" of what is commonly regarded as misinformation or hoaxes. As an example, websites of this kind include BuzzFeedNews[1] or Snopes[2] (Bondielli and Marcelloni, 2019). The main challenge with using websites like these is that they are typically limited to specific domains of interest and require experts in those particular fields. Furthermore, this approach, in addition to being costly due to the required work, does not generate algorithmically usable datasets but rather collections of fake news in different formats depending on the referenced website. Moreover, this makes it difficult to obtain good datasets that allow for a certain degree of generalization across different domains (Pérez-Rosas et al., 2017). However, one advantage is that it often allows for tracing back to the source of the news, which can be a video, a post on a social network (such as Facebook or Twitter), or an actual newspaper article.

Another data annotation strategy involves the use of crowdsourcing platforms for data evaluation. Additionally, some computational models utilize algorithms and external resources, such as knowledge graphs. These resources are employed for both data retrieval and annotation purposes.

Publicly available datasets

Up to now, not many resources are publicly available, concerning Fake News Detection. This may depend on several factors. It is not an easy task to identify relevant data and propose effective strategies to collect them. There are no available standard definitions of fake news or rumors. It may increase the difficulty when labeling the data. Furthermore, the majority of data are found in social media, and such platforms are restrictive when giving access to their data for several reasons, such as privacy.

[1] `https://www.buzzfeednews.com/`
[2] `https://www.snopes.com/`

For what concerns fake news in particular, there are no agreed-upon benchmark datasets (Shu et al., 2017), but several publicly available resources are worth mentioning.

BuzzFeedNews[3] is a dataset available on GitHub and contains a total of 1627 articles that represent a true sample of news published on Facebook during periods closely related to the 2016 American presidential elections. The included articles were published between September 19th and September 23rd, and between September 26th and September 27th (Shu et al., 2017). Each post also includes the link that connects it to the referenced article, which has been verified by five BuzzFeed journalists. The dataset was subsequently enriched with additional metadata (Potthast et al., 2017).

LIAR[4] is a dataset whose data was collected from the PolitiFact[5] website through its API (Wang, 2017). It consists of 12,836 instances from different contexts, including newspaper articles or radio interventions. This dataset is structured in such a way that the labels intended to identify the truthfulness of the texts include multiple classes, ranging from "false" to "true," with intermediate classes such as "mostly true," "half true," and "barely true." In addition to the label, each annotation includes a justification for the assigned label, the author, and the context from which it was taken (Bondielli and Marcelloni, 2019).

BS Detector[6] is a dataset developed by Kaggle and is unique as it is constructed using a web crawler called "BS detector[7]," which is designed to verify the truthfulness of news (Shu et al., 2017). In this case, the labels for the dataset instances are determined by the outputs of an automated system, the BS detector extension, rather than human annotators, as in BuzzFeedNews and LIAR. Therefore, it is not possible to define the Kaggle dataset as a gold standard due to this reason (Bondielli and Marcelloni, 2019).

CREDBANK[8] is a dataset that contains approximately 60 million tweets published over a hundred days starting from October 2015. These tweets are grouped into events, and each event is annotated using the Amazon Mechanical Turk crowdsourcing platform by thirty annotators (Mitra and Gilbert, 2015).

In Shu et al. (2017), the four datasets were compared, and it was noted that none of them individually can evaluate all the interesting features for a Fake News Detection task. In fact, the first three datasets (BuzzFeedNews, LIAR, and BS detector) only provide content features, specifically linguistic features. Only CREDBANK allows the analysis of contextual features. These datasets also have other limitations.

[3] https://github.com/BuzzFeedNews/2016-10-facebookfact-check/tree/master/data
[4] https://www.cs.ucsb.edu/william/data/liardataset.zip
[5] https://www.politifact.com/
[6] https://www.kaggle.com/mrisdal/fake-news
[7] https://github.com/bs-detector/bs-detector
[8] http://compsocial.github.io/CREDBANK-data/

For example, BuzzFeedNews lacks any annotation related to its social context and contains articles from relatively few newspapers. Looking at the LIAR dataset, we can view that the instances include very short snippets and not entire newspaper articles, and they were collected from different speakers, not from actual news outlets. Furthermore, the labels of the dataset used by the BS detector were automatically annotated and not validated by human experts, which greatly affects the credibility of the data. As mentioned in Shu et al. (2017), any model trained on these data will not learn from annotations of expert annotators but rather from the parameters of the BS detector itself. Lastly, CREDBANK has the major drawback of not being originally designed for Fake News recognition. It was initially intended as a collection of data for evaluating the credibility of tweets.

Another interesting dataset is called FakeNewsNet[9], a collection of articles that includes both content-related information and contextual information (Shu et al., 2017).

An issue that arises in the design of datasets for the task of Fake News Detection is the existence, on the web, of articles that fall under the broad category of false information but are not necessarily fake news as they belong to the realm of satire. In Rubin et al. (2016), efforts were made to differentiate satire from real news, resulting in a corpus of 240 articles encompassing both satirical and real news across four domains (civic, science, business, and soft news).

FakeNewsAMT and Celebrity (Pérez-Rosas et al., 2017) are two datasets designed and constructed using different approaches, which were then used to develop computational models to tackle Fake News Detection tasks.

FakeNewsAMT is a dataset of news that refers to six different domains (sports, business, entertainment, politics, technology, and education). The real news articles were collected using reliable sources, while their fake counterparts were created using Amazon Mechanical Turk, a crowdsourcing platform. Users were asked to try to transform the real news into fake news and emulate, if possible, the journalistic style (Pérez-Rosas et al., 2017).

Celebrity is a dataset where fake news, unlike the previous dataset, was collected from the web. The authors chose to focus their research on the domain of famous public figures, as they are often subject to rumors, hoaxes, and fake news. The sources used include, among others, Entertainment Weekly, People Magazine, and RadarOnline (Pérez-Rosas et al., 2017).

Regarding rumors, efforts have been made particularly within the scope of the PHEME project (Derczynski and Bontcheva, 2014). A dataset considered a benchmark for various evaluation purposes was collected by Zubiaga et al. (2016b). The dataset consists of tweets linked to nine distinct rumors gathered during the period between 2014 and 2015. This dataset has been employed in various research stud-

[9] `https://github.com/KaiDMML/FakeNewsNet`

ies (Zubiaga et al., 2016b) as well as the RumourEval task within the SemEval 2017 evaluation campaign (Derczynski et al., 2017). Additionally, the aforementioned CREDBANK (Mitra and Gilbert, 2015) can serve as a valuable resource, specifically for the task of classifying the veracity of rumors on social media.

Challenges in the present work

In the present work, we address the Fake News Detection problem from two different perspectives.

First, we introduce a novel, topic-oriented methodology designed for collecting and labelling potentially fake and verified news from social media, given the context of a specific event. The methodology is aimed at facilitating the real-world-focused analysis of fake news. We produced two datasets for Fake News Detection and Multimodal Fake News Detection.

Second, we leverage and adapt the similarity-based method to highlight information divergence between articles, described in Chapter 4, and propose an episode mining-inspired approach in order to take the initial steps for addressing the Fake News Detection task in a novel perspective.

Chapter 4

Case Study: Information Divergence with Sentence Similarity

Newspapers often differ in their approach to specific topics, making it challenging for users to access clear and comprehensive information. Different sources may present the same event from various perspectives, leading to potential contradictions in content. Again, some newspapers may disseminate misleading or fake news (Bondielli and Marcelloni, 2019). Consequently, each news article may contain different details about a given topic. Users typically desire a comprehensive understanding of the subject to form their own opinions, possibly skipping the reading of all the related news articles, which often include redundant information, preferring instead to concentrate solely on their differences.

In this Chapter, a case study on the use of sentence similarity to extract new information on a set of articles based on the information divergence with respect to a reference article is presented. In other words, multiple newspaper articles (denoted as *target articles*) that deal with specific topics are considered events of a longer sequence into which we want to extract new information with respect to an external newspaper article (denoted as *the reference article*). We developed a system that, given a reference article, retrieves a set of target articles on the same topic, and automatically mines the contents of these articles to extract differences with respect to the reference article. The system is based on a methodology that leverages the similarity between sentences to find groups of them with varying degrees of similarity, or divergence.

In recent years, Transformer architectures like BERT (Devlin et al., 2018) have established themselves as the standard, achieving state-of-the-art performance in various NLP-related tasks, including token-level contextualized representations. However, when it comes to sentence-level comparisons, there are two significant limitations associated with using a conventional classification approach.

Firstly, as highlighted by the creators of BERT (Devlin et al., 2018), sequence-level

representations are not naturally suited for gauging semantic similarity between sentences. While token-level representations extracted from BERT-like models often deliver results that are comparable with or even superior to traditional methods like word2vec (Mikolov et al., 2013b) for token-level semantic similarity, these models are not particularly reliable in assessing the similarity between entire sentences.

Second, employing a classification model for tasks like sentence-level similarity assessment can be computationally expensive. Transformer-based neural language models often demand GPU acceleration during the training phase, making them resource-intensive, especially for tasks that involve comparing the similarity of entire sentences.

As for the quality of sentence-level representations, we considered Sentence-BERT (Reimers and Gurevych, 2019). It is a method for further fine-tuning a transformer model that leverages siamese and triplet network structures, and semantic textual similarity, and natural language inferencing training tasks to improve the sequence-level representations provided by a Transformer model. Sentence-BERT is specifically designed to derive semantically meaningful sentence embeddings that can be compared using cosine as a similarity metric (Reimers and Gurevych, 2019). This is particularly relevant for mitigating the computational problem. We can argue that the computational cost associated with transformer-based models is a less prominent issue when these models are solely used for feature extraction purposes. For this reason, we leverage Sentence-BERT to derive similarity ratings at the sentence level for newspaper articles.

The rest of the Chapter is organized as follows. In Section 4.1 we describe the system developed and, in particular, we describe how each module works. The core of the system, which exploits a proposed methodology aimed at highlighting information divergence, is the sentence classification algorithm implemented in the second module. In Section 4.2 we show some experiments performed to validate the sentence classification process. Section 4.3 shows some discussion and future possible applications. Finally, Section 4.4 reports the summary of the Chapter.

4.1 Description of the system

The system takes as input a reference article. It retrieves a selection of target articles from several newspaper websites related to the same topic or event. Then, the system produces as an output a summary of the information contained in the target articles, focusing on content that is considered dissimilar to the information found in the reference article (i.e. the information included in the target articles that diverge from those in the reference article and other target articles).

The proposed system is based on four main steps (i.e. Target Articles Retrieval, Sentence Classification, Target Sentence Reduction, and Target Sentence Summa-

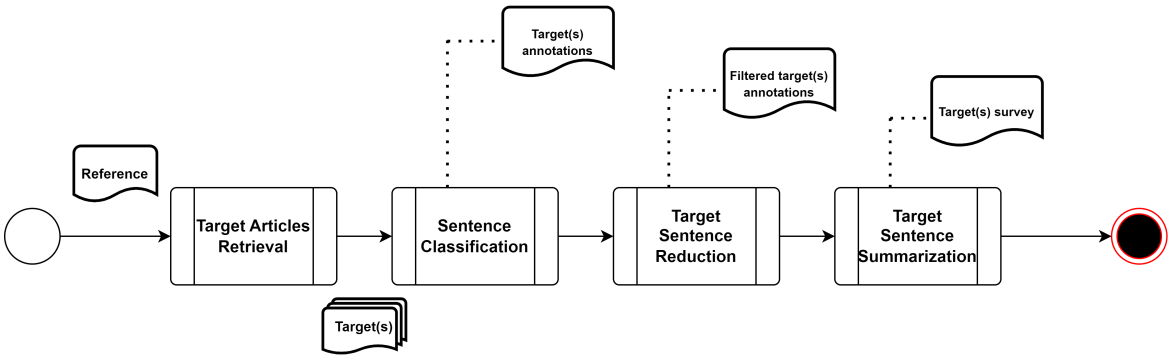

Figure 4.1: Flowchart of the proposed system

rization) and is presented in the flowchart in Figure 4.1. The central component of the system is the algorithm used in the Sentence Classification module, which allows labelling the sentences of the target articles into three classes, namely *Similar Sentence* (*SS*), *Different Sentence* (*DS*), and *Very Different Sentence* (*VDS*). The labelling is determined by assessing the similarity values between the sentences and the reference article, a process facilitated by sentence embeddings generated using a pre-trained Sentence-BERT model.

In the following part of this section, all the modules of the system are described, namely the Target Articles Retrieval, the Sentence Classification, the Target Sentence Reduction, and the Target Sentence Summarization modules.

Target Articles Retrieval

Given a reference article, that is an article already read by the user, the Target Articles Retrieval module searches for articles from a pre-defined set of sources, comprising widely popular newspaper websites. Then, a simple yet efficient and effective method based on Named Entity Recognition (NER) is exploited to verify if the article potentially shares the same topic as the reference. Named Entities are extracted from both the reference and the target articles exploiting the SpaCy NLP pipeline[1]. SpaCy is an open-source text processing library in Python. We used version 3.5.0, and the module `en_core_web_lg`.

If the potential target article share at least a third of the Named Entities extracted also from the reference article, then the article is maintained. Using Named Entities helps ensure that the selected articles are related to the same topic as the reference article since they refer to the same Named Entities. At the same time, the system also considers articles that share only some, but not all, Named Entities with the reference article, ensuring that the target articles contain new or different information with respect to the reference article.

[1] `spacy.io`

This preliminary step enables the system to retrieve potential target articles from the web without applying expensive tasks related to semantic similarity. While this initial step may involve minimal loss of information, the system will conduct a more in-depth analysis and employ specific techniques to calculate the similarity between the target and reference articles in the subsequent steps.

Sentence Classification

When provided with a reference article and the set of target articles obtained through the previously described process, the Sentence Classification module classifies all the sentences in the target articles into one of three classes: SS (Similar Sentence), DS (Different Sentence), or VDS (Very Different Sentence), with respect to the reference article.

Before the actual classification, it is necessary to obtain the sentences for both the reference and target articles and a viable representation of the sentences. The reference article is given by the user, while the target articles are automatically retrieved during the previous step. To obtain the viable representation of the sentences of all the articles, we perform an NLP Preprocessing step.

For each input article, sentence splitting is performed. The Spacy pipeline, which was previously employed to extract Named Entities during the Target Article Retrieval module, as previously described, is employed as well. Additionally, a list of keywords from both the reference and target articles is extracted using KeyBERT [2]. KeyBERT is a recent and straightforward yet powerful method for extracting keywords and keyphrases. It utilizes BERT embeddings and cosine similarity to identify sub-phrases within a document that are close to the document itself.

Then, to obtain reliable representations of news that are semantically relevant, we exploit a Sentence-BERT pre-trained model to encode the sentences. In particular, the model used is the *distiluse-base-multilingual-cased-v2* pre-trained model (Reimers and Gurevych, 2020). It is a multilingual model, and its training is based on the idea that a translated sentence should be mapped in the same location within the vector space as the original sentence. We used this model with our approach to enhance the generalization of the proposed methodology for newspaper texts in different languages (Reimers and Gurevych, 2020).

Finally, all the sentences in the target articles are classified into SS, DS, or VDS. We formally define our classification problem as follows.

Let r and $T = \{t_1, t_2, ..., t_n\}$ be, respectively, a *reference* news article and a set of one or more *target* articles. Both r and t_i are represented as sets of sentences $r = \{s_1^r, s_2^r, ..., s_n^r\}$ and $t_i = \left\{s_1^{t_i}, s_2^{t_i}, ..., s_{m_i}^{t_i}\right\}$. The goal is to classify each sentence $s_j^{t_i}$ in each t_i with one of the three classes (SS, DS, VDS) based on degrees of sim-

[2] `https://github.com/MaartenGr/KeyBERT`

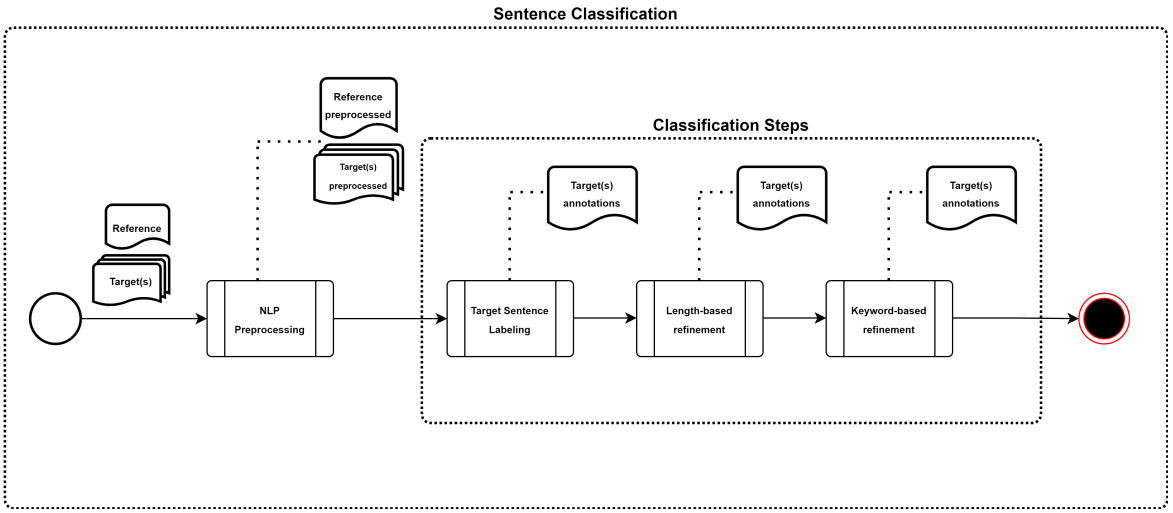

Figure 4.2: Flowchart of the Sentence Classification. After the NLP Preprocessing, the three main steps are performed.

ilarity with all the sentences in r. In particular, the label is assigned to $s_j^{t_i}$ by considering the maximum similarity between $s_j^{t_i}$ and all the sentences in r. We exploit two thresholds, considered as parameters: the High Similarity (HS) and the Low Similarity (LS). They serve to discriminate, respectively, between SS and DS, and DS and VDS. A method to select the best thresholds that are able to generalize on several events is described in Section 4.2. If at least one sentence in r has a similarity score higher than HS with $s_j^{t_i}$, then $s_j^{t_i}$ is classified into the SS class. If no sentence in r has a similarity score higher than LS with $s_j^{t_i}$, then $s_j^{t_i}$ is classified into the VDS class. Finally, if the maximum similarity score between $s_j^{t_i}$ and a sentence in r is between LS and HS thresholds, $s_j^{t_i}$ is classified into the DS class.

After selecting the appropriate thresholds, the Sentence Classification process can be subdivided into three subsequent steps:

- **Coarse-grained labelling**: the target sentences are labelled based solely on the cosine similarity of sentence-level embeddings.
- **Length-based refinement**: the labelling is revised by taking sentence length into account.
- **Keyword-based refinement**: the labelling is further refined by considering shared keywords between sentences

The flowchart of the whole process, including pre-processing, is shown in Figure 4.2.

In the following, we describe in detail the three substeps performed for target sentence classification.

Target Sentence labelling

Once each sentence $s_j^{t_i}$ in t_i and each sentence s_l^r in r are represented as embeddings, we compute the similarity between $s_j^{t_i}$ and s_l^r, for each sentence in t_i and r, by exploiting the cosine similarity. Table 4.1 shows how the sentences $s_j^{t_i}$ in t_i are classified with respect to the values of maximum cosine similarity and give the definition for each class.

Table 4.1: Description of the three classes and corresponding classification rules based on the maximum cosine similarity between the sentence $s_j^{t_i}$ of the target article t_i under analysis and the sentences of the reference article r.

Labels	Classification Rules	Description
SS	**IF** Maximum Cosine Similarity between $s_j^{t_i}$ and each sentence in reference article $r >=$ HS **THEN** classify $s_j^{t_i}$ as SS	Sentences in this class include pieces of information which are shared with the reference article. Probably, this information refers to the objective description of the event.
DS	**IF** Maximum Cosine Similarity between $s_j^{t_i}$ and each sentence in reference article $r <$ HS **AND** $>=$ LS **THEN** classify $s_j^{t_i}$ as DS	Most of the sentences in this class include pieces of information which probably are shared with reference sentences but reported differently.
VDS	**IF** Maximum Cosine Similarity between $s_j^{t_i}$ and each sentence in reference article $r <$ LS **THEN** classify $s_j^{t_i}$ as VDS	Sentences containing information different from the one in the reference sentences; it contains also some noise.

Length-based refinement

An important issue that emerges after the Target Sentence Labelling step is that when comparing two sentences of different lengths, their similarity score can appear low even if they share similar content. This occurs because traditional similarity measures often consider the length of sentences as a factor, which can lead to misleading results when assessing the similarity of sentences with differing lengths. Length is a critical factor when working with BERT-like models. Sentence-BERT has indeed demonstrated its capability to produce semantically meaningful sentence representations in the vector space (Reimers and Gurevych, 2019). However, it is important to note that the cosine similarity computed between embeddings of sentences with similar content but varying lengths may not consistently return high similarity scores.

Table 4.2 provides an illustrative example, where the two sentences express the same information. Nevertheless, due to their significant difference in length, the cosine similarity computed between their respective embeddings is too low to label the target sentence as SS. Thus, the system reiterates all the pairs of target-reference with only those target sentences that are not labelled as SS and are significantly shorter in terms of token count compared to the reference sentence (specifically, the reference sentence is at least two times longer in terms of tokens than the target sentence). To determine if the longer sentence contains additional information, the system employs the following method.

Table 4.2: Example sentences with a low cosine score (0.58) that share some core information.

Source	Sentence
Fox News (target)	*'The FBI also has been called to investigate the incident.'*
CNN International (reference)	*'The FBI is investigating the incident*, which drew widespread condemnation of the officers after a video showing part of the encounter circulated on social media.'

The system identifies the most representative words, including keywords and named entities, that are shared between the reference sentence and the target one. For each of these keywords and named entities, the system extracts the text fragments from both the left and right context in the reference sentence.

Table 4.3: An example of fragment-reference sentence pair with high cosine similarity score.

Fragment	Target Sentence	Cosine Similarity
'The **FBI** is investigating the incident'	'The **FBI** also has been called to investigate the incident.'	0.8

Each of these text fragments is compared with the target sentence. If the two sentences contain overlapping content information, at least one pair will have a cosine similarity greater than or equal to the threshold HS. In such cases, the system updates the labeling from VDS or DS to SS for the target sentence. If no such pair reaches the similarity threshold, no updates are made. This iteration continues until there are no more pairs in the VDS or DS groups to compare.

Table 4.2 shows a fragment obtained from the reference sentence of CNN International, along with a comparison to the target sentence of Fox News. Their cosine score is 0.8.

Keyword-based refinement

Another issue to take into account is that Sentence-BERT representations often put sentences with shared representative words (i.e. named entities, keywords) closer in the latent space, thus limiting the effectiveness of the cosine similarity metrics in such cases where sentences share these representative words but differ in overall content.

Table 4.4: Two sentences with a high cosine score that actually report different information about the same topic but share a set of keywords.

Reference Sentence	Target Sentence	Cosine Similarity
'Pfizer anticipates applying for emergency use authorization for a **third dose** of its vaccine as soon as next month, **Mikael Dolsten**, who leads worldwide research, development and medical for **Pfizer**, was quoted by CNN as saying at the teleconference.'	'During a company earnings call on Wednesday morning, Dr. **Mikael Dolsten**, who leads worldwide research, development and medical for **Pfizer**, called the new data on a **third dose** of vaccine ëncouraging.'	0.7

In the example shown in Table 4.4, the two sentences share a set of Named Entities (Mikael Dolsten, Pfizer, third dose), and this leads to a high cosine similarity even if the sentences contain rather different informative content.

To solve this problem, the system re-examines all the target sentences labelled SS and DS. The system checks for each sentence if the information contained in the target sentence is similar to the information in the reference sentence. In order to do this, all the common representative words are excluded.

Formally, for each pair $s_j^{t_i}$ in t_i and s_l^r in r first common keywords and named entities are identified. Then, the system generates text fragments from both sentences, excluding those keywords and named entities. Subsequently, the system compares each text fragment from the reference article with all the text fragments from the target article. If all possible combinations of text fragments extracted result in cosine similarity scores lower than the threshold LS, the classification of the target sentence is changed to VDS. This refinement helps ensure that similarity judgments are made based on content beyond just shared keywords and entities.

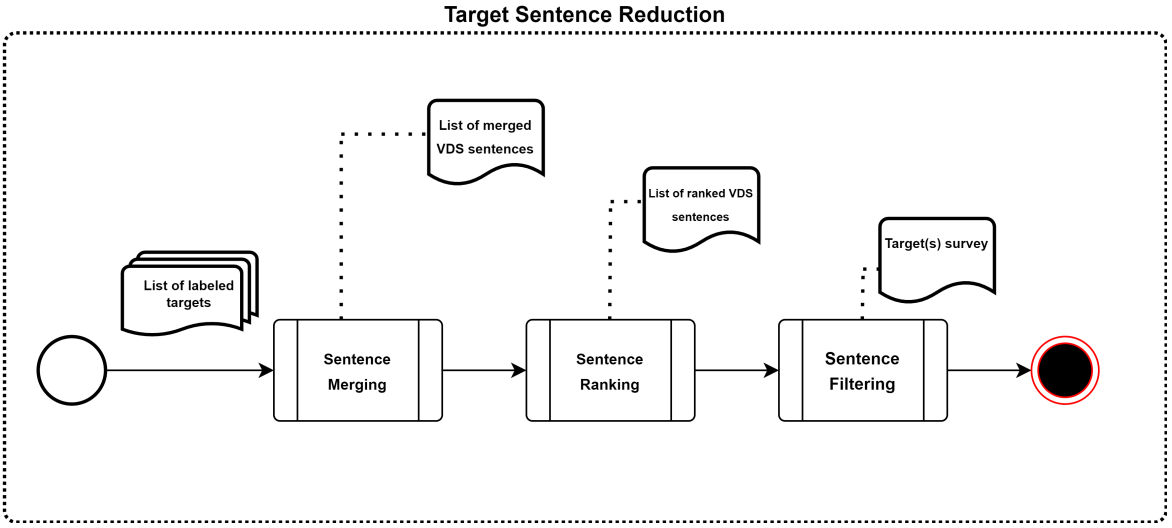

Figure 4.3: Flowchart of the Target Sentence Reduction Sub-Process, which consists of three main steps: Sentence Merging, Sentence Ranking, and Sentence Filtering.

At the end of all these steps, the system outputs a set of classified target articles $T_C = \{t_1, t_2, ..., t_n\}$, where $t_i = \left\{s_1^{t_i}, s_2^{t_i}, ..., s_{n_i}^{t_i}\right\}$ is the set of all the sentences of the article t_i, labelled as SS, DS and VDS.

Target Sentence Reduction

After obtaining the labelled set of sentences within T_C, the process of selecting sentences is refined by detecting similarity or redundancy among sentences in T_C. As these redundant sentences usually provide minimal extra information, they are excluded from further consideration. The Target Sentence Reduction module is illustrated in Figure 4.3.

To identify and eliminate redundant sentences within the set of labeled target articles $T_C = \{t_1, t_2, ..., t_n\}$, the system first combine the sentences from target articles labelled as DS and VDS, excluding those labelled as SS, as they can be considered redundant. Then, it ranks the DS+VDS list in descending order based on the maximum similarity of the target sentences with the reference sentences. Starting from the top of the ranked list, for each sentence, the similarity score with all the other sentences in the list is computed. If at least one of these similarity scores is higher than the HS threshold, indicating that there is another sentence similar to the one being considered and with a lower similarity score to the reference article, the system marks the sentence as redundant and removes it from the list.

The final list includes only sentences that are considered non-redundant.

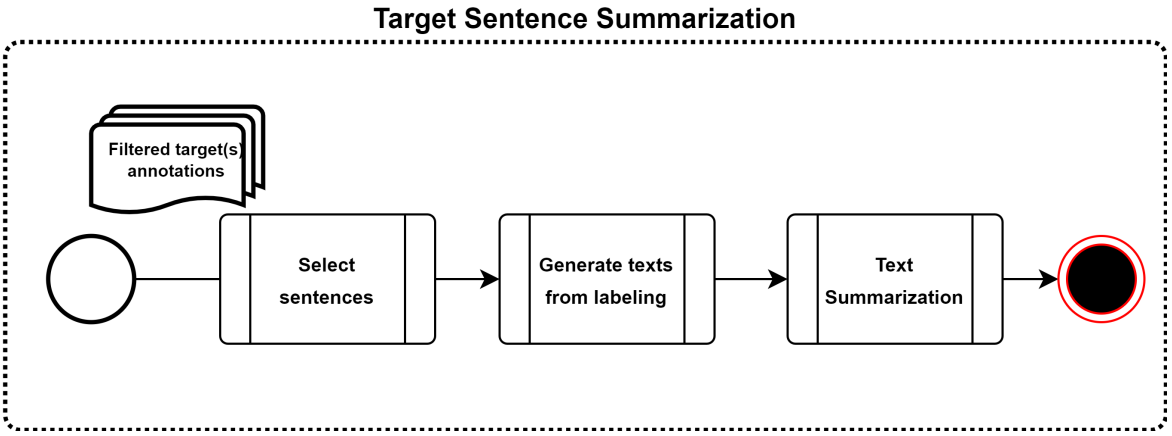

Figure 4.4: Flowchart of the Target Sentence Summarization Sub-Process.

Target Sentence Summarization

The end goal of the proposed system is to provide users with information content they may have missed with the article of their choice (i.e., the reference article) in an easy-to-read format. It is clear that providing a set of sentences extrapolated from the target articles may be not very appealing and understandable. A simple strategy could be to highlight target sentences in the target articles. However, this implies that the user has to at least skim the target articles to find the novel content. With the aim of making the information promptly available to the user, the system automatically generates an abstractive summary of the target sentences and outputs this summary. The flowchart of the Target Sentence Summarization process is shown in Figure 4.4.

Based on our definition, all sentences labelled as VDS are selected because they certainly correspond to sentences with information that diverges to some extent from the reference article. All sentences classified as DS with a cosine similarity score lower than LS when compared to all the sentences in VDS are selected as well. All these sentences are then combined into a unique text. The system then generates a summary of this text by exploiting an abstractive text summarization model. In particular, we employed the Google PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence models) model. PEGASUS is a large pre-trained Transformer-based encoder-decoder model that employs a gap-sentences generation strategy. The summary produced is a coherent and comprehensible text that presents information exclusively from the VDS sentences and some of the DS sentences for each target article. This approach enables us to effectively visualize the novel content contained in the target articles compared to the reference article.

Table 4.5 presents an example of a summary generated from segments of arti-

cles related to the Notre Dame Fire. The reference article is a segment of an article published on April 16, 2019, by NBC News.

Table 4.5: Reference article and abstractive summary generated from VDS sentences and some of the DS sentences collected from two target articles. The event concerns the Notre Dame Fire in 2019

Reference Article (NBC News)	Summary of target articles (USA Today & Al Jazeere English)
'Some of the most prized, centuries-old relics of France and Christianity survived the devastating Notre Dame Cathedral fire that almost wiped out the cherished Paris landmark, authorities said Tuesday. Culture Minister Franck Riester breathed a huge sigh of relief, telling reporters outside Notre Dame that the nation's "most precious treasures" were largely spared. Saved treasures from the Notre Dame Cathedral are temporarily stored in city hall after a massive fire devastated large parts of the Gothic cathedral in Paris. Benoit Tessier / Reuters Many of the works will be stored at Paris' City Hall and the Louvre, where they will be examined, treated for damage and protected, officials said. Audrey Azoulay, director-general of UNESCO, the U.N. culture agency, said Notre Dame has "a particular place in the world's collective imagination" and has pledged her agency's help to rebuild. Monday's fire almost destroyed the entire cathedral, which has stood in Paris and survived nearly 900 years of tumultuous French history. Paris prosecutor Remy Heitz has launched an investigation, which he said would be "long and complex."'	'The huge fire that tore through Notre Dame Cathedral in Paris last night has left the famous landmark in ruins, but not as badly as many had feared. The famous spire was reduced to ash, but the building's stone face and bell towers were saved, the BBC reports. The main altar has been badly damaged, but the famous white marble Piet' sculpture by French sculptor Nicolas Coustou is still standing, seemingly unscathed, at its place before the main altar. "The great tragedy is the structure itself and its fittings," a Brown University art professor tells the Los Angeles Times. "Many people don't know that the roofs are timber, largely replaced in the 19th century, and they have been totally consumed. Masonry will burn and degrade under intense heat, which is clearly what is happening, but we can only wait and see what kind of damage has spread to the church within." A full investigation is underway.'

4.2 Experiments and validation

In this section, we present the results of experiments conducted to evaluate the proposed approach. Our primary goal was to assess the performance of the Sentence Classification module and its generalizability across different events. To facilitate this evaluation, we compiled and annotated a dataset comprising online newspaper articles spanning various topics. This dataset served a dual purpose: it acted as a benchmark for performance assessment and as a training dataset for optimizing parameters, particularly the thresholds. Within this dataset, each reference article was

paired with at least two other target articles related to the same news event. Crowdsourcing was employed to label the target sentences with three classes: SS, DS, and VDS, indicating their similarity with sentences in the reference article. Our experiments involve comparing the Sentence Classification module's performance to human judgments. Additionally, we conducted a secondary comparison involving two BERT classifiers, one based on model tuning and the other on prompt tuning. This step aimed to establish a reliable benchmark against state-of-the-art approaches.

Lastly, we evaluated the coherence, correctness, and completeness of the summary generated as the final output of the system. To do this, we gathered human judgments through crowdsourcing. Participants were asked to rate the summary's quality in terms of completeness, factual accuracy, and novelty, given a reference article and the target sentences.

Dataset

The dataset we collected comprises 43 articles related to 8 distinct news events, with at least three articles associated with each event. The primary assumption is that articles concerning the same event and published on similar dates are comparable. For each news event, we marked one article as the Reference Article, while the others were considered as Target Articles. In the dataset, each article is accompanied by metadata, including the news it pertains to, the primary topic of the news (e.g., politics, health), the source newspaper, publication date, and URL. To streamline the data collection process, we leveraged CrowdTangle.[3] We compiled a list of events occurring between 2020 and 2021 and gathered social media posts from the Facebook pages of international newspapers (e.g., CNN, The Guardian, Al Jazeera English) containing links to news articles. Subsequently, we retrieved the news content directly from the respective newspaper websites. To establish ground-truth labels for sentences within the target articles, we conducted a crowdsourcing experiment using Prolific.[4] Annotators were presented with pairs consisting of a target sentence and a reference sentence. They were tasked with determining whether the target sentence was similar, different, or very different from the reference sentence. Each pair was assessed by 7 different annotators, and the gold label was determined by the majority vote among the annotations. Pairs with an uncertain majority vote were excluded from consideration. The dataset is available on GitHub.[5]

[3]https://apps.crowdtangle.com/
[4]https://www.prolific.co/
[5]https://github.com/pietrodelloglio/dataset-for-system-acquiring-content-differences

Target Sentence Classification Evaluation

First, we carried out a set of experiments aimed at determining the best threshold parameters for the proposed system. Given a set of different values for parameters LS and HS, we compared the automatic classification performed by our system with human labelling. We executed a leave-one-out cross-validation: we held out one news as test and used the remaining ones to determine the threshold values that best fit the human labelling. For each fold, we obtained 8 different best pairs of LS and HS. We selected the pair of thresholds which obtained the best performance in the highest number of news (in case of tie, we selected a pair randomly). We point out that we obtained the same pair of LS and HS for all the folds. Thus, we can conclude that the choice of the LS and HS thresholds is not particularly sensitive.

Table 4.6 shows an example of F1-scores obtained in a single fold.

Table 4.6: F1-scores of different threshold configurations in a single fold.

LS	**HS**	Mars Landing	Italy Lock-down	Italy Euro 2020	Global Warming	George Floyd Death	Elliot Page Coming Out	China Landed on Mars
0.4	**0.6**	0.46	0.61	0.62	0.61	0.47	0.87	0.59
0.4	**0.7**	0.52	0.58	0.63	0.62	0.47	**0.89**	0.61
0.4	**0.8**	0.54	0.58	0.65	0.62	0.47	**0.89**	0.59
0.5	**0.6**	0.6	0.75	0.75	0.7	**0.79**	0.85	0.75
0.5	**0.7**	0.65	**0.79**	0.76	0.71	**0.79**	0.87	0.77
0.5	**0.8**	**0.67**	**0.79**	0.78	**0.72**	**0.79**	0.87	0.75
0.6	**0.7**	0.56	0.72	0.82	0.71	**0.79**	0.87	**0.88**
0.6	**0.8**	0.58	0.72	**0.84**	**0.72**	**0.79**	0.87	0.86

We opted for LS = 0.5 and HS = 0.8 as our parameter values. They consistently obtained the best results across the majority of news events.

Table 4.7 shows the performances in terms of F1-score on the test news for each fold. On average, our system was able to obtain an F1-score equal to 0.77, with the lowest value equal to 0.67 and the highest one equal to 0.87.

To compare our system with state-of-the-art classification models, we employed two BERT classifiers, involving model tuning and prompt tuning. We used the `bert-base-uncased` model (Devlin et al., 2018). For the first classifier, we fine-tuned BERT in a leave-one-out cross-validation setting. The fine-tuning parameters are set as follows: 128 maximum sequence length, 2e-5 learning rate, with a batch size of 8, for computational limitations. We trained the model for 5 epochs in each fold.

Table 4.7: F1-score for the single news, and the average F1score

News	F1-Score
Mars Landing	0.67
Italy Lockdown	0.79
Italy Euro 2020	0.78
Global Warming	0.72
George Floyd Death	0.79
Elliot Page	0.87
China Landed on Mars	0.75
Beirut Explosion	0.81
Average	**0.772**

The other parameters are the standard of the huggingface[6] implementation. For the second classifier, we employed prompt tuning, with the following parameters: 128 maximum sequence length, 3e-2 learning rate, with a batch size of 8, for computational limitations. We trained the model for 30 epochs in each fold.

We report the results of the two BERT classifiers and their comparison with our system in Table 4.8.

The table shows that, on average, fine-tuned BERT performs slightly better than our system, which, in turn, outperforms the model tuned with prompts. It's important to note, however, that fine-tuning a BERT-based classification model is resource-intensive in terms of time and computational power. In contrast, our system is considerably more efficient. It only requires extracting sentence-level embeddings and computing the similarity matrix. Prompt tuning, on the other hand, is a technique that tends to perform better with larger models and also requires substantial computational resources.

In addition to this, we show an example of the confusion matrices for three of the considered news: the Beirut explosion (Figures 4.5 and 4.6), the coming out of Elliot Page (Figures 4.8 and 4.9), and the victory of Italy in the Euro 2020 Championship (Figures 4.11 and 4.12).

The analysis of the results indicates that, on average, our system achieves slightly lower accuracy compared to BERT. However, it is crucial to note that the performance varies depending on the class of similarity and the specific news event. For example, our system performs slightly better on the DS class in the news about the Beirut Explosion and the news about Italy's victory, even if the f1-score of the fine-

[6] `https://huggingface.co/bert-base-uncased`

Table 4.8: F1-score for the single news and the average F1-score for our classifier and the two BERT classifiers.

News	Our System	Bert M.T.	Bert P.T.
Mars Landing	0.67	0.73	0.56
Italy Lockdown	0.79	0.75	0.61
Italy Euro 2020	0.78	0.82	0.77
Global Warming	0.72	0.72	0.66
George Floyd Death	0.79	0.74	0.58
Elliot Page	0.84	0.92	0.83
China Landed on Mars	0.75	0.89	0.84
Beirut Explosion	0.81	0.81	0.56
Average	**0.772**	**0.797**	**0.676**

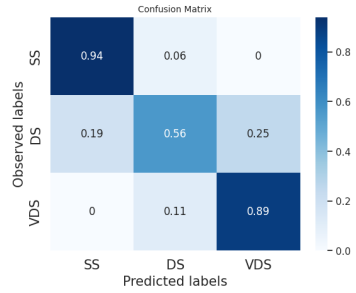

Figure 4.5: Confusion Matrix for our system on the news about Beirut Explosion

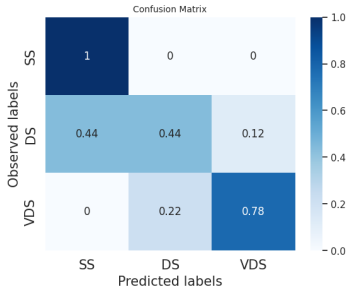

Figure 4.6: Confusion Matrix for BERT on the news about the Beirut Explosion

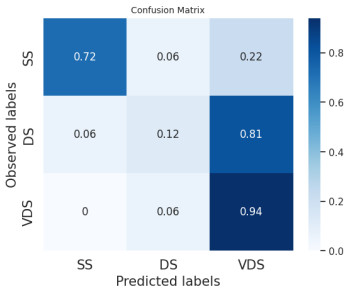

Figure 4.7: Confusion Matrix for BERT Prompt tuning on the news about the Beirut Explosion

tuned BERT is higher in general.

For the event about Elliot Page coming out, both our system and BERT demonstrate similar performance in the DS class, while the results for Prompt Tuning are not comparable.

Finally, Figures 4.14, 4.15, and 4.16 present the average confusion matrices for all the tested news. These figures reveal that the BERT classifiers perform slightly better on the VDS classes, while our system performs better on the DS class. Fine-tuned BERT outperforms on the SS class. Generally, both SS and DS classes are the most challenging classes to predict for all the classifiers. The SS class, in particular, appears slightly unbalanced in some events within our dataset. This occurs because different articles about the same event often share a couple of similar sentences,

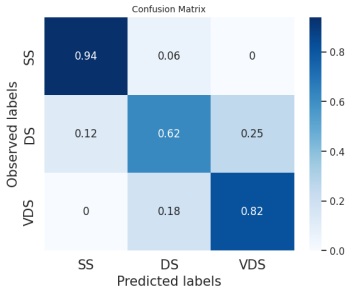

Figure 4.8: Confusion Matrix for our system on the news about Elliot Page coming out

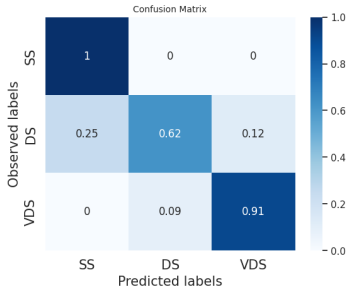

Figure 4.9: Confusion Matrix for BERT on the news about the Elliot Page coming out

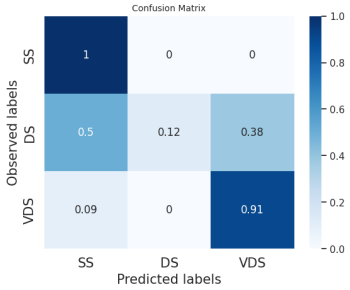

Figure 4.10: Confusion Matrix for BERT Prompt tuning on the news about the Elliot Page coming out

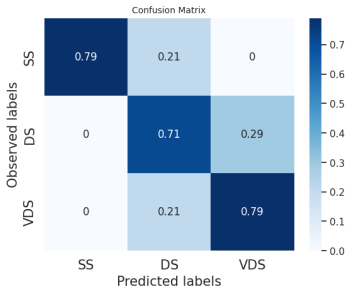

Figure 4.11: Confusion Matrix for our system on the news about Italy Euro 2020

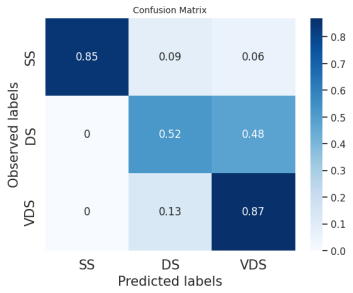

Figure 4.12: Confusion Matrix for BERT on the news about the Italy Euro 2020

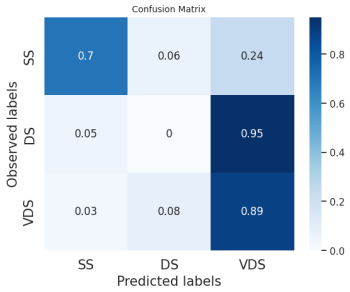

Figure 4.13: Confusion Matrix for BERT Prompt tuning on the news about the Italy Euro 2020

causing the rest to fall into the DS and VDS classes. Conversely, the DS class is inherently less homogeneous than the other two. It typically includes sentences with the same informative content as the most similar sentence in the reference article but presented in a different linguistic style.

In conclusion, we can state that our sentence classification method performs quite well compared to human-labeled ground truth data. On average, our classifier's performance is slightly below that of the fine-tuned BERT classifier. To assess whether the differences in performance between our system and fine-tuned BERT (as well as prompt-tuned BERT) are statistically significant, we conducted the Wilcoxon signed-rank test using the default parameters provided by Scipy[7]. For the comparison between our system and fine-tuned BERT, we obtained a test statistic of $W = 5.0$ (the minimum of the sum of ranks above and below zero) with a *p-value* $= 0.87$, indicating that there is no statistically significant difference between

[7] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html`

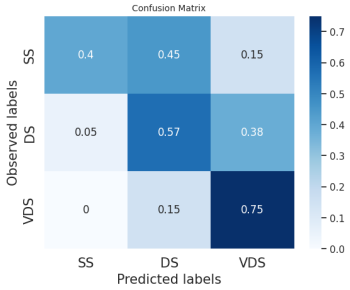

Figure 4.14: Average Confusion Matrix for our system

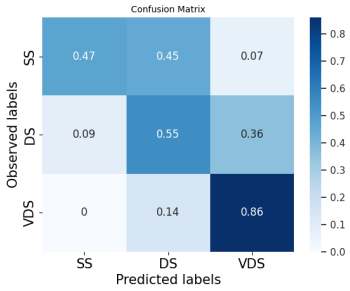

Figure 4.15: Average Confusion Matrix for BERT

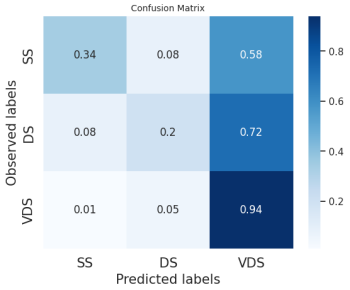

Figure 4.16: Average Confusion Matrix for prompt tuning

the two classifiers. In the comparison between our system and prompt-tuned BERT, we obtained a test statistic of $W = 4.5$ with a $p\text{-}value = 0.05$, suggesting a statistically significant difference between the two classifiers.

Furthermore, it is important to note that BERT always requires fine-tuning with annotated data, while our system can be used without the need for a tuning phase, as we can directly employ default thresholds, as demonstrated in our experiments. Prompt tuning, on the other hand, tends to obtain better results when applied to larger models but it is also expensive in terms of computational resources.

Text Summarization Evaluation

To evaluate the quality of the generated summaries, we conducted a crowdsourcing experiment. For each news article in the dataset, we presented the raters with three components: the reference article, the target sentences extracted from the target articles, and the summary generated from these sentences. Raters have then been asked to rate each summary based on three criteria: novelty compared to the reference article, factual correctness compared to the target sentences, and information completeness compared to the target sentences. These ratings have been provided by 7 different users using a Likert-type scale ranging from 1 (lowest) to 7 (highest) for each criterion.

Figure 4.17 displays the human judgments for each piece of news regarding the novelty of the summaries, which assesses the amount of new information in the target summary compared to the reference article. On average, the score is 3.5. Some news articles received better evaluations than others. The summaries related to topics like Elliot Page coming out, the George Floyd Death, and the Mars Landing achieved majority scores between 4 and 6, indicating a moderate to high level of novelty. The news about Italy winning the Euro Championship received a lower score, suggesting a lower level of novelty. It should be noted that the assessment of novelty is influenced by a certain degree of subjectivity that depends on the percep-

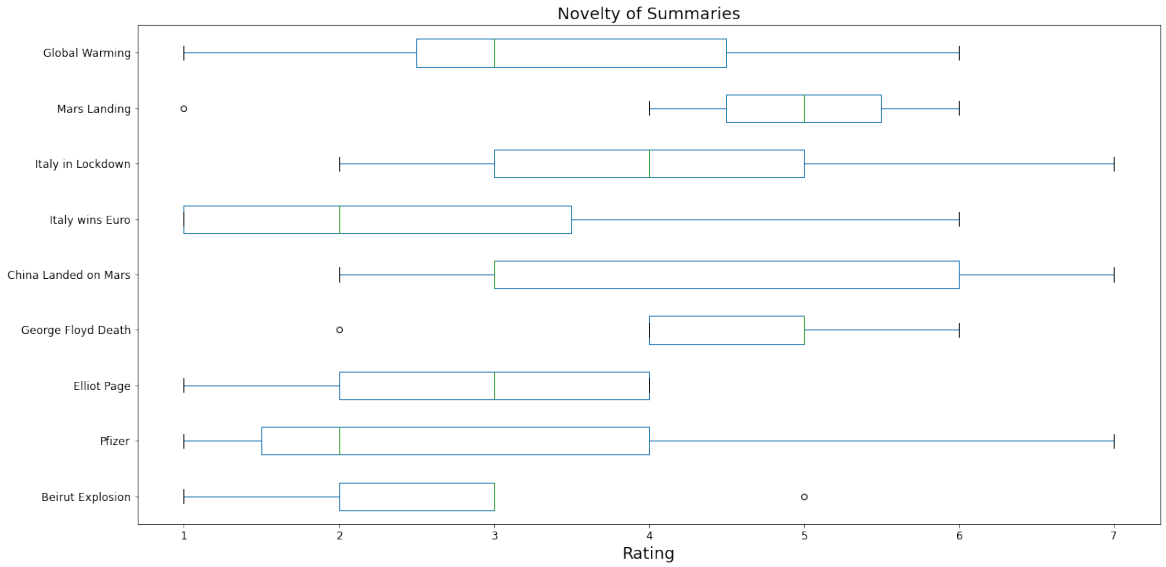

Figure 4.17: Box plot of human judgments for each news on the amount of new information contained in the summary compared to the reference article

tion of what constitutes "novelty". The same piece of information, conveyed through two distinct stylistic approaches, may be perceived as either identical content or as new content. For instance, in the case of Elliot Page Coming Out, the information directly expressed by Elliot Page himself regarding his coming out, and the same information filtered through the lens of the article's author could be evaluated as either novel or as similar information by different users.

Figure 4.18 presents the human judgments for each news regarding the factual correctness of the target summary compared to the reference article. The average score is 4.6. For most news articles, the majority of scores fall between 5 and 6, suggesting that the target summaries are generally factually accurate with respect to the reference articles.

Figure 4.19 displays the human judgments for each news article regarding the information completeness of the target summary compared to the reference article. The average score is 3.8, indicating a reasonable level of information completeness. The majority of scores vary, with some news articles receiving higher scores for information completeness (e.g., Elliot Page coming out and George Floyd Death), while others have slightly lower scores. Overall, the results are encouraging, suggesting that the target summaries contain a satisfactory level of information.

The experiment results indicate that the automatically generated summaries generally provide an adequate summarization of the target sentences. Specifically, the correctness ratings are on average satisfactory, indicating that the summaries are factually accurate with respect to the original news. The completeness ratings are also quite high, although there is some variation in quality for specific news articles.

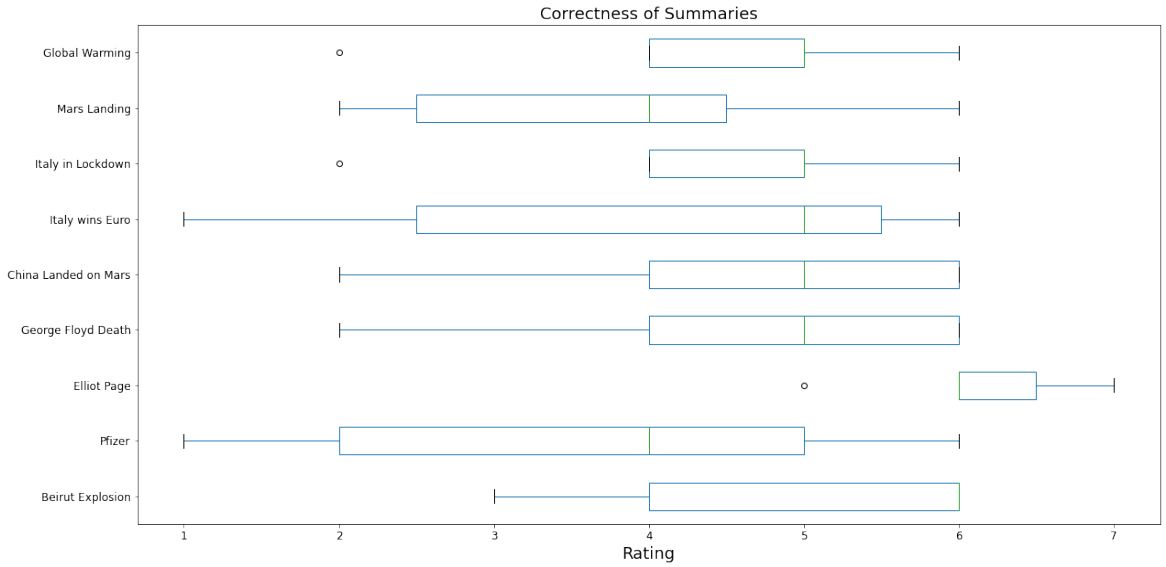

Figure 4.18: Box plot of human judgments for each news on the factual correctness of the target summaries compared to the reference article

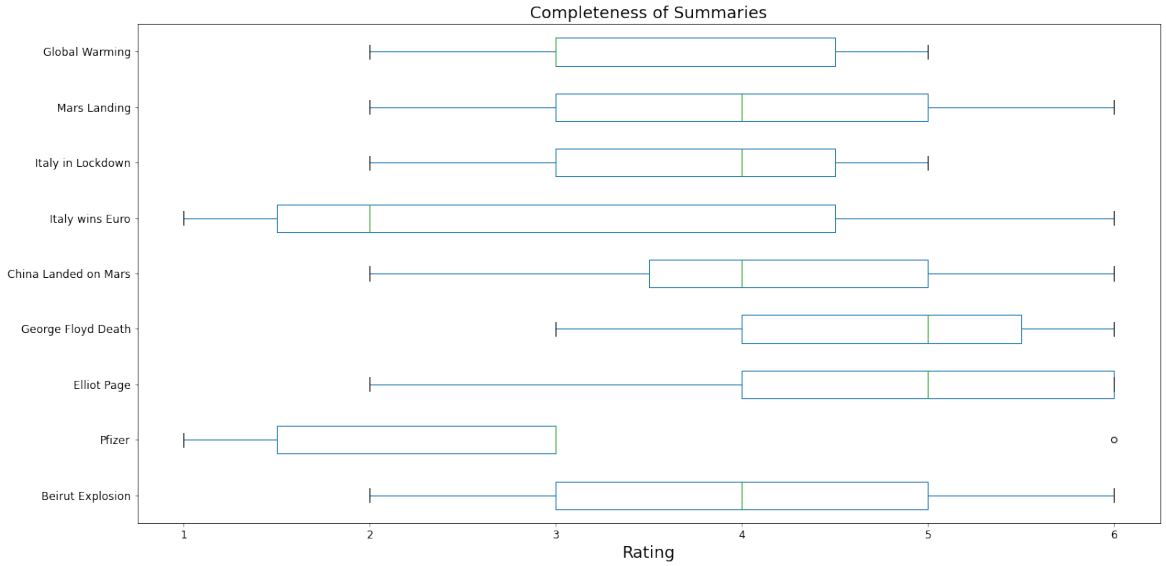

Figure 4.19: Box plot of human judgments for each news on the information completeness of the summaries of the target sentences extracted from the target articles compared to the reference article

While novelty ratings are generally lower, they are still encouraging.

In summary, while user satisfaction varies depending on the specific news, on average, the system is capable of producing coherent and comprehensible summaries of the target sentences.

4.3 Discussion

The system described in this Chapter implements a methodology that employs cosine similarity based on Sentence-BERT embeddings to classify all the sentences from target articles as Similar, Different, and Very Different with respect to sentences in a reference article. This method is aimed at extracting the most relevant sentences (i.e., the Different and Very Different ones), that correspond to novel information with respect to the reference article, meaning all the information that diverges from the one included in the reference text. We performed a set of experiments by using the human judgments collected by crowdsourcing to evaluate the effectiveness of the classifier and the ability of the summary to represent the information contained in the target articles and different from the one in the reference article. Moreover, we evaluated our approach against state-of-the-art techniques.

This method, and the system in which it is implemented, is a first step towards the end goal of empowering users to discover novel and relevant information and will be further improved in future works. In particular, we intend to investigate other similarity metrics and other classification algorithms to better address the problem of semantic similarity between sentences. Moreover, we strongly believe that such a methodology can be used from new perspectives. We will work to improve it in order to fine-tune an accurate and efficient strategy for detecting information divergence in several main topics of NLP, such as Fake News Detection.

4.4 Summary

In this Chapter, Section 4.1 has presented a system that leverages sentence similarity and association rules to extract new information from a collection of articles in comparison to a reference article. This system serves as a case study to introduce the concept of Information Divergence.

In Section 4.2 we tested and evaluated the similarity methodology based on Information Divergence, that is the main core of our system. Additionally, an assessment is conducted on the Novelty, Completeness, and Correctness of the generated summaries.

Finally, in Section 4.3 concluding remarks and future directions are provided.

Chapter 5

Case Study Pattern Discovering with Frequent Episode Mining

In recent years, alongside the development of advanced language models and machine learning systems, there has been growing attention towards the design of specialized software systems aimed at automating repetitive interactions between human operators and software applications. Internet bots, commonly referred to as "bots", serve as a prime example. Bots are software systems that carry out automated tasks online, with the goal of replicating human actions. One critical aspect of this automation process is the identification and modeling of target processes, which correspond to sequences of actions. In order to identify these sequences, the analysis of recordings of interactions between human operators and computers is required. These interactions are typically captured using action-tracking software tools, which generate detailed logs of the activities performed. However, not all activities are suitable for automation, as they may either lack repetition or not offer significant advantages when replaced by automated processes. The primary focus should be on identifying frequent sequences that follow specific patterns and have the potential to yield substantial benefits when automated. Manual identification of such sequences can be challenging, especially in cases where human operators are simultaneously engaged in multiple tasks.

In this chapter, we introduce a framework for mining sequences of actions, particularly those that can be automated using log data generated from human interactions with specific software applications. This framework is a key component of the AUTOMIA project, which encompasses a range of software technologies, including Robotic Process Automation (RPA) systems, bot management algorithms, and Frequent Episode Mining techniques.

The reason for discussing the AUTOMIA project and, more specifically, this framework in the context of this work is driven by a significant factor: it provides an opportunity to emphasize that a problem that falls more within the realm of Data Min-

ing than Natural Language Processing, can be effectively reversed and subsequently approached within another domain of interest. This presents a valuable chance to highlight that Episode Mining techniques, originating from the field of Frequent Episode Mining, can be successfully adapted for application from a Natural Language Processing perspective.

In the rest of the chapter, the context of the AUTOMIA project in which the framework was developed is presented in Section 5.1. Then, in Section 5.2 the proposed methodology is effectively described, while its evaluation and some experiments are reported in Section 5.3. The proposed approach was also evaluated in a specific use case within a real-world setting, extensively presented in 5.4. Finally, in Section 5.5 a discussion is drawn, that serves as the starting point for the next Chapter.

5.1 The context: the AUTOMIA Project

The AUTOMIA project encompassed research, design, development, and experimentation of a Robotic Process Automation (RPA) system. This system aimed to assist operators in executing tasks that were highly repetitive and of low complexity. The project has particular relevance within the context of Industry 4.0, which aims for complete integration between human operators and the surrounding industrial world through the use of state-of-the-art systems. The primary goal of the AUTOMIA system was to collaborate with human operators to free them from repetitive tasks of low complexity. These tasks often exhibit high error rates and can demotivate workers. By automating these tasks, operators could concentrate on higher-value tasks, leading to resource savings for the company. This project was funded by the Tuscany Region as part of the POR FESR 2014-2020 program.

Brief definition of Robot Process Automation

Robotic Process Automation (RPA) involves automating processes through the robotic execution of tasks. RPA is a software-based solution designed to automate business processes that are rule-based, involve routine tasks, deal with structured data, and yield deterministic outcomes (Aguirre and Rodriguez, 2017). It serves as a gateway to introduce automation into processes, aiming to achieve commercial benefits with sustainable costs and minimal risk. The core idea is simple: a software *robot* replicates typical human-computer interactions, automating tasks that would otherwise be repetitive and monotonous.

In contrast to general-purpose software or commonly used apps, the software that underlies RPA is custom-made. It typically utilizes a variety of specially crafted software routines (workflows). Each workflow can contain other routines that control fundamental software operations, often referred to as software BOTs or Robots.

The effectiveness of RPA relies on how well the optimization flow is interpreted and, more importantly, how accurately RPA replicates the operational flow.

The role of UNIPI in the AUTOMIA project

The Department of Information Engineering at the University of Pisa was responsible for developing a module within the system that extracts processes from data logs. This module followed a two-step framework, incorporating Frequent Episode Mining and fuzzy string matching. In the first step, the focus was on applying Frequent Episode Mining algorithms to identify potentially relevant patterns in the data. This step aimed to detect recurring sequences of events or actions within the data. The second step involved identifying possible occurrences in the available data based on specific patterns of interest. This was achieved by utilizing fuzzy string matching algorithms, which are based on metrics like the Levenshtein distance. Fuzzy string matching allows for the identification of similar sequences even when there are slight variations or errors in the data. To evaluate the effectiveness of this approach, it was tested using a benchmark dataset to assess its performance under controlled conditions. Additionally, a real-world dataset consisting of activity logs and actions of interest obtained from the AUTOMIA project was used to further test the module and provide a practical case study for the research conducted in this work.

5.2 Methodology

The proposed methodology relies on a two-step process, as shown in the flowchart in Figure 5.1. The *Frequent Episode Miner* step (FEM-M) performs Frequent Episode Mining to discover statistically significant episodes in the action log. We refer to these episodes as *candidate episodes* and they encompass processes worthy of automation. As their number may be very large, and considering that a portion of them might not be relevant to our goals, a downstream selection becomes necessary. The *Target Episode Finder* (TEF-M) step plays a crucial role in identifying the most significant episodes within the data. This is achieved by utilizing a set of patterns known as *archetype patterns* and making comparisons using similarity metrics. Archetype patterns are defined as patterns that are considered relevant for the specific task at hand. For example, in the case study focused on automating actions based on activity logs, potential patterns could include actions like “opening a web browser, performing a Google search, and opening the first result of the search” and “opening a file, modifying it, saving and closing.” To handle these data, we choose to exploit fuzzy string matching.

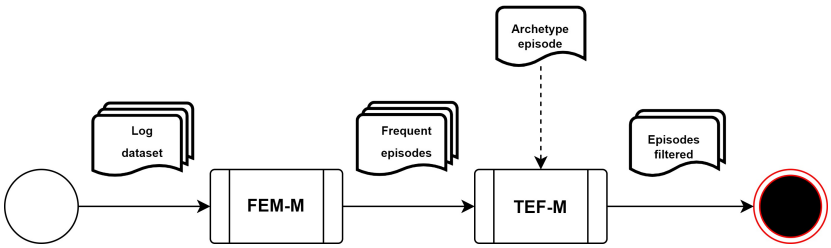

Figure 5.1: Flowchart of the proposed approach

FEM-M: Frequent Episode Miner

The approach takes as input a dataset containing sequential data, with each event associated with a timestamp. It is important so that temporal windows can be identified in the mining algorithm. The data undergoes an initial pre-processing stage. In particular, each event is associated with a positive numerical value. In terms of timestamps, the Unix timestamp convention is employed for representation. As part of the pre-processing, alternative formats are converted into Unix timestamps. Following this, the pre-processed data are utilized as input for the algorithm utilized in FEM-M.

In our experimentation, we opted for the use of MINEPI+ (Fournier-Viger et al., 2019). MINEPI+ and similar algorithms like EMMA possess a feature that makes them particularly well-suited for a wide range of problems. In comparison to other Frequent Episode Mining (FEM) algorithms, they utilize the concept of "head frequency" for calculating support (Fournier-Viger et al., 2016). To implement this algorithm, we utilized the versions available in the SPMF library[1]. These versions incorporate all the optimizations described in (Huang and Chang, 2008), which helps enhance their efficiency and performance.

The algorithm requires the configuration of just two parameters: the minimum support threshold and the maximum window size for an episode. It's worth emphasizing that when dealing with data that lacks timestamps, the window size can be readily defined based on the number of events within episodes. As a result, episodes occurring less frequently than the minimum support threshold or those exceeding the maximum temporal window are excluded from consideration during the mining process.

The output of FEM-M consists of the set of frequent episodes in the dataset. For each frequent episode, a list of the individual events that compose it is provided, along with the episode support value relative to the dataset.

[1] `https://www.philippe-fournier-viger.com/spmf/MINEPI_PLUS_EPISODE.php`

TEF-M: Target Episode Finder

The second step of the framework focuses on identifying *relevant* episodes concerning specific application goals. The TEF-M module takes as input a set of frequent episodes and an *archetype pattern*, which represents a sequence of events describing an activity of interest within the application domain. The goal is to find episodes that closely match the provided archetype pattern. The presence of such episodes with statistical significance indicates that the archetype pattern is frequently observed, making it a promising candidate for automation.

To evaluate the similarity between episodes and the archetype pattern, a specially designed distance function is utilized. More precisely, the chosen metric for refining the set of frequent episodes and selecting the episode most similar to the archetype pattern is the normalized Levenshtein distance. In this particular context, this metric quantifies the degree of fuzzy matching between two episodes and yields values within the range of $[0, 1]$. Fuzzy string matching is useful to identify similar sequences of actions, even in cases where there are variations or slight differences in the recorded data. Furthermore, it directly operates on strings, thus not requiring an additional layer of representation of episodes.

The implementation of the adopted distance has been based on the FuzzyWuzzy Python library[2].

Each episode is compared with the archetype pattern to determine their degree of similarity. An episode is considered similar if the Levenshtein similarity, i.e., $1 - \textit{Normalized Levenshtein Distance}$, exceeds a certain threshold. The actual implementation incorporates a tunable threshold parameter in the range of $[0, 1]$. When this threshold is set to 1, only episodes that are identical to the archetype pattern are chosen.

5.3 Evaluation

To assess the effectiveness of the fuzzy string-matching metrics employed in the TEF-M module, we conducted experiments using a publicly available benchmark dataset. The dataset chosen consists of public domain books that have been transformed into sequences of elements (Pokou et al., 2016), making it a suitable choice for evaluating the process automation domain. In texts, the parts of speech are structured within each sentence based on specific rules of usage, similar to how actions within process logs are structured. Additionally, the finite yet extensive number of parts of speech closely resembles the finite set of interactions in a process automation scenario. The dataset is included in the SPMF library[3] and comprises novels

[2] `https://pypi.org/project/fuzzywuzzy/`

[3] `https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`

from the 19th century in English. Each novel has been divided into sequences of words and their corresponding Part-of-Speech (PoS) tags.

For our experiments, we selected one of the available books from the repository, “A Romance of the Republic" by author Lydia Maria Child. We chose to work with the PoS-based variant, where the distinct PoS tags are considered as event types. This decision was motivated by the fact that the number of distinct parts of speech is smaller than the vocabulary of the text, making it a more suitable representation for modeling typical process automation scenarios. This is only one of the challenges that arise when working with textual data. We delve deeper into this topic in Section 5.5 and in the upcoming chapter.

The PoS tags used in our experiments are the standard ones defined in the Penn Treebank Project[4]. In this setup, the PoS tags correspond to the elements in the set of items, denoted as I for the dataset.

Consequently, we treated each PoS as an event, and we associated each event with an integer timestamp based on its position within the text. For example, the first PoS in the novel was assigned the timestamp 1, the second one received the timestamp 2, and so on. The dataset we worked with consisted of a sequence of 125395 events, corresponding to the PoS tags in the text. Subsequently, with the Frequent Episode Miner, we identified all the frequent episodes of parts of speech within the novel. In terms of parameterization, we set the minimum support threshold to 500 and the maximum window size to 10. We successfully identified 4,121 frequent episodes. An example of episode Ep: $Ep = \langle RB, DT, JJ \rangle$. This episode comprises an adverb (“RB"), followed by a determiner (“DT"), and finally, an adjective (“JJ").

To evaluate our approach using real data, we created a test set of archetype episodes. Specifically, we randomly selected 100 episodes from the output of FEM-M and made slight modifications to each episode by adding a single event (in this case, a PoS) randomly. For instance, using the previous example, the resulting modified archetype episode would be $Ep_{mod} = \langle RB, WDT, DT, JJ \rangle$. In this case, “WDT" represents a “Wh-determiner".

We then compiled an *archetype pattern list* that includes each episode from the original list, modified by adding a Wh-determiner. The primary goal of this experiment was to employ the TEF-M module using input from both the list of frequent episodes generated by FEM-M and each episode from the archetype pattern list. Subsequently, we ranked the lists of generated episodes to assess how frequently the corresponding target episode was positioned near the top of the ranking. This evaluation enabled us to assess the ability of the approach to accurately identify the similarity between an archetype pattern and a target episode.

[4] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

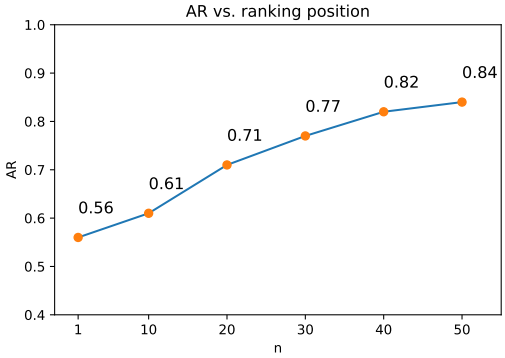

Figure 5.2: Average recall versus number of positions considered in the ranking, as mentioned in Dell'Oglio et al. (2022).

The metric we employed is the Average Recall at n (AR@n), with n corresponding to the upper limit position in the ranking. When $n = 1$, AR@1 represents the proportion of target episodes accurately retrieved in the first position. For $n = 10$, AR@10 indicates the proportion of target episodes that appear in the first ten positions, and so on for increasing values of n.

Figure 5.2 presents the results obtained for a test set of 100 episodes, displaying both specific values and the overall trend. For our 100 episodes, AR@1 is 0.56. As we increase n, AR@1 also increases. For example, AR@1 goes up from 0.56 to 0.61 when $n = 10$. Clearly, the higher the n value, the greater the likelihood of finding a target episode in the topmost n positions of the ranking. With an output containing numerous episodes, we can effectively search through them to locate the target pattern, and the results indicate that we are reasonably successful in identifying the target pattern among the top positions of the ranking.

5.4 The AUTOMIA use case

We examined an authentic dataset of activity logs, encompassing actions executed by human operators in their interactions with a computer. These tests were carried out in the context of the AUTOMIA project. The project aimed to pinpoint processes within users' activities that are automatable, specifically focusing on the identification of repetitive processes.

The data represent the interactions between a user and a computer. Within these interactions, the user undertakes tasks that are systematically logged at diverse levels of detail for distinct operations. The tracking system, an integral component of the project, comprises proprietary algorithms. Each record within the dataset pertains to an event and encompasses details such as the user's executed action,

the active operating system graphical user interface (GUI) window, accompanying metadata, and a corresponding timestamp.

An example of a few elements of the dataset is reported below.

```
...

Keyboard event
Window :66188
Keyboard key : user wrote a string
WindowName : Cerca
20220614190550

Button:mouse right down
WindowName : obs64
Window : OBS 27.2.3 (64-bit, windows) - Profilo: Senza titolo -
Scene: Senza titolo
Mouse Event: User clicked in a position
20220614190602

...
```

We evaluated our proposed method using a limited subset of archetype episodes. Initially, we employed FEM-M to extract frequent episodes from the dataset. Given the relatively modest size of the available dataset and the fact that the expected outputs are relatively short episodes, we opted to set the minimum support threshold to 4 and the maximum window width to 18. Subsequently, we utilized TEF-M to measure the degree of similarity between each archetype episode and the complete set of frequent episodes identified by FEM-M. In this instance, we aimed to isolate episodes that closely matched the archetype patterns. We set the Levenshtein similarity threshold at 0.75, with similarity defined as $1 - Levenshtein, distance$.

A brief example is given in Figure 5.3. The figure displays an archetype episode and one of the most similar retrieved episodes. In this case, the archetype episode pertains to the activity of searching and downloading resumes from a resume aggregator website, specifically IProgrammatori.[5]. It's important to note that the two episodes are nearly identical.

The results obtained were subject to manual evaluation by the project partners. This evaluation served two critical purposes: firstly, to assess the effectiveness of the method in identifying frequent episodes with potential automation relevance, and secondly, to evaluate its ability to accurately retrieve predefined patterns within the identified episodes. This evaluation has strongly confirmed the efficacy of the proposed method.

[5]https://www.iprogrammatori.it/

Archetype Episode	Retrieved Episode
Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position 20220922161135	Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position
Button:mouse left down WindowName : Chrome Legacy Window Window : IProgrammatori.it - Stampa CV - Google Chrome Mouse Event: User clicked in a position 20220922161137	Button:mouse left down WindowName : Chrome Legacy Window Window : IProgrammatori.it - Stampa CV - Google Chrome Mouse Event: User clicked in a position
Keyboard event Window : Keyboard key : user wrote a string WindowName : Save As 20220922161139	Keyboard event Window : Keyboard key : user wrote a string WindowName : Save As
Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position 20220922161151	Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position
Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position 20220922161153	Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position
Button:mouse left down WindowName : Chrome Legacy Window Window : IProgrammatori.it - Stampa CV - Google Chrome Mouse Event: User clicked in a position 20220922161157	Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position
Keyboard event Window : Keyboard key : user wrote a string WindowName : Save As 20220922161159	Button:mouse left down WindowName : Chrome Legacy Window Window : IProgrammatori.it - Stampa CV - Google Chrome Mouse Event: User clicked in a position
Button:mouse left down WindowName : Chrome Legacy Window Window : Database CV - IProgrammatori.it - Google Chrome Mouse Event: User clicked in a position 20220922161207	Button:mouse left down WindowName : IProgrammatori.it - Stampa CV - Google Chrom Window : IProgrammatori.it - Stampa CV - Google Chrome Mouse Event: User clicked in a position

Figure 5.3: An archetypal episode (left) and one of the retrieved episodes (right) with the highest similarity

5.5 Discussion

The approach described in this chapter has both strengths and weaknesses. The approach is based on tested algorithms and methods, which provides a strong baseline for its effectiveness, making it applicable across a wide range of contexts. The choice of algorithms is well-suited for several scenarios, and the flexibility of the approach allows for potential implementations of new algorithms in future iterations of the framework. Recent advancements in Episode Mining move on to novel lines of re-

search. Some of these include High-Utility Frequent Episode Mining (Wu et al., 2013), the mining of partially-ordered episode rules (Fournier-Viger et al., 2013), or the mining of the top-k high utility episodes (Rathore et al., 2016). However, it's worth noting that the employed techniques are relatively simple. While they establish a solid foundation and effectively demonstrate the feasibility of the proposed approach, it can be further refined and improved.

In terms of the scope of this work, it might be interesting to update the framework to enable the processing of textual data. Section 5.3 provided an illustrative example of utilizing the framework by inputting a novel where each sentence is divided into sequences of Part-of-Speech. To handle data at the word or sentence level, modifications to the framework are necessary. When dealing with words or sentences, the vocabulary of events grows exponentially in comparison to working exclusively with Part-of-Speech. Establishing a notably high support value becomes crucial, even if this could potentially restrict certain possibilities. The FEM-M step employs algorithms that begin by generating candidate episodes, which are frequent elements. By increasing the support (i.e., the minimum number of occurrences of a given sequence), the number of elements decreases, resulting in a narrower search space. Setting the minimum frequency high enough enhances system efficiency but also reduces a significant number of elements.

Another drawback to consider is that, unlike structured data such as log data, frequency distributions of words follow Zipf's law and are highly skewed, with few very frequent lexical items and a large number of extremely rare ones (Lenci, 2018). Zipf's law states that the frequency of a word, $F(w)$, is inversely proportional to its rank, $r(w)$, given the constants C and a:

$$F(w)\frac{C}{r(w)^a} \tag{5.1}$$

The fuzzy matching metrics itself is problematic to handle with textual data. To employ metrics and techniques capable of better modeling the semantics of the textual information becomes necessary. To address this, we can consider two strategies. First, we may enhance event descriptions by incorporating supplementary information and metadata, thus creating more robust representations for comparison using similarity metrics. Second, we can leverage learned event representations through techniques commonly applied to natural languages, such as word embeddings (Lenci, 2008).

In future advancements, the concept of *interesting text episode* as described in Chapter 2 will be further explored to enhance the utility of the proposed framework from a Natural Language Processing perspective. An initial baseline in this domain is given in the subsequent chapter, in which a simple approach to fake news detection is presented, drawing inspiration from the TEF-M module discussed in

the present chapter and the concept of information divergence given in the previous chapter.

5.6 Summary

In this Chapter, we discussed a two-step framework designed to mine sequences of actions that have the potential to be automated.

In Section 5.1 the context in which the framework was developed is presented. It is a project named AUTOMIA, which encompassed research, design, development, and experimentation of a Robotic Process Automation (RPA) system, aimed at assisting operators in executing tasks that are both highly repetitive and of low complexity. A brief definition of Robot Process Automation is given, which is a solution used in the context of the AUTOMIA project but that takes less importance in the context of the presented work. Finally, the role of the Department of Information Engineering of the University of Pisa in the project is explained.

Section 5.2 describes the proposed framework, based on two steps. The *Frequent Episode Miner* step utilizes Frequent Episode Mining algorithms for extracting all the sequences of actions that occur with at least a minimum frequency. Then, the *Target Episode Finder* exploits fuzzy string matching based on the Levenshtein distance for filtering out the sequences that do not match established patterns.

Section 5.3 shows the evaluation of the proposed framework and, finally, in Section 5.5 the strengths and weaknesses of the proposed framework are discussed. Also, future directions are proposed. One important concept to highlight is that the actual framework is not able to generalize from structured sequence data such as log data to unstructured data such as textual ones. Improvements with a mixture of data mining techniques and NLP approaches will be further employed.

Chapter 6

Case Study: Fake and Real news dataset collection and Fake News Detection with Information Divergence

The problem of fake news detection remains still debated within the research community. It has received considerable diffusion in recent years, largely driven by the uncontrolled proliferation of fake news across the internet and social media platforms, particularly during critical and dramatic events such as the COVID-19 pandemic and the Ukraine-Russia conflict. The very definition of fake news continues to be a matter of debate, as explored in Chapter 3, since it encompasses a significant range of variations, including rumors, hoaxes, clickbait, and more. Fake news impacts areas like information reliability and social media interactions.

Several definitions have been proposed for fake news and rumors. In this work, we adhere to the comprehensive definition presented in Bondielli and Marcelloni (2019), which encompasses the following characteristics. Firstly, fake news refers to articles that are intentionally and verifiably false (Allcott and Gentzkow, 2017). Thus, a key factor is their form as news articles, while a second important aspect is related to the properties of intention and verifiability. Fake news is, in essence, intentionally false content that can be confirmed as such, and that can potentially mislead readers. A third noteworthy characteristic is that their primary mean of dissemination is social media platforms (Bondielli and Marcelloni, 2019).

Fake news and rumors are often disseminated along with events relevant to public opinion. Notably, these events encompass newsworthy occurrences, as exemplified in Zubiaga et al. (2018b). Two illustrative instances of such events are the aforementioned COVID-19 pandemic and the Ukraine-Russia conflict. An essential strategy in fighting the proliferation of fake news on social media is to consider how

they relate to genuine, verified news concerning these pivotal events.

The majority of research in Fake News Detection approaches the problem as a standard binary or multi-class classification task. Supervised models for classification are usually trained on datasets containing fake and real news or rumor and non-rumor social media posts, utilizing content-based or context-based features. It is worth emphasizing the significant key limitations of this approach.

Classification-based approaches do not consider how to debunk fake news or how to verify the information contained in real ones. After being trained on relevant data, classifiers can reasonably predict whether a piece of news on a particular topic is real or fake or whether it's a rumor or not. However, they may not address the fundamental issue of confirming the accuracy of the information. Nevertheless, such information could potentially prove invaluable, serving both research purposes and the needs of everyday users.

One potential solution to this problem could involve addressing the tasks of fake news and rumor detection while incorporating methods for performing fact-checking. Fake news detection can be viewed as the inverse of fact-checking. While the goal of fact-checking is to identify information supported by verified facts, fake news, on the other hand, represents the exact opposite. A piece of fake news can be considered as news that lacks factual support. Thus, it may be interesting to explore the characteristics of fact-checking systems and integrate their principles into a fake news detection system. In particular, instead of aiming to definitively prove something as false, it might be feasible to approach the task by recognizing information that is substantiated by facts and consider other candidates as potentially fake. Recently, computational-oriented fact-checking has garnered significant attention. Researchers have begun to develop and train automatic fact-checking systems, such as Passaro et al. (2020) and Shaar et al. (2021).

Another drawback to the classification approach in the Fake News Detection problem is that key attributes of emerging fake news or rumors are greatly influenced by the breaking news story with which they are associated. From a Data Mining perspective, fake news can be viewed as an *interesting episode*, as previously defined in Chapter 2. The entire breaking news story constitutes the sequence S and encompasses the collection of all the scrutinized articles or social media posts. An episode is represented by an individual article or social media post, comprising a sequence of sentences conveying information. Given a manually collected set of verified fake news and misleading claims, a fake news item (i.e., an interesting text episode) is an article (i.e., an event) that is similar to one or more of the verified fake news. Conversely, with a collected set of verified real news, a fake news item (i.e., an interesting text episode) is an article (i.e., an event) that exhibits a substantial divergence from one or more of the verified real news.

The goal of this chapter is threefold. First, we propose a topic-oriented methodol-

ogy for collecting and labelling potentially fake and verified news from social media for a given event. This approach aims to obtain real-world datasets that encompass both real and fake news related to specific events, facilitating real-world-focused analysis. Second, we adapt the methodology to collect a second dataset that includes both textual and visual elements, which is subsequently made available within the context of the EVALITA Challenge known as MULTI-Fake-DetectiVE. The systems proposed by the participants are then analyzed, and building upon the results obtained by the participants in EVALITA, we employ a recent technique called Prompt Tuning to continue the trajectory initiated by them. Third, we adapt the similarity-based method described in Chapter 4 and the episode mining method in Chapter 5 to address the task. Finally, we provide an analysis of the results.

6.1 A novel dataset strategy collection: the Notre Dame case

Most current methods for fake news and rumor detection consider the issue as a classification task, often utilizing limited datasets. There have been some efforts aimed at constructing extensive datasets, which we described in detail in Chapter 3. However, this field still lacks a high-quality benchmark dataset (Shu et al., 2017)

Three key issues need to be addressed regarding the available fake news datasets.

First, most of these datasets are not primarily designed as resources for solving Fake News Detection tasks. They are often developed for different specific purposes, such as fact-checking tasks or evaluating text credibility, and are subsequently reused for fake news detection. This makes them insufficient and not comprehensive enough, particularly in recent years when the prevalence of fake news has increased, and they have become more realistic, similar to real ones.

Second, many approaches primarily focus on classifying real and fake news in isolation. They rely on the surface or textual properties and the configuration of their diffusion patterns. However, these approaches often do not take into consideration the real news surrounding fake news. This choice can limit the generalization capabilities of the systems. Such systems do not learn how to identify when a fake news occurrence emerges in relation to real news or how to reason about fake news. Instead, their primary objective is merely to differentiate between the characteristics of fake and real news within the context of the provided dataset. Consequently, these systems may exhibit a higher susceptibility to concept drift and may struggle to generalize the concept of fake and real news in real-world scenarios.

Third, it is worth recognizing that there are several types of fake news, each with varying degrees of deceitfulness. While some may be easily identifiable based on their surface properties, the primary goal of fake news is to deceive readers. Most of

them closely imitate the style and language of real news articles and are built on false yet plausible information regarding real-world events. Detecting these types of fake news requires more in-depth text understanding abilities and a broader knowledge of the news story associated with the fake news.

In the following, we introduce a hybrid methodology designed for the collection and labelling of news articles related to specific topics and subjects (Passaro et al., 2022). Our proposed approach focuses on gathering data from social media platforms concerning real-world events that are known to be susceptible to the dissemination of fake news. To achieve this, we present a labelling method that leverages crowdsourcing. This method is fast, reliable, and reasonably accurate in approximating expert human annotation. Our approach takes into account both the content of the data, including the text of the articles, and contextual information related to fake news within the context of a particular real-world event. The underlying principle of this approach is that including contextual knowledge in a crowdsourcing experiment is crucial for closely approximating expert human annotation.

We have applied this methodology to collect and annotate two datasets: the Notre-Dame Fire Dataset and the MULTI-Fake-DetectiVE dataset. The latter was part of our participation in organizing the EVALITA challenge.

Data Collection

The dataset is collected from social media posts, given a specific event. More precisely, since social media have been proven to be the most fertile ground for the spread of fake or unverified information (Zubiaga et al., 2018b), we choose to collect social media posts and news articles linked in such posts. In terms of practical implementation, Twitter was chosen as the social media platform of reference, because it is one of the most widely studied social networks, particularly in the context of rumors and fake news. Furthermore, at the time we were carrying out this work, Twitter imposed fewer restrictions on researchers regarding the quantity and quality of collected information, enhancing the feasibility of data acquisition.

We begin the process by identifying and collecting a set of posts related to a specific subject, by searching for specific hashtags and keywords associated with the target event. From the complete dataset, we eliminate posts that are directly shared by international newspapers and news agencies, specifically those originating from their official accounts. However, if these news items are subsequently retweeted or shared by other users, they are retained in the dataset and subjected to subsequent annotation and verification. All the retained tweets and news articles, which are considered to potentially contain fake news, form the raw data for the topic-specific dataset.

Data Annotation

The labelling process involves the assignment of *Real* and *Fake* labels to a set of tweets and articles. To achieve this, we employ two distinct annotations. The objectives are twofold: first, to acquire a definitive gold standard annotation for distinguishing between fake and real news, and second, to gain insights into the level of difficulty associated with detecting fake news in the absence of contextual knowledge. This operational approach is executed through the execution of two separate crowdsourcing tasks, namely respectively *In-Context Annotation* and *Out-of-Context Annotation*. The two tasks are structured as follows:

In-Context Annotation: Participants in the crowdsourcing experiment are asked to label posts and articles as fake or real news. To obtain a gold labelling, annotators are provided with a manually selected list of known false news. The list includes the initial source tweets or articles that initiated the fake news, as well as concise descriptions of the fake news stories. These fake news are presumed to have been definitively debunked, confirming their falsehood. Our intention in furnishing annotators with contextual information about specific fake posts is to enhance their ability to identify fake news in a broader sense. This includes not only the more straightforward cases but also the more intricate and challenging instances of fake news that may be harder to discern. Formally, given a piece of text t (article or post) related to a topic or story s (event or public figure), and a list F_s of known fake news related to s, participants are asked to decide whether t is fake or real.

Out-of-Context Annotation: In this scenario, participants are not provided with a list of known fake or real news. Instead, they are tasked with labelling the content as either real or fake solely based on their judgment, without any accompanying supporting information. This annotation establishes a baseline that operates under the assumption that the participants have no prior knowledge regarding the event or fake news being presented to them. Formally, given a piece of text t (article or post) related to the topic or story s (event or public figure), participants are asked to decide whether t is real or fake, without any supplementary or additional supporting information.

In addition to the two initial annotations, we conducted a third annotation to fact-check each piece of content. To perform this annotation we had unrestricted access to time and resources, including the web.

The Notre Dame Fire dataset

The Notre Dame Fire dataset represents the first dataset we collected using the proposed methodology. We chose the Notre Dame fire incident from April 2019 be-

cause it served as a trigger for the proliferation of fake news on social media. During this period, a substantial amount of false information circulated on social media, attributing blame for the fire either to the yellow vests or Islamic extremists, while official sources consistently maintained that the fire was not deliberately set.

Table 6.1: Verified fake news collected for the Notre Dame Fire Event.

Context	Fake News
A person working on the Notre Dame cathedral at the moment of the fire claims that it was deliberately started and not an accident.	"Notre Dame cathedral in Paris on fire, worker claims it was deliberately started"
A fake account for CNN stated that the Notre Dame fire was a terroristic attack.	"CNN can now confirm the Notre Dame fire was caused by an act of terrorism"
A tweet claimed that an unmarked car was found in Paris with gas tanks and Arabic documents inside. The news is actually real, but is from 2015, and does not involve Notre Dame.	"Gas tanks and Arabic documents found in unmarked car by Paris"
A Muslim girl was allegedly plotting to blow up a car using gas canisters near the Notre Dame cathedral for love.	"Muslim girl in search of love plotted to blow up car packed with gas canisters near Notre Dame Cathedral"
A fake Fox News account presented a fabricated tweet from Rep. Ilhan Omar (a Muslim American politician) saying "They reap what they sow" in reference to Notre Dame.	"Ilhan Omar - They reap what they sow #NotreDame"
An account tweeted a video of Notre Dame burning with shouts of "Allahu Akbar" edited over the video. Other similar videos were shared after.	"Who yells Allah Ahkbar when they see an 850 yr old beloved and cherished Catholic Religious Monument that has a roaring, blazing fire coming from it?"
A low-quality video of a person walking on the outside of the cathedral was used to make false claims about the fire, including that the person shown was an "Imam" or a Yellow Vest protester setting the fire.	"Notre-Dame: who is this person in yellow vest on a tower?"

The Notre-Dame Fire (NDF) Dataset [1] includes news articles and tweets produced during the Notre-Dame fire of April 2019 (Passaro et al., 2022). NDF is a subset of the whole dataset about the topic, obtained and labelled via crowdsourcing on Prolific according to the methodology proposed. Specifically, all contents in the NDF were annotated twice via crowdsourcing, exploiting the out-of-context and the in-context strategy for the annotation, and subsequently manually fact-checked by us. This process resulted in a dataset annotated at three distinct levels: In-Context annotation, Out-of-Context annotation, and manual annotation. The final label for each item and task is determined by the majority vote (in the case of the crowd-

[1]The dataset is available at `https://github.com/Unipisa/NDFDataset`.

sourcing experiments) or the decision from manual fact-checking (for the manually fact-checked annotation).

Both the OOC and IC annotation tasks involved 20 participants each.

The two annotation methods produced different outcomes when assigning the final class (fake or real) to each text piece. To determine the final class, we relied on a majority vote. However, in certain cases, the assignment was uncertain due to the even number of participants for each text (10 ratings per class). This suggests that definitively determining whether a given text is disseminating fake news or not based solely on the available data can be challenging.

The agreement among the Prolific annotators is evaluated using Fleiss' Kappa (Fleiss, 1971). It is a measure utilized to assess the agreement between two or more raters when assigning categorical labels to a set of items. For Task 1, the agreement is 0.47. In Task 2, the agreement is 0.24. The observed low agreement can be attributed to both the relatively high number of annotators and the inherent complexity of the Annotation task, which is somewhat subjective in nature. Nevertheless, the results are considered promising. The lower agreement between annotators observed in the second task can be primarily attributed to its higher complexity. In fact, in this task, annotators were not provided with any ground truth for the annotation.

The final datasets include 578 texts, where 87 are newspaper articles and 481 are tweets. Figure 6.1 reports the class distribution across the annotation tasks. The real news is represented by the blue regions, while the fake news is represented by the orange areas. The data points with an uncertain majority vote in a specific rating task are represented by the gray regions.

Evaluation of the methodology

Several experiments were conducted to assess the effectiveness of the annotation methodology. They were explained in an in-depth analysis in Passaro et al. (2022). In the present work, we discuss them shortly, focusing on the impact of the different annotation methods on the performance of state-of-the-art classifiers using the Notre-Dame Fire dataset, and on what emerges by analyzing the results obtained with the experimentation of the IC method on the PHEME dataset. This last experiment allows us to evaluate its robustness and effectiveness in contexts and events that diverge from the ones already discussed.

Performance on BERT

Two BERT (Devlin et al., 2018) classifiers were fine-tuned, the first using the OOC dataset and the second employing the IC dataset. Their performances are then evaluated to predict the labels of the manual fact-checked annotation. This evaluation

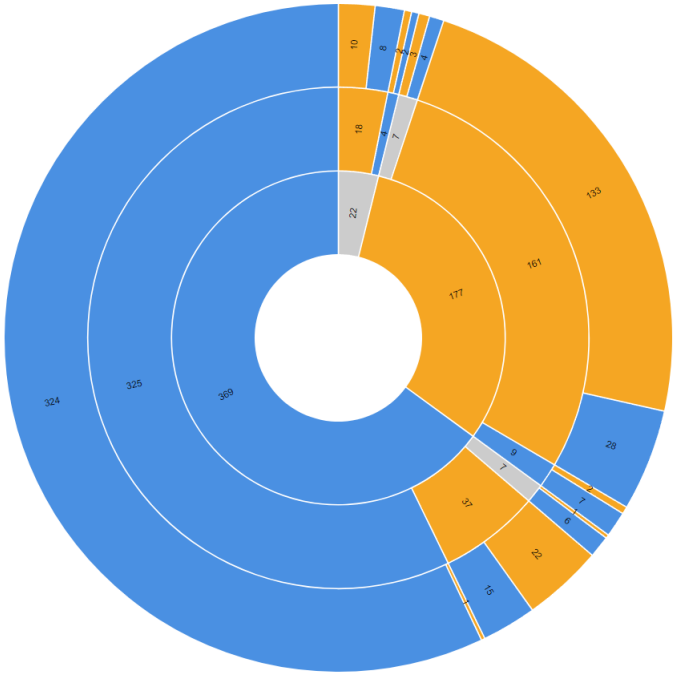

Figure 6.1: The class distribution across the annotation tasks as mentioned in Passaro et al. (2022). The rings correspond to the out-of-context (OOC), in-context (IC), and manually-fact-checked (MFC) annotation levels, from the innermost to the outermost.

allows us to evaluate the extent to which the IC annotation methodology can approximate a manually fact-checked annotation.

We performed 10-fold cross-validation to ensure that the proportion between instances that belong to Fake and Real classes is similar in each fold.

The pre-trained model used is `bert-base-uncased`. We used a batch size of 8 due to computational limitations and the learning rate was set to 2e-5 following (Devlin et al., 2018). For the other parameters, we maintained the default configuration in the huggingface[2] implementation.

The performances obtained for each classifier fine-tuned with the OOC and IC datasets are shown in Tables 6.2 and 6.3. The overall accuracy is comparable for both classifiers, but there are notable differences in their ability to identify fake news, specifically in terms of the recall on the Fake class).

An additional experiment is performed by adding for each training fold 7 instances belonging to the Fake class that correspond to the context used by the raters in the IC rating task. The fine-tuning with the IC dataset determines an improvement in the classifier performance. Conversely, in the case of the OOC dataset, the performances degrade. This could be attributed to the different linguistic cues and

[2] `https://huggingface.co/`

accuracy	0.823
weighted avg precision	0.831
weighted avg recall	0.823
weighted avg f1-score	0.823
macro avg precision	0.786
macro avg recall	0.783
macro avg f1-score	0.779
class Real precision	0.881
class Real recall	0.873
class Real f1-score	0.875
class Fake precision	0.692
class Fake recall	0.693
class Fake f1-score	0.682

Table 6.2: Performance obtained by the BERT classifier trained with the OOC annotated dataset.

accuracy	0.824
weighted avg precision	0.843
weighted avg recall	0.824
weighted avg f1-score	0.828
macro avg precision	0.792
macro avg recall	0.817
macro avg f1-score	0.797
class Real precision	0.911
class Real recall	0.835
class Real f1-score	0.870
class Fake precision	0.673
class Fake recall	0.799
class Fake f1-score	0.725

Table 6.3: Performance obtained by the BERT classifier trained with the IC annotated dataset.

complexity of the data points already present in the training set. As a result, the classifier exhibited a decrease in its generalization capability for the fake news class. The results obtained are shown in Tables 6.4 and 6.5.

Evaluation on a simulated dataset

In order to assess the effectiveness of the IC methodology, we applied it in a different context and with a different event. To the best of our knowledge, at the time we conducted these experiments, no other datasets focused on real-world events were available. Hence, we decided to simulate our target real-world scenario by re-annotating a dataset that already existed. We used the PHEME dataset, originally gathered for rumor detection and veracity classification (Zubiaga et al., 2016a, 2017). PHEME contains *Twitter conversation threads* related to nine real-world events. Each *conversation* is based on a source tweet that propagates a rumor, and its responses, supporting, or denying it. Each instance includes a binary class (rumor, non-rumor), and all the rumors are clustered into *stories* by expert journalists, each classified as true, false, or unverified (Zubiaga et al., 2016a, 2017).

We selected the tweets that belong to a specific event included in the PHEME

accuracy	0.804
weighted avg precision	0.818
weighted avg recall	0.804
weighted avg f1-score	0.8
macro avg precision	0.768
macro avg recall	0.766
macro avg f1-score	0.752
class Real precision	0.871
class Real recall	0.862
class Real f1-score	0.862
class Fake precision	0.664
class Fake recall	0.671
class Fake f1-score	0.643

Table 6.4: Performance obtained by the BERT classifier trained with the OOC dataset + seven known fake news.

accuracy	0.849
weighted avg precision	0.865
weighted avg recall	0.849
weighted avg f1-score	0.852
macro avg precision	0.817
macro avg recall	0.848
macro avg f1-score	0.826
class Real precision	0.933
class Real recall	0.847
class Real f1-score	0.887
class Fake precision	0.7
class Fake recall	0.849
class Fake f1-score	0.765

Table 6.5: Performance obtained by the BERT classifier trained with the IC dataset + seven known fake news.

dataset: the Charlie Hebdo attack of January 7, 2015. The categories included in the dataset are then used to infer the verified fake news (i.e. the context for the annotators). We provided the annotators with 9 context fake news, and we asked them to annotate 262 tweets. To ensure representative samples for both real and fake news, we selected real examples from non-rumor tweets, and fake ones from all the rumors that were verified as fake.

We collected the ratings through a Prolific experiment and then we fine-tuned a BERT (Devlin et al., 2018) classifier with the same parameters as the previous experiments. In the following, we refer to this annotation as PHEME-IC. We also collected a PHEME-OOC following the same procedure described in 6.1.

As for the first experiments, we used PHEME-IC and PHEME-OOC annotated data to fine-tune the classifier and we challenged the system to predict the original PHEME labels. Tables 6.6 and 6.7 report the obtained results.

To obtain manually fact-checked labels similar to those in the NDF dataset, we conducted an additional round of manual annotation. In a similar way to the manual annotation process for the NDF dataset, the use of additional web sources is allowed. This additional annotation is referred to as PHEME-MFC. The performances of the two classifiers in predicting the PHEME-MFC labels are reported in Tables 6.8 and

accuracy	0.642
weighted avg precision	0.419
weighted avg recall	0.642
weighted avg f1-score	0.506
macro avg precision	0.321
macro avg recall	0.500
macro avg f1-score	0.389
class Real precision	0.642
class Real recall	1.000
class Real f1-score	0.779
class Fake precision	0.000
class Fake recall	0.000
class Fake f1-score	0.000

Table 6.6: Performance obtained by the BERT classifier trained on PHEME-OOC and tested on the PHEME original labels.

accuracy	0.727
weighted avg precision	0.752
weighted avg recall	0.727
weighted avg f1-score	0.724
macro avg precision	0.717
macro avg recall	0.705
macro avg f1-score	0.695
class Real precision	0.813
class Real recall	0.753
class Real f1-score	0.768
class Fake precision	0.620
class Fake recall	0.658
class Fake f1-score	0.623

Table 6.7: Performance obtained by the BERT classifier trained on PHEME-IC and tested on the PHEME original labels.

6.9.

The results indicate that the OOC annotation method is inadequate for the task. This observation is consistent with the findings from the previous experiment. The reason is that the IC annotation scheme, where raters are provided with more contextual information about potential fake news, aids in generating more consistent labels compared to the dataset labelled by the expert. This contributes to improved classifier performance in the "Fake" class, a pattern evident in both the NDF and PHEME datasets.

6.2 The application of the strategy in a realword scenario: MULTI-Fake-DetectiVE

The problem of fake news has gained significant attention in recent years, leading to the development of several approaches that focus on different aspects, including texts, social network structures, temporal information, or a combination of these factors (Bozarth and Budak, 2020). Text is the most common medium for spreading

accuracy	0.481
weighted avg precision	0.240
weighted avg recall	0.481
weighted avg f1-score	0.318
macro avg precision	0.240
macro avg recall	0.500
macro avg f1-score	0.322
class Real precision	0.481
class Real recall	1.000
class Real f1-score	0.645
class Fake precision	0.000
class Fake recall	0.000
class Fake f1-score	0.000

Table 6.8: Performance obtained by the BERT classifier trained on PHEME-OOC and tested on PHEME-MFC.

accuracy	0.743
weighted avg precision	0.781
weighted avg recall	0.743
weighted avg f1-score	0.734
macro avg precision	0.774
macro avg recall	0.739
macro avg f1-score	0.727
class Real precision	0.710
class Real recall	0.844
class Real f1-score	0.757
class Fake precision	0.838
class Fake recall	0.635
class Fake f1-score	0.698

Table 6.9: Performance obtained by the BERT classifier trained on PHEME-IC and tested on PHEME-MFC.

disinformation. However, the inclusion of images in misleading content can enhance the perceived credibility of the text, as images can be manipulated to deceive readers or attract a larger audience on social media.

Despite the potential of different modalities in influencing the spread of disinformation and fake news, the role of multimodality has not received as much attention (Alam et al., 2021). While some efforts have been made in this direction (Dimitrov et al., 2021; Kiela et al., 2020) creating models that effectively combine multiple modalities for fake news detection remains a significant challenge in the field. Additionally, there's a need for datasets that encompass various modalities and different sources of fake news (Alam et al., 2021).

We created two datasets of fake news that include both textual and visual content. To collect and label data, we applied the IC methodology previously described. Both datasets are collected and made publicly available in the context of the shared task, named MULTI-Fake-DetectiVE, proposed during the EVALITA 2023 Evaluation Campaign. The task focuses on the automatic detection of fake news in a multimodal setting including texts and images. It is divided into two specific subtasks.

Subtask 1: Multimodal Fake News Detection

The first subtask was structured as a multi-class classification problem in a multimodal setting, defined as follows: given a piece of content $c = \langle t, v \rangle$ which includes a textual element t and a visual element v (i.e., an image), classify it as being one of the following labels on a scale: "*Certainly Fake*", "*Probably Fake*", "*Probably Real*", "*Certainly Real*". The labels refer to the entire content, and not to the single element (i.e. text or image). A piece of news could still probably (or certainly) be fake news even if it includes a real image (for example, in a misleading context).

The labels are defined as follows:

***Certainly Fake*:** news that is most certain to be fake, whatever the context.

***Probably Fake*:** news that is still likely to be fake, but may include some real information or at the very least be somewhat credible.

***Probably Real*:** news that is very credible but still retains some degree of uncertainty regarding the information provided.

***Certainly Real*:** news that is most certain to be real and incontestable, whatever the context.

Either of the two components, or both, can be leveraged to make the final prediction. Participants are encouraged to develop multimodal models for the task.

The dataset collected focuses on the Ukrainian-Russian war and includes social media posts and news articles, containing both textual and visual components in a time frame going from February 2022 to December 2022, and is split into a training set and two test sets:

Training Set: It includes data from February 2022 to September 2022.

Test Set (Official): It includes data from October 2022 to December 2022. It was created to provide a more realistic and challenging scenario for evaluating the performance of participating systems in classifying fake news and misleading content. Indeed, different time frames can determine different data distributions.

Test Set (Additional): an additional batch of test data including data from the same time window as the training set. It was created to give a picture of how participating systems are adaptable to changes in context over time.

	C.F.	P.F.	P.R.	C.R.
Train	153	219	476	199
Test (official)	16	52	106	21
Test (additional)	27	58	101	32
Total	196	329	683	252

Table 6.10: Dataset size for sub-task 1.

	Misl.	Not Misl.	Unrel.
Train	373	546	417
Test (official)	45	75	99
Test (additional)	67	84	89
Total	**485**	**705**	**605**

Table 6.11: Dataset size for sub-task 2.

Subtask 2: Cross-modal relations in Fake and Real News

The second subtask was aimed at assessing how the textual and the visual elements relate to each other in the context of fake and real news. We aimed to understand how multimodality in fake and real news can influence the interpretation of the content relating to specific modalities, and the whole news.

The task was formulated as a three-class classification problem, and is defined as follows: given a piece of content $c = \langle t, v \rangle$ which includes a textual element t and a visual element v, determine whether their combination is *misleading* or *not misleading* in the interpretation of the information given by their elements, or the two are *unrelated*. The three classes are defined as follows:

***Misleading*:** The image or the text is misleading of the information in the other modality or overall.

***Not Misleading*:** The image and the text are related to each other, support the overall information provided, and are not used with misleading intent.

***Unrelated*:** The image and the text are unrelated to each other in a meaningful way. This absence of a relationship does not result in varied interpretations of the information in any way.

The dataset focuses on the same event as the first subtask, i.e., the Ukrainian-Russian war, at the same time frame going from February 2022 to December 2022. It is split into a training set and two test sets in a similar way.

Tables 6.10 and 6.11 report sizes and class distribution of the dataset for the first and second subtasks. In the first one, we computed the average Spearman corre-

Rank	TEAM-RUN	F1-Score
1	**Polito-P1**	**0.512**
2	extremITA-camoscio_lora	0.507
3	AIMH-MYPRIMARYRUN	0.488
4	*Baseline-SVM_TEXT*	0.479
5	*Baseline-SVM_MULTI*	0.463
6	*Baseline-MLP_TEXT*	0.448
7	*Baseline-MLP_IMAGE*	0.402
8	HIJLI-JU-CLEF-Multi	0.393
9	*Baseline-SVM_IMAGE*	0.386
10	*Baseline-MLP_MULTI*	0.374

Table 6.12: Sub-task 1 - Official Test Set.

Rank	TEAM-RUN	F1-Score
1	**extremITA-camoscio_lora**	**0.464**
2	PoliTo	0.460
3	extremITA-it5	0.348

Table 6.13: Sub-task 1 - Additional Test Set.

lation coefficient between annotator pairs, as inter-annotator agreement, because of the ordered nature of the labels, obtaining a correlation of 0.43 ($\sigma = 0.04$). In the second sub-task, we instead employed Fleiss' Kappa obtaining $k = 0.25$

Participants to the task

A total of four teams with their systems participated in MULTI-Fake-DetectiVE. All of them participated in sub-task 1, and only two also participated in sub-task 2. Two proposed approaches are truly multimodal, and are Polito (D'Amico et al., 2023), and AIMH (Puccetti and Esuli, 2023), and the other two are text-oriented: ExtremITA (Hromei et al., 2023), and HIJLI-JU-CLEF (Sarkar et al., 2023). All of them were evaluated against each a set of baseline models, which were a Support Vector Machine (SVM) and a Multi-Layer Perceptron (MLP), with three different feature sets as the baseline models:

Text-only features extracted with a multilingual BERT model (Devlin et al., 2018).

Image-only features extracted with ResNet-18 (He et al., 2016).

Multimodal features obtained by concatenating the two type of features.

Tables 6.12 and 6.13 show the results of each system and baselines on the Official and Additional test sets, for the sub-task 1. The results obtained by the two

Rank	TEAM-RUN	F1-Score
1	**Polito-P1**	**0.517**
2	*Baseline-MLP-TEXT*	0.506
3	*Baseline-SVM-TEXT*	0.482
4	*Baseline-MLP-MULTI*	0.461
5	*Baseline-SVM-MULTI*	0.442
6	*Baseline-SVM-IMAGE*	0.436
7	AIMH-MYPRIMARYSUB	0.421
8	*Baseline-MLP-IMAGE*	0.373

Table 6.14: Sub-task 2 - Official Test Set.

systems that participated in the Official test set also for sub-task 2 are detailed in Table 6.14. Only the PoliTo participated in the Additional evaluation, obtaining a weighted average F1-Score of 0.51.

These results suggest that multimodal fake news detection and cross-modal analysis of images and texts in the context of fake news are two rather challenging tasks, both for participants and annotators, as the agreement metrics demonstrate. Even the best-performing systems were not able to significantly improve over the baseline model results.

In sub-task 1, only two out of the four participating systems could be regarded as truly multimodal, as they explicitly incorporate image-level features by leveraging an image encoder model. These systems have a similar architecture, both employing a dual encoder architecture with a classification layer (Gan et al., 2022). Both approaches chose to use a ViT image encoder but trained on different data. For text encoding, the AIMH system employs RoBERTa for English translations and an Italian version of BERT for original texts. The PoliTo team uses the FND-CLIP text encoder, which is based on GPT. They also propose extensions, by including sentiment-aware text features and image transformations, reporting increasing performance with all of them, with the ensemble classifier performing best (D'Amico et al., 2023)

The last two systems are both text-only. HIJLI-JU-CLEF considers the automatically generated caption of the images, shifting the problem to a text-only scenario. ExtremIta is a text-only architecture based on LLaMA, using a few-shot prompting approach.

The results of sub-task 1 do not highlight an evident advantage of a multimodal approach over a uni-modal one. Two out of the three models that gain an outperforming score with respect to all the baselines are multimodal. However, the best model was the text-only based on LLaMA.

While using both images and text can help identify fake or real news, it seems that a large part of the key information that solves the task is contained in the textual element. If we assume this, it becomes easier to understand how model scale also

plays a crucial role in performances. It is underlined by the fact that only the Large Langauge Model (LLM), among the participants, gets a score close to the performances of more refined approaches in a few-shot setting via prompting.

Experiments

The emergence and spread of Large Language Models certainly cannot be ignored, and it would be interesting to explore a scenario other than the zero-shot one.

In the following we present preliminary experiments conducted using the MULTI-Fake-Detective dataset, employing the prompt tuning technique with a pre-trained BERT model. Note that due to computational and time constraints, we employed a smaller model. The following results offer an initial insight into the performance of the prompt tuning technique when applied to Fake News Detection, treated as a multi-class classification task. These results can be compared with the previously mentioned baselines and serve as a baseline for future research in this field. Our future endeavors will involve exploring larger models, specifically large pre-trained Vision-Language models augmented with additional features.

Table 6.15 reports the performance in terms of F1-score for all the experiments conducted. It is evident that the results, while not exceptionally high, align with the EVALITA task baselines.

Table 6.15: Prompt tuning result

Model	Test	Additional Test
Normal	0.46	0.46
Initial text prompt	0.37	0.32
Initial text prompt with source	0.35	0.36
Daataset improved with source	0.41	0.37

Respectively, from the first to the last row in the Table, the experiments performed are:

- Prompt Tuning is performed by initializing the configuration of the prompt to learn with 30 virtual tokens and, as initial text: "Classify if the text is Certainly Fake (0), Probably Fake (1), Probably True (2), Certainly True (3):". The classifier is evaluated with both the test sets (normal and additional).
- Prompt Tuning is performed by initializing the configuration of the prompt to learn with 150 virtual tokens and, in conjunction with the previous text, a list

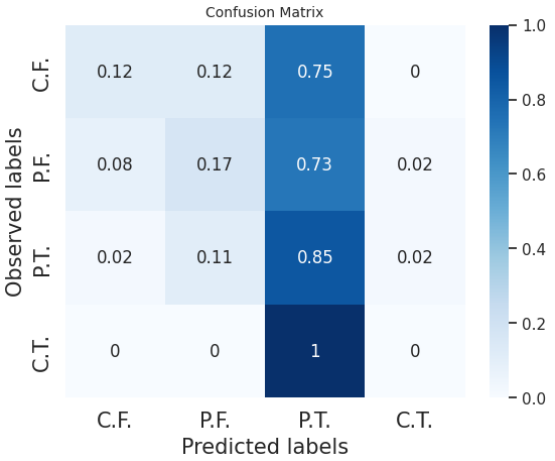

Figure 6.2: Normal Standard

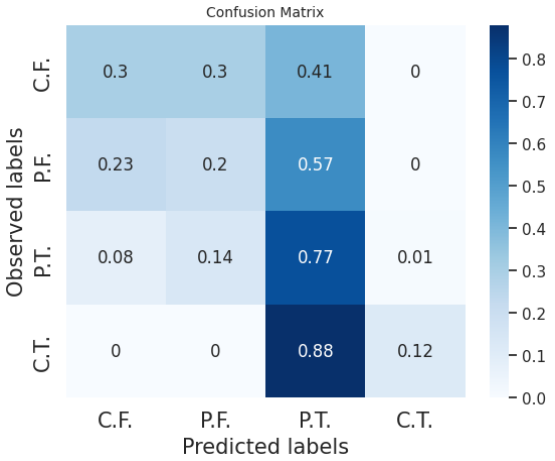

Figure 6.3: Normal Additional

of all the descriptions of the verified fake news (i.e. the fake news utilized by the annotators as context) is provided.

- Prompt Tuning is performed by initializing the configuration of the prompt to learn with 150 virtual tokens and, in conjunction with the previous text, a list of all the source texts of the verified fake news (i.e. the fake news utilized by the annotators as context) is provided.

- Prompt Tuning is performed by initializing the configuration of the prompt to learn with 30 virtual tokens and all the source texts of the verified fake news (i.e. the fake news utilized by the annotators as context) are added in the training set, with the class "0" (i.e. "Certainly Fake").

We observe that the overall performance is relatively low, with the highest F1-score achieved by the first classifier, initialized only with the basic prompt. It appears that as the complexity of the initial text increases, the model's classification accuracy decreases.

As for the performance on the specific classes, the confusion matrices for all the experiments are reported in Figures 6.2 and 6.3 for the first experiment, in Figures 6.4 and 6.5 for the second experiment, in Figures 6.6 and 6.7 for the third experiment, and finally in Figures 6.8 and 6.9 for the last experiments. All the confusion matrices provide evidence that the easiest class to predict is the 'Probably Real' one, with most of the other classes being frequently confused with it.

A noticeable trend indicates an increase in the classification of the 'Certainly Fake' class in the additional test set. This is likely because this test set was collected within the same time frame as the training set, which probably contains more similar fake content. Our findings suggest that the problem remains open, and moving

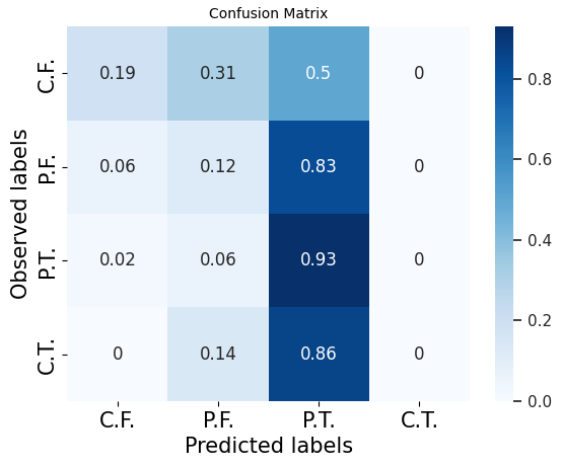

Figure 6.4: Prompt description Standard

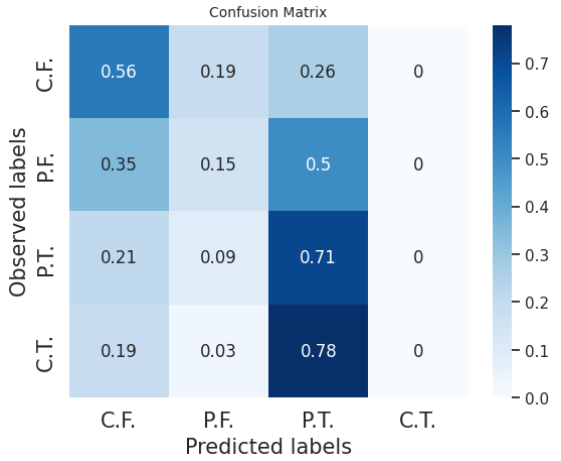

Figure 6.5: Prompt description Additional

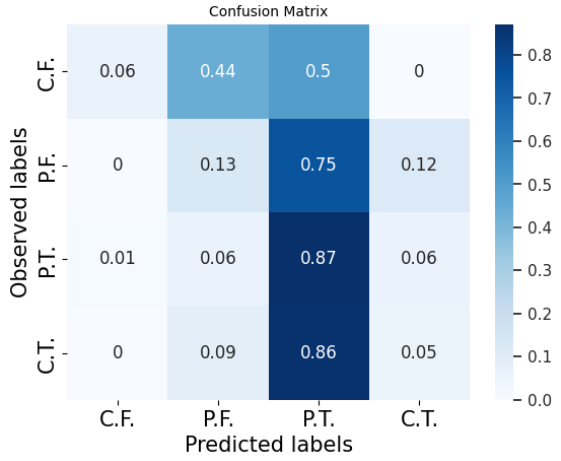

Figure 6.6: Prompt Source Standard

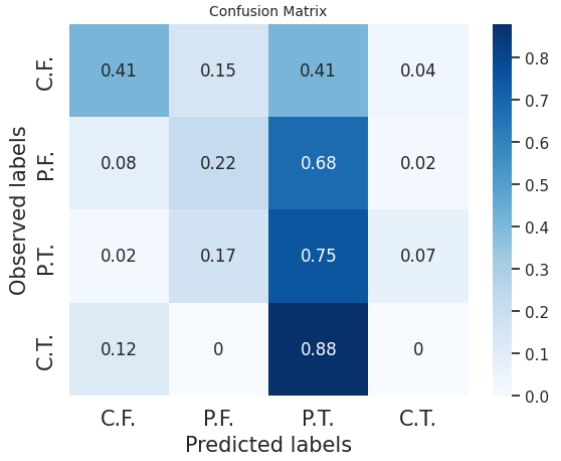

Figure 6.7: Prompt Source Additional

forward, it may be valuable to combine the strengths of the best-performing approaches. For instance, we could focus on utilizing large pre-trained Vision-Language models augmented with additional features, such as available emotive resources (Passaro and Lenci, 2016), either through fine-tuning or appropriate prompt tuning/engineering.

Given the ongoing challenges posed by deceptive or misleading content on social media, we believe that an effective understanding and modeling of this complex problem can be highly beneficial in fighting online disinformation.

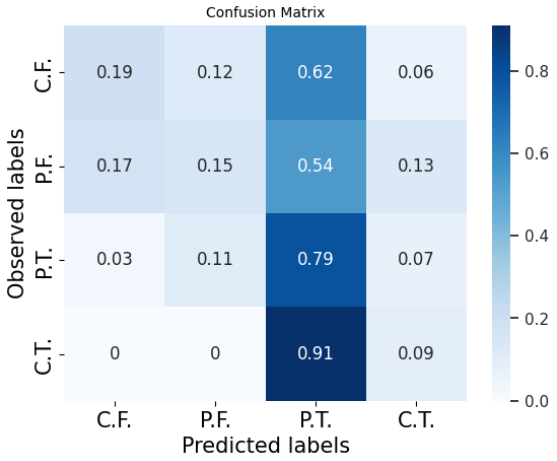

Figure 6.8: Dataset Source Standard

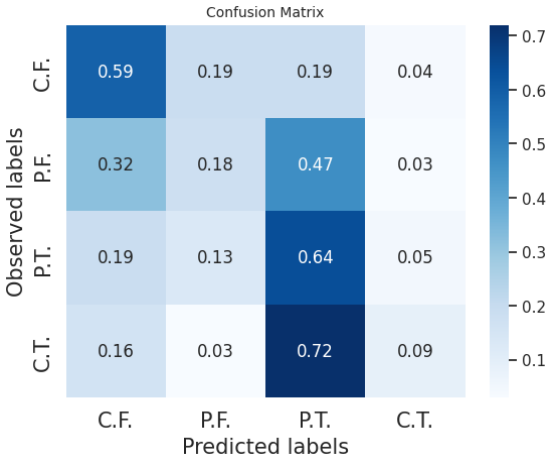

Figure 6.9: Dataset Source Additional

6.3 Fake News Detection and information divergence

Many existing fake news detection approaches rely on binary or multi-class classifiers trained to differentiate between fake and real news, employing deep learning algorithms. They assume that specific features can consistently distinguish fake news from real news, independently of the context. However, classifiers trained on particular datasets often display reduced performance when tested across different domains and timeframes, revealing their limitations.

We believe that an approach based on information divergence and the comparison of claims with ground truth could offer a promising method to model the problem. This approach operates under the assumption that suspicious and potentially fake news contains elements of information not found in a ground truth related to the real-world event it describes. The concept aligns with the notion that fake news often emerges by altering aspects of real-world events to manipulate the narrative surrounding them or inventing entirely fictional events. The idea is to validate each claim by comparing it with the information contained in the ground truth. The higher the information divergence, the greater the likelihood that the claim is potentially fake.

However, successful implementation would require access to a trustworthy ground truth dataset, which may not always be readily available or easily obtainable. Additionally, defining appropriate measures of information divergence and developing effective algorithms for this purpose are important challenges to address. Nonetheless, this approach offers a novel perspective on fake news detection, focusing on the semantic differences between claims and trusted sources.

In the following, we conduct preliminary experiments to establish a baseline for future research in this area. We have developed a framework inspired by the second

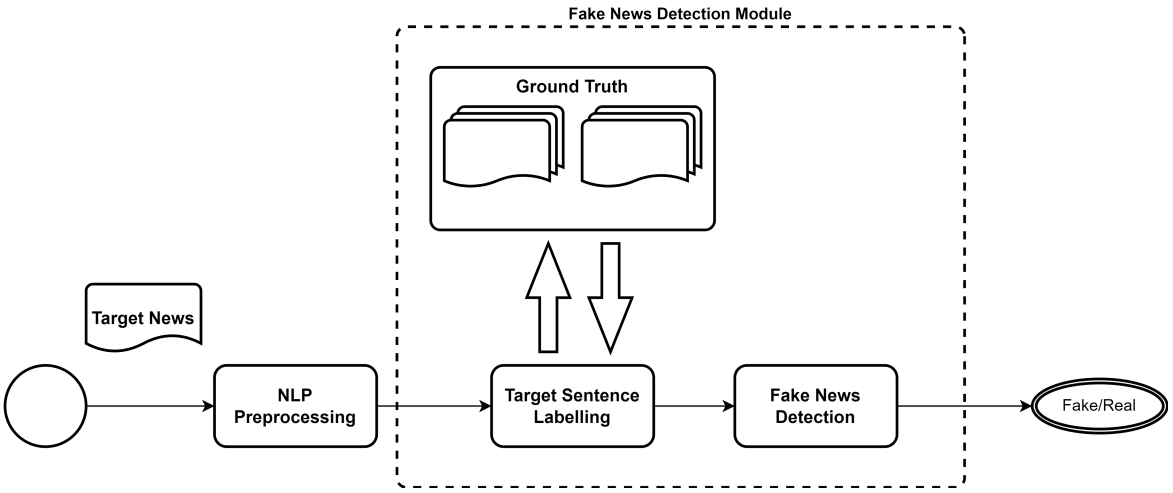

Figure 6.10: Flowchart of the Information Divergence based fake news detection methodology

step of the methodology presented in Chapter 5 for the action-logs, known as the Target Episode Finder (TEF-M). This framework applies the concept of Information Divergence computed at the sentence level, similar to the classification methodology employed in the News Collector system described in Chapter 4. It is important to note that TEF-M identifies the most significant episodes by utilizing a set of patterns known as archetype patterns and making comparisons using similarity metrics. In the context of the action-logs, we employed fuzzy string matching. However, addressing the fake news detection problem requires more precise modelling of natural language events, specifically textual data. In this regard, we utilize the cosine similarity of sentences, in a way similar to what is described in Chapter 4. The flowchart of the approach is shown in Figure 6.10.

The goal is, given a new piece of content, establish if it is fake or real. Firstly, the piece of text is preprocessed. Hashtags were deleted from the text. If it is composed of multiple sentences (i.e. it is a newspaper article), sentence splitting is performed, and sentence-BERT is applied for each sentence to generate reliable representations. Otherwise (i.e. it is a tweet) sentence-BERT is applied directly.

At this point, a Target Sentence Labelling phase is performed. All the sentences that compose the target news are compared with all the sentences of the ground-truth texts, which are the equivalent of archetypical patterns, in AUTOMIA. They are a list of verified news (or tweets propagating news) that are definitely true. All the pairs of sentences (target-archetypic news) are compared against a similarity measure (cosine) and the maximum value is considered. Then, the Fake News Detection is performed. If there is at least one pair whose cosine exceeds a similarity threshold, the information divergence between the two texts is considered not high enough to label the target as fake news.

We tested this strategy by exploiting the dataset collected and labelled using the in-context methodology, described in Section 6.1 of this Chapter. Seven real news and/or tweets were selected as ground truth, among those that were labelled as real by both Prolific users and the gold annotation. The resulting dataset was split into training and test sets (0.4%). The training set was used to select the best threshold, among a selection of values. The threshold that generated better performance in the training phase was used in the testing phase. Table 6.16 reports the results in terms of f1-score on all selected threshold values.

Threshold	F1-Score
0.5	0.66
0.6	**0.70**
0.7	0.55
0.8	0.41

Table 6.16: f1-score for all the thresholds in the training phase

The threshold value selected during the testing phase is 0.6. The results are shown in the Table 6.17.

Metric	Score
f1-score avg	**0.74**
f1-score Fake	0.70
f1-score Real	0.76
Recall Fake	0.81
Recall Real	0.69
Precision Fake	0.62
Precision Real	0.86

Table 6.17: Performances for all the test phase, with 0.6 as threshold, real-based

The results obtained are promising, especially when considering that the similarity measure used, cosine similarity, is a coarse metric and has several limitations. Dealing with sequences of varying lengths can be challenging, and it results in some information loss. For instance, sentences like 'Pietro shot Veronica' and 'Veronica shot Pietro' receive a very high cosine similarity score.

The same experiment was also conducted using verified fake news as ground truth, which is the same dataset used by the Prolific annotators as context. However,

when trying the same approach in reverse, the results were much less promising, as shown in Table 6.18.

Metric	Score
f1-score avg	**0.61**
f1-score Fake	0.70
f1-score Real	0.44
Recall Fake	0.74
Recall Real	0.40
Precision Fake	0.66
Precision Real	0.49

Table 6.18: Performances for all the test phase, with 0.5 as threshold, fake-based

Finally, we report the same experiment, by using the real news as ground truth, exploiting the simulated dataset based on PHEME as described in Section 6.1, in order to apply the methodology in a different context and with a different event. Again, we split the dataset into training and test sets (0.4%) and used the training set to select the best threshold. The training phase is shown in Table 6.19

Threshold	F1-Score
0.5	**0.64**
0.6	0.54
0.7	0.38
0.8	0.30

Table 6.19: f1-score for all the thresholds in the training phase for the PHEME simulated dataset

The selected threshold is 0.5, and the performances on the test set are shown in Table 6.20

We are confident that an approach that delves into the meaning of words and sentences, while harnessing the relationships between the content of real and fake news to distinguish between them, may provide a promising modelling approach for this problem. We can view it as an approach based on information divergence between the news to be debunked and the real and verified story, which could prove to be an intriguing method. The preliminary experiment reported here serves as a baseline and a starting point. In the future, our focus will be on better modeling

Metric	Score
f1-score avg	**0.63**
f1-score Fake	0.33
f1-score Real	0.75
Recall Fake	0.29
Recall Real	0.79
Precision Fake	0.38
Precision Real	0.71

Table 6.20: Performances for the PHEME test phase, with 0.5 as threshold

which types of information to use, going beyond the use of simple cosine as a similarity measure. We aim to provide a divergence-aware metric (i.e., How different is this target news from the reference one?), improve scalability and generalization to new scenarios, and approach the problem on a broader scale, rather than as a binary (or multi-class) classification task.

6.4 Discussion

In this Chapter, a novel methodology for collecting and labeling datasets containing fake and real news has been introduced. The proposed method is specifically tailored for real-world applications that may necessitate a rapid domain adaptation. The collection strategy starts from social media platforms and encompasses the gathering of both social media data and associated newspaper articles. The effectiveness of this approach has been assessed exploiting a dataset that comprises real and fake news pertaining to the Notre Dame Fire of 2019. Furthermore, the methodology has been successfully adapted to a multimodal dataset related to the Ukraine-Russian war.

Our annotation process leverages contextual data, specifically focusing on previously identified instances of fake news related to the topic. The objective is twofold: to accelerate the labelling process and to enhance the quality of the labelled data. This approach underscores the challenges associated with utilizing conventional crowdsourcing settings for tasks that are inherently subjective. Overcoming these limitations entails furnishing all raters with sufficient contextual knowledge pertaining to the subject matter.

Furthermore, our research underscores the ongoing necessity for further investigation of particularly complex fake news, which remain invisible and challenging

to detect for both human annotators and automated detection systems.

We believe that an approach that delves into the semantic nuances of words and sentences, while also leveraging the interrelations between the content of real and fake news to discern between them, is a promising strategy for this issue. This can be conceptualized as an approach based on information divergence between the news articles to be debunked and the authentic and verified narratives, presenting an intriguing avenue to explore. In this Chapter, we have conducted a preliminary experiment that serves as a foundational baseline. Moving forward, we will focus on refining the modelling techniques to determine which types of information are most pertinent, overcoming the utilization of simple cosine similarity measures. We aim to develop a divergence-aware metric to better evaluate the extent of difference between the target news and the reference one. Our aim is also to enhance scalability and adaptability to novel scenarios, and address the problem from a broader perspective rather than framing it solely as a binary or multi-class classification task.

6.5 Summary

In this Chapter, in Section 6.1 we proposed a topic-oriented methodology for the collection and labeling of potentially fake and verified news from social media related to a specific event. This approach is designed to create real-world datasets that encompass both genuine and fake news regarding particular events, facilitating real-world-oriented analysis.

In Section 6.2 we exploited the methodology to create a second dataset that includes both textual and visual components, which is subsequently made available in the context of the EVALITA Challenge known as MULTI-Fake-DetectiVE. The systems proposed by the participants are then discussed, and building upon their results, we employ a recent technique called Prompt Tuning to further advance the work initiated by them.

Finally, in Section 6.3 we adapt the similarity-based method detailed in Chapter 4 and the episode mining approach in Chapter 5 to address the task, presenting an analysis of the results.

Chapter 7

Conclusions and future directions

The goal of the present work was to explore the concept of *episode* derived from Data Mining in a wider sense, with a particular focus on a Natural Language Processing perspective. In order to do this, the concept of *interesting text episode* was introduced in Chapter 2. Starting from the definition of *episode* given in the Frequent Episode Mining field, we defined the *interesting text episode* as a sub-sequence of a longer sequence S that fulfills the conditions specific to the task at hand. For instance, in the context of Fake News Detection, the sequence S represents the collection of all the examined articles or social media posts. Each distinct article or social media post constitutes an episode, composed of a sequence of sentences conveying information. An *interesting text episode* refers to a particular newspaper article or social media post that has been identified as Fake News.

Three main scenarios have been investigated, with two of them falling within the domain of Natural Language Processing, and the third in the domain of Data Mining.

Firstly, in Chapter 4 a system that employs cosine similarity based on Sentence-BERT embeddings to classify all the sentences from target newspaper articles as Similar, Different, and Very Different with respect to sentences in a reference article is presented. This system serves as a case study that implements a method to extract and highlight novel information from one text given another, or, in other words, to detect information divergence between two or more texts. From the episode mining perspective, the text episode we aimed to mine consisted of a piece of information in the form of a set of sentences, which constitutes a sub-sequence within the larger sequence of articles. We evaluated the system, especially the information divergence methodology, by comparing its predictions with human judgments and with the predictions of two BERT-based classifiers in both a model-tuning and prompt-tuning scenario. The results obtained were promising.

Secondly, in Chapter 5 a two-step framework for mining sequences of actions was described as part of the AUTOMIA project. This project encompassed research, de-

sign, development, and experimentation of a Robotic Process Automation (RPA) system aimed at assisting operators in executing tasks that are highly repetitive and of low complexity. The proposed framework is based on two steps: the *Frequent Episode Miner* step, which, given a long sequence of action logs, utilizes Frequent Episode Mining algorithms to extract all sequences of actions occurring with at least a minimum frequency, and the *Target Episode Finder* step, which leverages fuzzy string matching based on the Levenshtein distance to filter out sequences that do not match established patterns. A discussion about the utilization of a similar framework with textual data was conducted, highlighting that in its current form, it cannot generalize from structured sequence data, such as log data, to unstructured data, such as textual content.

Thirdly, in Chapter 6, we delve into the fake news detection problem, a task that has gained significant prominence in recent years, primarily due to the uncontrolled proliferation of fake news on the internet and social media platforms. The very definition of fake news remains a subject of debate, as it encompasses a broad spectrum of variations, including rumors, hoaxes, clickbait, and more. In this context, we described a topic-oriented methodology for collecting and labelling fake and verified news from social media for specific events. This approach aims to create real-world datasets that encompass both real and fake news related to these events, enabling real-world-focused analysis. Additionally, we adapted the methodology to collect a second dataset that includes both textual and visual elements, which is subsequently made available within the context of the EVALITA Challenge known as MULTI-Fake-DetectiVE. We then analyzed the systems proposed by the participants. Building upon the results obtained by the participants in EVALITA, we employed a recent technique called Prompt Tuning to continue the trajectory initiated by them. Finally, we adapted the similarity-based method described in Chapter 4 and the episode mining method in Chapter 5 to address the task. This approach offers a way to explore and combine strategies from both data mining and textual similarity fields.

Regarding the first case study, specifically the similarity-based method for detecting information divergence, the method and the system in which it is implemented represent an initial step toward the ultimate goal of enabling users to discover novel and relevant information. We plan to further enhance this system in future work. In particular, we intend to explore alternative similarity metrics and classification algorithms to better address the challenge of semantic similarity between sentences. We have a strong conviction that this methodology can be applied from various new perspectives. We believe that an approach based on information divergence and the comparison of claims with ground truth could offer a promising method for modeling the problem of Fake News Detection. The framework described in Chapter 6 could be a valuable starting point.

The presented work has tried to demonstrate the usefulness of a multidisciplinary approach in addressing several problems by drawing a common thread that intersects across different domains. The concept of *episode* is explored from both its data mining definition (developing the Frequent Episode Mining framework) and in a traditional Natural Language Processing context (such as the semantic textual similarity task). Subsequently, these two aspects are merged to provide a baseline for tackling a more complex problem from a novel perspective, namely, the fake news detection task.

Appendix A

Publications

Journal papers

1. Passaro L. C., Bondielli A., **Dell'Oglio P.**, Lenci A., Marcelloni F., "In-context annotation of Topic-Oriented Datasets of Fake News: A Case study on the Notre-Dame Fire Event", *Information Sciences, 615*, pages: 657-677., 2022. **Candidate's contributions**: Design, development, and implementation of the In-Context annotation scheme, data annotation, evaluation of the results.

2. Bechini A., Bondielli A., **Dell'Oglio P.**, Marcelloni F., "From basic approaches to novel challenges and applications in Sequential Pattern Mining", *Applied Computing and Intelligence 3.1*, pages:44-78, 2023. **Candidate's contributions**: Review of the literature on the different topics concerned by the survey; evaluation of the results, paper writing.

3. **P. Dell'Oglio**, A. Bondielli, F. Marcelloni, "A System to Support Readers in Automatically Acquiring Complete Summarized Information on an Event from Different Sources.", *Algorithms*, 16.11 (2023): 513. **Candidate's contributions**: design and development of the proposed methodology, design and development of the proposed system, validation, data collection, experiments and evaluation, paper writing.

Peer reviewed conference papers

1. **Dell'Oglio P.**, Bondielli A., Bechini A., Marcelloni F., "Leveraging sequence mining for robot process automation", *Intelligent Systems Design and Applications: 22nd International Conference on Intelligent Systems Design and Applications (ISDA 2022) Held December 12-14*, Volume 4. Cham: Springer Nature Switzerland, 2023. **Candidate's contributions**: Design and development of the proposed framework, experiments, evaluation of the results, paper writing.

2. Bondielli A., **Dell'Oglio P.**, Lenci A., Marcelloni F., Passaro C.L., Sabbatini M., "MULTI-Fake-DetectiVE at EVALITA 2023: Overview of the MULTImodal Fake News Detection and VErification Task", *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, CEUR. org, Parma, Italy. 2023. **Candidate's contributions**: Data retrieval, implementation of the annotation scheme, evaluation of the results.

3. **P. Dell'Oglio**, A. Bondielli, F. Marcelloni, "A system for assisting users in automatically obtaining comprehensive and condensed information about an event from various sources", *Proceedings of Intelligent Systems Design and Applications: 23nd International Conference on Intelligent Systems Design and Applications (ISDA 2023)* **Candidate's contributions**: design and development of the proposed methodology, design and development of the proposed system, validation, data collection, experiments, paper evaluation, and paper writing.

Bibliography

Afroz, S., Brennan, M., and Greenstadt, R. (2012). Detecting hoaxes, frauds, and deception in writing style online. In *2012 IEEE Symposium on Security and Privacy*, pages 461–475. IEEE.

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE.

Aguirre, S. and Rodriguez, A. (2017). Automation of a business process using robotic process automation (rpa): A case study. In *Applied Computer Sciences in Engineering: 4th Workshop on Engineering Applications, WEA 2017, Cartagena, Colombia, September 27-29, 2017, Proceedings 4*, pages 65–71. Springer.

Ahonen, H., Heinonen, O., Klemettinen, M., and Verkamo, A. I. (1997). Applying data mining techniques in text analysis. Technical report, Citeseer.

Aker, A., Derczynski, L., and Bontcheva, K. (2017). Simple open stance classification for rumour analysis. *arXiv preprint arXiv:1708.05286*.

Alam, F., Cresci, S., Chakraborty, T., Silvestri, F., Dimitrov, D., Martino, G. D. S., Shaar, S., Firooz, H., and Nakov, P. (2021). A survey on multimodal disinformation detection. *arXiv preprint arXiv:2103.12541*.

Alam, F., Cresci, S., Chakraborty, T., Silvestri, F., Dimitrov, D., Martino, G. D. S., Shaar, S., Firooz, H., and Nakov, P. (2022). A survey on multimodal disinformation detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6625–6643, Gyeongju, Republic of Korea.

Allcott, H. and Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–236.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Barrón-Cedeno, A., Elsayed, T., Nakov, P., Da San Martino, G., Hasanain, M., Suwaileh, R., Haouari, F., Babulkov, N., Hamdan, B., Nikolov, A., et al. (2020). Overview of checkthat! 2020: Automatic identification and verification of claims in social media. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 215–236. Springer.

Barsacchi, M., Bechini, A., and Marcelloni, F. (2021). Implicitly distributed fuzzy random forests. In *Proc. of the 36th Annual ACM Symposium on Applied Computing*, page 392–399, New York, NY, USA. ACM.

Bechini, A., Bondielli, A., Dell'Oglio, P., and Marcelloni, F. (2023). From basic approaches to novel challenges and applications in sequential pattern mining. *Applied Computing and Intelligence*, 3(1):44–78.

Beedkar, K. and Gemulla, R. (2015). Lash: Large-scale sequence mining with hierarchies. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 491–503.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Bondielli, A. and Marcelloni, F. (2019). A survey on fake news and rumour detection techniques. *Information Sciences*, 497:38–55.

Bozarth, L. and Budak, C. (2020). Toward a better performance evaluation framework for fake news classification. *Proceedings of the International AAAI Conference on Web and Social Media*, 14(1):60–71.

Briscoe, E. J., Appling, D. S., and Hayes, H. (2014). Cues to deception in social media communications. In *2014 47th Hawaii international conference on system sciences*, pages 1435–1443. IEEE.

Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Burgess, C. (1998). From simple associations to the building blocks of language: Modeling meaning in memory with the hal model. *Behavior Research Methods, Instruments, & Computers*, 30(2):188–198.

Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684.

Chang, J. H. and Lee, W. S. (2005). Efficient mining method for retrieving sequential patterns over online data streams. *Journal of Information Science*, 31(5):420–432.

Chang, L., Wang, T., Yang, D., and Luan, H. (2008). Seqstream: Mining closed sequential patterns over stream sliding windows. In *2008 Eighth IEEE International Conference on Data Mining*, pages 83–92. IEEE.

Chen, C.-C., Shuai, H.-H., and Chen, M.-S. (2017). Distributed and scalable sequential pattern mining through stream processing. *Knowledge and Information Systems*, 53(2):365–390.

Chen, C.-C., Tseng, C.-Y., and Chen, M.-S. (2013). Highly scalable sequential pattern mining based on MapReduce model on the cloud. In *2013 IEEE International Congress on Big Data*, pages 310–317. IEEE.

Chen, J. (2009). An updown directed acyclic graph approach for sequential pattern mining. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):913–928.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Chiu, D.-Y., Wu, Y.-H., and Chen, A. L. (2004). An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *Proceedings. 20th International Conference on Data Engineering*, pages 375–386. IEEE.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chua, A. Y. and Banerjee, S. (2016). Linguistic predictors of rumor veracity on the internet. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, page 387. Nanyang Technological University Singapore.

Ciampaglia, G. L., Shiralkar, P., Rocha, L. M., Bollen, J., Menczer, F., and Flammini, A. (2015). Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193.

Cong, S., Han, J., and Padua, D. (2005). Parallel mining of closed sequential patterns. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 562–567.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Czubryt, T. J., Leung, C. K., and Pazdor, A. G. M. (2022). Q-VIPER: Quantitative vertical bitwise algorithm to mine frequent patterns. In Wrembel, R., Gamper, J., Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Big Data Analytics and Knowledge Discovery*, pages 219–233, Cham. Springer International Publishing.

Da San Martino, G., Barrón-Cedeño, A., Wachsmuth, H., Petrov, R., and Nakov, P. (2020). SemEval-2020 task 11: Detection of propaganda techniques in news articles. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, Barcelona (online). ICCL.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

D'Amico, L., Napolitano, D., Vaiani, L., and Cagliero, L. (2023). Polito at multi-fake-detective: Improving fnd-clip for multimodal italian fake news detection. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy. CEUR.org.

Das, A., Yenala, H., Chinnakotla, M., and Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 378–387.

Dell'Oglio, P., Bondielli, A., Bechini, A., and Marcelloni, F. (2022). Leveraging sequence mining for robot process automation. In *International Conference on Intelligent Systems Design and Applications*, pages 224–233. Springer.

Demiriz, A. (2002). webSPADE: a parallel sequence mining algorithm to analyze web log data. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 755–758. IEEE.

Derczynski, L. and Bontcheva, K. (2014). Pheme: Veracity in digital social networks. In *UMAP workshops*.

Derczynski, L., Bontcheva, K., Liakata, M., Procter, R., Hoi, G. W. S., and Zubiaga, A. (2017). Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

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

Diao, S., Wang, P., Lin, Y., and Zhang, T. (2023). Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

DiFonzo, N. and Bordia, P. (2007). Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35.

Dimitrov, D., Bin Ali, B., Shaar, S., Alam, F., Silvestri, F., Firooz, H., Nakov, P., and Da San Martino, G. (2021). SemEval-2021 task 6: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 70–98, Online. ACL.

Ezeife, C. I., Lu, Y., and Liu, Y. (2005). Plwap sequential mining: open source code. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 26–35.

Fei, X., Zheng, S., Yan, L.-j., and Fan, C. (2016). A improved sequential pattern mining algorithm based on PrefixSpan. In *2016 World Automation Congress (WAC)*, pages 1–4. IEEE.

Feng, V. W. and Hirst, G. (2013). Detecting deceptive opinions with profile compatibility. In *Proceedings of the sixth international joint conference on natural language processing*, pages 338–346.

Feremans, L., Cule, B., and Goethals, B. (2018). Mining top-k quantile-based cohesive sequential patterns. In *Proceedings of the 2018 SIAM international conference on data mining*, pages 90–98. SIAM.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Fournier-Viger, P., Lin, J. C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., and Lam, H. T. (2016). The spmf open-source data mining library version 2. In *Machine Learning and Knowledge Discovery in Databases*, pages 36–40, Cham. Springer International Publishing.

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77.

Fournier-Viger, P., Wu, C.-W., Gomariz, A., and Tseng, V. S. (2014). VMSP: Efficient vertical mining of maximal sequential patterns. In *Canadian conference on artificial intelligence*, pages 83–94. Springer.

Fournier-Viger, P., Wu, C.-W., and Tseng, V. S. (2013). Mining maximal sequential patterns without candidate maintenance. In *International Conference on Advanced Data Mining and Applications*, pages 169–180. Springer.

Fournier-Viger, P., Yang, P., Lin, J. C.-W., and Yun, U. (2019). Hue-span: Fast high utility episode mining. In *International Conference on Advanced Data Mining and Applications*, pages 169–184. Springer.

Fournier-Viger, P., Yang, Y., Yang, P., Lin, J. C.-W., and Yun, U. (2020). TKE: Mining top-k frequent episodes. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 832–845. Springer.

Fowkes, J. and Sutton, C. (2016). A subsequence interleaving model for sequential pattern mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 835–844.

Fumarola, F., Lanotte, P. F., Ceci, M., and Malerba, D. (2016). CloFAST: closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems*, 48(2):429–463.

Gan, W., Lin, J. C.-W., Fournier-Viger, P., Chao, H.-C., and Yu, P. S. (2019). A survey of parallel sequential pattern mining. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(3):1–34.

Gan, Z., Li, L., Li, C., Wang, L., Liu, Z., and Gao, J. (2022). Vision-language pre-training:: Basics, recent advances, and future trends. *Foundations and Trends® in Computer Graphics and Vision*, 14(3-4):163–352.

Garofalakis, M., Rastogi, R., and Shim, K. (2002). Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):530–552.

Ge, J. and Xia, Y. (2016). Distributed sequential pattern mining in large scale uncertain databases. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 17–29. Springer.

Ge, J., Xia, Y., Wang, J., Nadungodage, C. H., and Prabhakar, S. (2017). Sequential pattern mining in databases with temporal uncertainty. *Knowledge and Information Systems*, 51(3):821–850.

Giasemidis, G., Singleton, C., Agrafiotis, I., Nurse, J. R., Pilgrim, A., Willis, C., and Greetham, D. V. (2016). Determining the veracity of rumours on twitter. In *Social Informatics: 8th International Conference, SocInfo 2016, Bellevue, WA, USA, November 11-14, 2016, Proceedings, Part I 8*, pages 185–205. Springer.

Gomariz, A., Campos, M., Marin, R., and Goethals, B. (2013). Clasp: An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Guyet, T., Zhang, W., and Bifet, A. (2022). Incremental mining of frequent serial episodes considering multiple occurrences. In *International Conference on Computational Science*, pages 460–472. Springer.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M. (2000). Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 6, pages 355–359.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224.

Hardalov, M., Koychev, I., and Nakov, P. (2016). In search of credible news. In *Artificial Intelligence: Methodology, Systems, and Applications: 17th International Conference, AIMSA 2016, Varna, Bulgaria, September 7-10, 2016, Proceedings 17*, pages 172–180. Springer.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ho, C.-C., Li, H.-F., Kuo, F.-F., and Lee, S.-Y. (2006). Incremental mining of sequential patterns over a stream sliding window. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, pages 677–681. IEEE.

Horne, B. and Adali, S. (2017). This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 759–766.

Hosseininasab, A., van Hoeve, W.-J., and Cire, A. A. (2019). Constraint-based sequential pattern mining with decision diagrams. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1495–1502.

Hromei, C. D., Croce, D., Basile, V., and Basili, R. (2023). Extremita at evalita2023: Multi-task sustainable scaling to large language models at its extreme. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy. CEUR.org.

Huang, J.-W., Lin, S.-C., and Chen, M.-S. (2010). DPSP: Distributed progressive sequential pattern mining on the cloud. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 27–34. Springer.

Huang, K.-Y. and Chang, C.-H. (2008). Efficient mining of frequent episodes from complex sequences. *Information Systems*, 33(1):96–114.

Jelinek, F. (1980). Interpolated estimation of markov source parameters from sparse data. In *Proceeding of the Workshop on Pattern Recognition in Practice*, pages 381–397.

Jin, Z., Cao, J., Zhang, Y., Zhou, J., and Tian, Q. (2016). Novel visual and statistical image features for microblogs news verification. *IEEE transactions on multimedia*, 19(3):598–608.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., and Testuggine, D. (2020). The hateful memes challenge: Detecting hate speech in multimodal memes. In *Advances in Neural Information Processing Systems*, volume 33, pages 2611–2624. Curran Associates, Inc.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Kumar, D. et al. (2023). Mining frequent serial positioning episode rules with forward and backward search technique from event sequences. *The Computer Journal*, 66(7):1622–1643.

Kwon, S., Cha, M., and Jung, K. (2017). Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Lai, M., Menini, S., Polignano, M., Russo, V., Sprugnoli, R., and Venturi, G. (2023). Evalita 2023: Overview of the 8th evaluation campaign of natural language processing and speech tools for italian. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy. CEUR.org.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Le, B., Duong, H., Truong, T., and Fournier-Viger, P. (2017). Fclosm, fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. *Knowledge and Information Systems*, 53(1):71 – 107.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Lenci, A. (2008). Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1):1–31.

Lenci, A. (2018). Distributional models of word meaning. *Annual review of Linguistics*, 4:151–171.

Lenci, A. (2023). Understanding natural language understanding systems. a critical analysis. *arXiv preprint arXiv:2303.04229*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Lewandowsky, S., Ecker, U. K., and Cook, J. (2017). Beyond misinformation: Understanding and coping with the "post-truth" era. *Journal of applied research in memory and cognition*, 6(4):353–369.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Li, Z., Peng, B., He, P., Galley, M., Gao, J., and Yan, X. (2023). Guiding large language models via directional stimulus prompting. *arXiv preprint arXiv:2302.11520*.

Liang, Y.-h. and Wu, S.-y. (2015). Sequence-growth: A scalable and effective frequent itemset mining algorithm for big data based on MapReduce framework. In *2015 IEEE International Congress on Big Data*, pages 393–400. IEEE.

Liao, V. C.-C. and Chen, M.-S. (2014). DFSP: a Depth-First SPelling algorithm for sequential pattern mining of biological sequences. *Knowledge and Information Systems*, 38(3):623–639.

Lin, J. C.-W., Djenouri, Y., Srivastava, G., Li, Y., and Yu, P. S. (2022). Scalable mining of high-utility sequential patterns with three-tier MapReduce model. *ACM Trans. on Knowledge Discovery from Data*, 16(3).

Lin, J. C.-W., Li, T., Pirouz, M., Zhang, J., and Fournier-Viger, P. (2020). High average-utility sequential pattern mining based on uncertain databases. *Knowledge and Information Systems*, 62:1199–1228.

Liu, F., Jiao, Y., Massiah, J., Yilmaz, E., and Havrylov, S. (2021a). Trans-encoder: unsupervised sentence-pair modelling through self-and mutual-distillations. *arXiv preprint arXiv:2109.13059*.

Liu, F., Vulić, I., Korhonen, A., and Collier, N. (2021b). Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders. *arXiv preprint arXiv:2104.08027*.

Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Bras, R. L., Choi, Y., and Hajishirzi, H. (2021c). Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023a). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Z., Yu, X., Fang, Y., and Zhang, X. (2023b). Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428.

Lo, D., Khoo, S.-C., and Li, J. (2008). Mining and ranking generators of sequential patterns. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, page 553 – 564.

Luna, J. M., Padillo, F., Pechenizkiy, M., and Ventura, S. (2018). Apriori versions based on MapReduce for mining frequent patterns on big data. *IEEE Transactions on Cybernetics*, 48(10):2851–2865.

Luo, C. and Chung, S. M. (2005). Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 415–426. SIAM.

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K.-F., and Cha, M. (2016). Detecting rumors from microblogs with recurrent neural networks.

Ma, J., Gao, W., Wei, Z., Lu, Y., and Wong, K.-F. (2015). Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1751–1754.

Magdy, A. and Wanas, N. (2010). Web-based statistical fact checking of textual documents. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 103–110.

Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289.

Marin-Castro, H. M. and Tello-Leal, E. (2021). Event log preprocessing for process mining: A review. *Applied Sciences*, 11(22).

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Miliaraki, I., Berberich, K., Gemulla, R., and Zoupanos, S. (2013). Mind the gap: Large-scale frequent sequence mining. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, pages 797–808.

Miller, G. A. (1971). Empirical methods in the study of semantics. *Semantics, an interdisciplinary reader in philosophy, linguistics, and psychology*, pages 569–585.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Mitra, T. and Gilbert, E. (2015). Credbank: A large-scale social media corpus with associated credibility annotations. In *Proceedings of the international AAAI conference on web and social media*, volume 9, pages 258–267.

Mooney, C. H. and Roddick, J. F. (2013). Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.*, 45(2).

Muzammal, M. and Raman, R. (2015). Mining sequential patterns from probabilistic databases. *Knowl. Inf. Syst.*, 44(2):325–358.

Nakov, P., Da San Martino, G., Alam, F., Shaar, S., Mubarak, H., and Babulkov, N. (2022). Overview of the clef-2022 checkthat! lab task 2 on detecting previously fact-checked claims.

Newman, N., Dutton, W., and Blank, G. (2013). Social media in the changing ecology of news: The fourth and fifth estates in britain. *International journal of internet science*, 7(1).

Nicosia, M. and Moschitti, A. (2017). Learning contextual embeddings for structural semantic similarity using categorical information. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 260–270.

OpenAI (2023). Gpt-4 technical report.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., and Ribeiro, M. T. (2023). Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Passaro, L., Bondielli, A., Lenci, A., Marcelloni, F., et al. (2020). Unipi-nle at checkthat! 2020: approaching fact checking from a sentence similarity perspective through the lens of transformers. In *CEUR WORKSHOP PROCEEDINGS*, volume 2696. CEUR.

Passaro, L. C., Bondielli, A., Dell'Oglio, P., Lenci, A., and Marcelloni, F. (2022). In-context annotation of topic-oriented datasets of fake news: A case study on the notre-dame fire event. *Information Sciences*, 615:657–677.

Passaro, L. C. and Lenci, A. (2016). Evaluating context selection strategies to build emotive vector space models. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2185–2191, Portorož, Slovenia. European Language Resources Association (ELRA).

Patwa, P., Sharma, S., Pykl, S., Guptha, V., Kumari, G., Akhtar, M. S., Ekbal, A., Das, A., and Chakraborty, T. (2021). Fighting an infodemic: Covid-19 fake news dataset. 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 1*, pages 21–29. Springer.

Pei, J., Han, J., and Wang, W. (2002). Mining sequential patterns with constraints in large databases. In *Proc. of the 11th Int'l Conf. on Information and Knowledge Management*, CIKM '02, page 18–25, New York, NY, USA. ACM.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., and Mihalcea, R. (2017). Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Pérez-Rosas, V. and Mihalcea, R. (2015). Experiments in open domain deception detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1120–1125.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

Pham, T.-T. (2015). Efficiently mining sequential generator patterns using prefix trees. *Fundamenta Informaticae*, 138(3):373 – 386.

Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q., and Dayal, U. (2001). Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88.

Pokou, Y. J. M., Fournier-Viger, P., and Moghrabi, C. (2016). Authorship attribution using small sets of frequent part-of-speech skip-grams. In *The Twenty-Ninth International Flairs Conference*.

Poongodi, K. and Kumar, D. (2022). Natural exponent inertia weight-based particle swarm optimization for mining serial episode rules from event sequences. *IETE Journal of Research*, pages 1–15.

Potthast, M., Kiesel, J., Reinartz, K., Bevendorff, J., and Stein, B. (2017). A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Potthast, M., Köpsel, S., Stein, B., and Hagen, M. (2016). Clickbait detection. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*, pages 810–817. Springer.

Puccetti, G. and Esuli, A. (2023). Aimh at multi-fake-detective: System report. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy. CEUR.org.

Qazi, M., Khan, M. U., and Ali, M. (2020). Detection of fake news using transformer model. In *2020 3rd international conference on computing, mathematics and engineering technologies (iCoMET)*, pages 1–6. IEEE.

Qazvinian, V., Rosengren, E., Radev, D., and Mei, Q. (2011). Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1589–1599.

Qiao, S., Tang, C., Dai, S., Zhu, M., Peng, J., Li, H., and Ku, Y. (2008). Partspan: Parallel sequence mining of trajectory patterns. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 363–367. IEEE.

Qin, Y., Wurzer, D., Lavrenko, V., and Tang, C. (2016). Spotting rumors via novelty detection. *arXiv preprint arXiv:1611.06322*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raissi, C., Poncelet, P., and Teisseire, M. (2006). SPEED: mining maximal sequential patterns over data streams. In *2006 3rd International IEEE Conference Intelligent Systems*, pages 546–552. IEEE.

Rathore, S., Dawar, S., Goyal, V., and Patel, D. (2016). Top-k high utility episode mining from a complex event sequence. In *Proceedings of the 21st international conference on management of data, computer society of India*.

Ravikumar, P., Likhitha, P., Venus Vikranth Raj, B., Uday Kiran, R., Watanobe, Y., and Zettsu, K. (2021). Efficient discovery of periodic-frequent patterns in columnar temporal databases. *Electronics*, 10(12):1478.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Reimers, N. and Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ritika and Gupta, S. K. (2022). Mining transactional databases for frequent and high-utility fuzzy sequential patterns with time intervals. *IEEE Access*, 10:71107–71119.

Roy, K. K., Moon, M. H. H., Rahman, M. M., Ahmed, C. F., and Leung, C. K. (2021). Mining sequential patterns in uncertain databases using hierarchical index structure. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 29–41. Springer.

Rubin, V. L., Chen, Y., and Conroy, N. K. (2015). Deception detection for news: three types of fakes. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Rubin, V. L., Conroy, N., Chen, Y., and Cornwell, S. (2016). Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the second workshop on computational approaches to deception detection*, pages 7–17.

Ruchansky, N., Seo, S., and Liu, Y. (2017). Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.

Sahli, M., Mansour, E., and Kalnis, P. (2013). Parallel motif extraction from very long sequences. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 549–558.

Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sarkar, S., Tudu, N., and Das, D. (2023). Hijli-ju-clef at multi-fake-detective: Multimodal fake news detection using deep learning approach. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy. CEUR.org.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. (2022). Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Segatori, A., Bechini, A., Ducange, P., and Marcelloni, F. (2018). A distributed fuzzy associative classifier for big data. *IEEE Transactions on Cybernetics*, 48(9):2656–2669.

Shaar, S., Georgiev, N., Alam, F., Martino, G. D. S., Mohamed, A., and Nakov, P. (2021). Assisting the human fact-checkers: Detecting all previously fact-checked claims in a document. *arXiv preprint arXiv:2109.07410*.

Shaoul, C. and Westbury, C. (2010). Exploring lexical co-occurrence space using hidex. *Behavior Research Methods*, 42(2):393–413.

Shi, B. and Weninger, T. (2016). Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102.

Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

Silverman, C. (2015). Lies, damn lies and viral content.

Slovikovskaya, V. (2019). Transfer learning from transformers to fake news challenge stance detection (fnc-1) task. *arXiv preprint arXiv:1910.14353*.

Song, S., Hu, H., and Jin, S. (2005). HVSM: a new sequential pattern mining algorithm using bitmap representation. In *International conference on advanced data mining and applications*, pages 455–463. Springer.

Songram, P., Boonjing, V., and Intakosum, S. (2006). Closed multidimensional sequential pattern mining. In *Third International Conference on Information Technology: New Generations (ITNG'06)*, pages 512–517.

Sowmya, H. K., Uma Reddy, N. V., Kavyashree, C., and Anandhi, R. J. (2022). Discovery of frequent pagesets from weblog using Hadoop Mapreduce based parallel apriori algorithm. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 765–770.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96)*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Truong, T., Duong, H., Le, B., Fournier-Viger, P., Yun, U., and Fujita, H. (2021). Efficient algorithms for mining frequent high utility sequences with constraints. *Information Sciences*, 568:239–264.

Tsai, C.-F., Lin, W.-C., and Ke, S.-W. (2016). Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies. *Journal of Systems and Software*, 122:83–92.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Van Der Aalst, W. (2012). Process mining. *Commun. ACM*, 55(8):76–83.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vieweg, S. (2010). Microblogged contributions to the emergency arena: Discovery, interpretation and implications. *Computer Supported Collaborative Work*, pages 515–516.

Vosoughi, S. (2015). *Automatic detection and verification of rumors on Twitter*. PhD thesis, Massachusetts Institute of Technology.

Vosoughi, S., Mohsenvand, M., and Roy, D. (2017). Rumor gauge: Predicting the veracity of rumors on twitter. *ACM transactions on knowledge discovery from data (TKDD)*, 11(4):1–36.

Wang, J. and Han, J. (2004). BIDE: efficient mining of frequent closed sequences. In *Proc. of the 20th International Conference on Data Engineering*, pages 79–90. IEEE.

Wang, S., Zhou, W., and Jiang, C. (2020). A survey of word embeddings based on deep learning. *Computing*, 102:717–740.

Wang, W. Y. (2017). " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Whitehouse, C., Weyde, T., Madhyastha, P., and Komninos, N. (2022). Evaluation of fake news detection with knowledge-enhanced language models. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 16, pages 1425–1429.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Wu, C.-H., Lai, C.-C., and Lo, Y.-C. (2012). An empirical study on mining sequential patterns in a grid computing environment. *Expert Systems with Applications*, 39(5):5748–5757.

Wu, C.-W., Lin, Y.-F., Yu, P. S., and Tseng, V. S. (2013). Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 536–544.

Wu, K., Yang, S., and Zhu, K. Q. (2015). False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*, pages 651–662. IEEE.

Wu, Y., Agarwal, P. K., Li, C., Yang, J., and Yu, C. (2014). Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600.

Wu, Y., Tong, Y., Zhu, X., and Wu, X. (2017). NOSEP: Nonoverlapping sequence pattern mining with gap constraints. *IEEE Transactions on Cybernetics*, 48(10):2809–2822.

Wu, Y., Zhu, C., Li, Y., Guo, L., and Wu, X. (2020). NetNCSP: Nonoverlapping closed sequential pattern mining. *Knowledge-based systems*, 196:105812.

Yan, X., Han, J., and Afshar, R. (2003). CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 166–177. SIAM.

Yang, F., Liu, Y., Yu, X., and Yang, M. (2012). Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pages 1–7.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yang, Z. and Kitsuregawa, M. (2005). LAPIN-SPAM: An improved algorithm for mining sequential pattern. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222. IEEE.

Yang, Z., Wang, Y., and Kitsuregawa, M. (2007). LAPIN: Effective sequential pattern mining algorithms by last position induction for dense databases. In Kotagiri, R., Krishna, P. R., Mohania, M., and Nantajeewarawat, E., editors, *Advances in Databases: Concepts, Systems and Applications - DASFAA 2007*, volume 4443 of *Lecture Notes in Computer Sciences*, pages 1020–1023, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yin, J., Zheng, Z., and Cao, L. (2012). Uspan: An efficient algorithm for mining high utility sequential patterns. *KDD 2012*.

You, T., Sun, Y., Zhang, Y., Chen, J., Zhang, P., and Yang, M. (2022). Accelerated frequent closed sequential pattern mining for uncertain data. *Expert Systems with Applications*, 204:117254.

Yu, X., Liu, J., Liu, X., Ma, C., and Li, B. (2015). A mapreduce reinforced distributed sequential pattern mining algorithm. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 183–197. Springer.

Yun, U. and Leggett, J. J. (2006). WSpan: Weighted sequential pattern mining in large sequence databases. In *2006 3Rd international IEEE conference intelligent systems*, pages 512–517. IEEE.

Zabihi, F., Ramezan, M., Pedram, M. M., and Memariani, A. (2010). Fuzzy sequential pattern mining with sliding window constraint. In *2010 2nd International Conference on Education Technology and Computer*, volume 5, pages V5–396–V5–400.

Zaki, M. J. (2001a). Parallel sequence mining on shared-memory machines. *Journal of Parallel and Distributed Computing*, 61(3):401–426.

Zaki, M. J. (2001b). SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1):31–60.

Zhang, H., Fan, Z., Zheng, J., and Liu, Q. (2012). An improving deception detection method in computer-mediated communication. *Journal of Networks*, 7(11):1811.

Zhang, J., Wang, Y., and Yang, D. (2015). CCSpan: Mining closed contiguous sequential patterns. *Knowledge-Based Systems*, 89:1–13.

Zhang, Z., Zhang, A., Li, M., and Smola, A. (2022). Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Zhang, Z., Zhang, A., Li, M., Zhao, H., Karypis, G., and Smola, A. (2023). Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

Zhao, Z., Resnick, P., and Mei, Q. (2015). Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th international conference on world wide web*, pages 1395–1405.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2022). Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Zhu, H., Wang, P., He, X., Li, Y., Wang, W., and Shi, B. (2010). Efficient episode mining with minimal and non-overlapping occurrences. In *2010 IEEE International Conference on Data Mining*, pages 1211–1216. IEEE.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., and Procter, R. (2018a). Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36.

Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., Lukasik, M., Bontcheva, K., Cohn, T., and Augenstein, I. (2018b). Discourse-aware rumour stance classification in

social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290.

Zubiaga, A., Liakata, M., and Procter, R. (2016a). Learning reporting dynamics during breaking news for rumour detection in social media. *arXiv preprint arXiv:1610.07363*.

Zubiaga, A., Liakata, M., and Procter, R. (2017). Exploiting context for rumour detection in social media. In *Social Informatics: 9th International Conference, SocInfo 2017, Oxford, UK, September 13-15, 2017, Proceedings, Part I 9*, pages 109–123. Springer.

Zubiaga, A., Liakata, M., Procter, R., Wong Sak Hoi, G., and Tolmie, P. (2016b). Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.