Odd Gunnar Aspaas & Oscar Carl Vik

Leveraging Graph Attention Networks and Knowledge Graphs for Fake News Detection

Casting Fact-Checking as a Link Prediction task

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2021

Odd Gunnar Aspaas & Oscar Carl Vik

Leveraging Graph Attention Networks and Knowledge Graphs for Fake News Detection

Casting Fact-Checking as a Link Prediction task

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

A severe downside of online media has been the increase in the propagation of fake news. Every day, technology is exposing millions of people to advertisements, articles, and social media posts, not subject to third-party filtering, fact-checking, or editorial judgment. Though there has been a rapid increase in the amount of research towards effective computational detection techniques, the majority of approaches cannot explain what is fake in the target news content. As the credibility of media and third-party checking organizations is at an all-time low, we argue the interpretability of any new method should be imperative when choosing detection techniques for fake news. One approach often argued to accommodate such requirement, which will be the focus of this thesis, are those based on computational-oriented fact-checking. In essence, this thesis present the study, design and implementation of a novel link prediction model and a framework to automatically extract and verify claims in order to classify news documents. The aim of our thesis has three parts: (i) explore the potential for attention-based Graph Neural Networks to reach state-of-the-art link prediction performance in knowledge graphs; (ii) investigate this model's ability to achieve state-of-the-art fake news detection performance; and (iii) evaluate the effectiveness of claim extraction in the context of fact-checking. The proposed link prediction model, named mrGraphStar, has shown ability to reach performance in close proximity to that of state-of-the-art models on two benchmark datasets. We argue, by investigating some potential venues of improvements, graph attention networks may be able to overtake the currently best performing link prediction models. While the efforts of adopting this model in a fake news detection framework did not yield the same success, we investigate multiple reasons explaining why the approach could still bear fruitful results.

Sammendrag

Sosiale medier har hatt stor innvirkning i hvordan informasjon deles og konsumeres. En klar negativ medfølge har vært den store økningen i spredning av falske nyheter. En falsk nyhet som spres i USA kan være i Norge kun sekunder senere. Hver dag blir mennesker verden over eksponert for målrettet annonsering, artikler og poster fra sosiale medier. Disse spres uten noen krav til tredjeparts filtrering eller faktasjekking. Samtidig har det blitt gjort store fremskritt i teknologi for automatisk deteksjon av falske nyheter. På tross av dette har det vært manglende fokus på teknologi hvor brukere kan forstå begrunnelsen bak dens konklusjoner. Ettersom kredibiliteten til media og faktasjekkere er lavere enn noen sinne, argumenterer vi for at forståelige prediksjoner er avgjørende ved valg av teknologi for å detektere falske nyheter. En type teknologi som støtter denne egenskapen, og som vil være fokus i denne oppgaven, er teknologi basert på computational-oriented fact-checking. Denne oppgaven presenterer studiet, design og implementeringen av en ny tilnærming til link prediction problemet gjennom en tilpasning av en eksisterende arkitektur, samt konstruksjon av et komplett system for å trekke ut og validere påstander fra nyhetsartikler. Målet med oppgaven er tredelt; (i) undersøke om Graph Attention Networks kan tilpasses for å gjennomføre og nå state-of-the-art link prediction ytelse i knowledge graphs; (ii) vil denne modellen kunne bli brukt til å nå samme ytelse som state-of-the-art teknologi for fake news detection; og (iii) evaluere effektiviteten til et system som trekker ut påstander fra tekst i forbindelse med deteksjon av falske nyheter. Vår foreslåtte attention-based GNN modell, ved navn mrGraphStar, har gjennom denne oppgaven vist potensiale til å nå ytelse nært opp mot dagens beste link prediction modeller. Vi utreder også hvorfor, med noen forbedringer, en slik modell vil kunne overgå selv de beste nåværende løsningene. Våre forsøk på å utnytte mrGraphStar til deteksjon av falske nyheter viser seg å ikke kunne konkurrere med ytelsen til dagens state-of-the-art modeller. På tross av dette undersøker vi flere mulige forklaringer, og setter lys på faktorer som kan gjøre at et slikt rammeverk kan fungere godt.

Preface

This thesis is submitted to the *Norwegian University of Science and Technology* (NTNU) as a part of the course *TDT4900 - Computer Science, Master Thesis*. The progress and direction of our research has been supervised by Özlem Özgöbek, Associate Professor at NTNU's *Department of Computer and Information Science* (IDI). The research builds upon, and proceeds the efforts and findings of our previous work [1]. We would like to thank our supervisor, Özlem Özgöbek, for guiding the research as well as providing valuable feedback throughout the process.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter will provide an introduction to the problem domain and context of the thesis. We start by highlighting some general information about the motivation and importance of our topic. Then, in a concise matter, we present the problem outline and the research goals and questions which form the basis of our research. Further, we present the contributions and thesis outline.

1.1 Background and motivation

On the last day of 2019, a novel virus was identified in Wuhan, China. This virus was later named SARS-CoV-2 and has, as of 10th of June 2021, claimed over 3.7 million lives and infected over 174 million people[1]. Surrounding the virus, a veil of myths and misinformation emerged. Among them were both false claims about the vaccines, as well as ideas of unproven substitutes [2]. As a result, when hydroxychloroquine was touted as a possible treatment, even though it has no documented effect on the virus, Lupus patients suffered a shortage of essential medicine [3][2]. During the Munich Security Conference in 2020, the World Health Organization Director-General stated:

> "*We are not just fighting an epidemic; we are fighting an infodemic*"
>
> — T. A. Ghebreyesus[3]

While online media has allowed for fast dissemination and easy access at a low cost, it has also been the source of wider propagation of fake news. Every day, millions of people are exposed to advertisements, articles, and social media posts, not subject to third-party filtering, fact-checking, or editorial judgment. Currently,

[1]`https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6`

[2]`https://www.sciencedaily.com/releases/2020/11/201106103103.htm`

[3]`https://www.who.int/director-general/speeches/detail/munich-security-conference`

more than half of U.S. adults read news from social media [4]. Additionally, research has found that for all categories of information, falsehood diffuses significantly faster and farther than the truth, and in many cases, by an order of magnitude [5]. Hesitancy to vaccines, one of the top ten threats to global health[4], can be directly linked with the spreading of fake news [6]. This is a growing trend where we could see a majority of people having anti-vaccination views in the next decade [7]. The anti-vaccination movement serves as just one example of the rising concern described in [8]; that online news makes it easier for like-minded citizens to form "echo chambers" insulating them from contrary perspectives. In general, we are at a point in time where media credibility is at an all-time low [9], and the need for automated and unbiased fake news detection tools is at an all-time high.

Meanwhile, every year an increasing amount of data about human and natural activities are being collected. The number of devices connected to IP networks is estimated to surpass three times the global population by 2023 [10], and with this growth follows an exponential increase of our digital footprint. Simply collecting this information is not very useful, but enormous opportunities lie in the ability to extrapolate knowledge from it. The past decades have bred the creation of many approaches to extract and store data. Semantic networks, also known as graphs, such as social graphs and knowledge graphs, have been widely employed due to their inherent capability to model structured data in a machine-interpretable way [11]. The sheer amount of knowledge contained within these networks is colossal. As an example, Google KG[5] contains an estimate of 18 billion facts about 570 million named entities. To extrapolate this knowledge, a considerable amount of attention has been devoted to the computational analysis of networks, especially in areas such as recommendation systems [12], combinatorial optimization [13], and fake news detection [14][15]. However, albeit the size of current state-of-the-art semantic networks, they all suffer from being incomplete. Consequently, efforts have been made to utilize the underlying properties of networks to infer missing facts, or links between entities; a task referred to as link prediction.

1.2 Problem Outline

While the issue of misinformation is more significant than ever, there has been a rapid increase in the amount of research towards effective detection techniques [16]. In particular, computational detection of fake news has been studied over the past few years, yielding promising results. Although these approaches can be effective, we argue a critical missing piece of these studies to be the interpretability of such detection. In the context of machine learning systems, interpretability refers to the degree to which a human can understand the cause of a decision made

[4]https://www.who.int/news-room/spotlight/ten-threats-to-global-health-in-2019
[5]https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

by an algorithm [17]. Unfortunately, the majority of approaches cannot explain what is fake in the target news content. As the credibility of media and third-party checking organizations is at an all-time low [9], the interpretability of any new method should be paramount when choosing detection techniques for fake news.

One approach often argued to accommodate such requirement are computational-oriented fact-checking approaches. An essential aspect of these approaches is the utilization of knowledge bases to conduct predictions. Each statement made in a document must be checked against this knowledge base to verify its veracity. Therein lies a promising opportunity; to extrapolate factual statements by utilizing the enormous amount of knowledge from semantic networks. Automatic systems built on semantic networks could help mitigate the spread of news having no significant third-party filtering, fact-checking, or editorial judgment. Ideally, fact-checking of statements could be done by conducting look-ups among the large amount of publicly available semantic networks. However, the issue of incompleteness remains, and the need for models that can infer missing facts arises.

The main issue with link prediction remains the quality of the inferred facts. A severe weakness in early deep learning approaches stemmed from the finding that Convolutional Neural Networks struggle with prediction tasks in large complex graphs [18]. This was a result of the difficulties of defining convolutional and pooling layers for non-Euclidean data. This, in turn, inspired the rise of Graph Neural Networks (GNN), which is currently a prominent research area in the artificial intelligence community. Among the most cutting-edge architectures are those based on the attention mechanism. These architectures omit several issues with previous GNN architectures by enabling (implicitly) specifying different weights to different nodes in a neighborhood without requiring any costly matrix operation. Graph attention models have achieved or matched state-of-the-art results across various tasks on multiple graph benchmarks. However, they have not yet been utilized for link prediction tasks in heterogeneous graphs such as knowledge graphs to the best of our knowledge.

This thesis will present the creation and implementation of a novel framework to automatically classify fake news. This is done through the combination of an attention-based GNN model for link prediction in knowledge graphs and an information extraction model, which takes a text as input and returns the claims made in the form of (subject, predicate, object) triple. This system is to be used to classify news articles based on the truthfulness of these extracted claims. The research is based on empirical testing of our hypotheses through objective data collection and interpretation, with the aim of producing observable and quantifiable results. We will base the performance of our implementation on multiple seminal and state-of-the-art approaches to link prediction and fake news detection.

1.3 Research Goals and Questions

The goal of this thesis is to contribute to the research on link prediction by developing a novel model for link prediction in knowledge graphs and evaluating its potential application in fake news detection for computational-oriented fact-checking. In connection to these goals we have identified the following three research questions:

RQ1	Can Graph Attention Networks reach state-of-the-art link prediction performance on multi-relational datasets?
RQ2	Can the link prediction model from RQ1, trained on an external knowledge graph, be used to accurately detect fake news documents?
RQ3	Can the link prediction model from RQ2 be extended by integrating a claim extraction system for automatic fake news detection?

1.4 Research Contributions

This research will contribute to the design, creation, and evaluation of a framework for fact-checking the truthfulness of claims in news content. This includes a system for extracting RDF-triples, as well as a novel link prediction model for for predicting the validity of triples. To the best of our knowledge, Graph Attention Networks have not been used for link prediction in knowledge graphs. Thus, this research will contribute to the adaption and evaluation of such architecture in context of fake news detection. To summarize, this thesis contributes the following:

Model	The design, creation, and evaluation of an attention-based GNN model for link prediction in knowledge graphs.
Framework	The design, creation, and evaluation of a novel framework to automatically detect fake news by combining a model that extracts triples from raw text and the said GNN model.
Dataset	A manually annotated dataset for fake news detection and triple classifcation.
Domain Overlap	The evaluation of the importance of domain overlap when using a LP model trained on an external KG to classify news.
Claim Extraction	The evaluation of current state-of-the-art triple extraction in the context of fake news detection.

1.5 Thesis Outline

Chapter 1 - Introduction	The current chapter gives an introduction to the problem domain and context of the thesis. This includes its motivation, approach, goals, and a short summary of the contributions.
Chapter 2 - Background	An overview of the theoretical basis of our research is presented in this chapter. This includes relevant theories pertinent to our work on both fake news detection and link prediction.
Chapter 3 - Related Work	This chapter presents a brief case study examining related work akin to the research of this thesis. This includes the research used for comparisons and evaluation of our contributions, as well as the research forming the basis of our approach.
Chapter 4 - Method	In this chapter the models and frameworks used to conduct the experiments of the thesis are described.
Chapter 5 - Experiments	This chapter presents the activities undertaken, including descriptions of tools, datasets, parameter selection, and evaluation metrics.
Chapter 6 - Results	In this chapter we present the results of the various experiments related to the goals and contributions of the thesis.
Chapter 7 - Discussion	This chapter discusses the results, strengths and weaknesses, as well as exploring the potential areas of improvement.
Chapter 8 - Conclusion	The final chapter concludes the motivation, choices and contributions of this thesis's. It provides short but concise answers, complementing the findings from our experiments. Additionally the potential venues still unexplored are presented.

Chapter 2

Theoretical Background

The following chapter presents some general theories pertinent to our research inquiry. First, relevant facets related to our work on fake news detection are introduced, transitioning over to the latter parts, which focus on topics related to our link prediction model. As the work of this thesis proceeds the efforts and findings of our previous work [1], some sections will bear close resemblance.

2.1 Fake News

Definition 1 (Fake News)**.** *Information containing false, inaccurate, or misleading statements, which can be proven to be false, regardless of the author's intent.*

Fake news has, over recent years, grown into an all-capturing term of generally false information. From media, the term is commonly used to describe a story which is seen as damaging to an agency, entity, or person. Fake news has become the de-facto expression for misleading information, especially web-related information, which is presented as news [19]. However, in computer science, and more specifically this thesis, fake news needs to be distilled further. Definition 1 of fake news represents a pragmatic way to determine whether or not a text is categorized as fake news based on the content of a text.

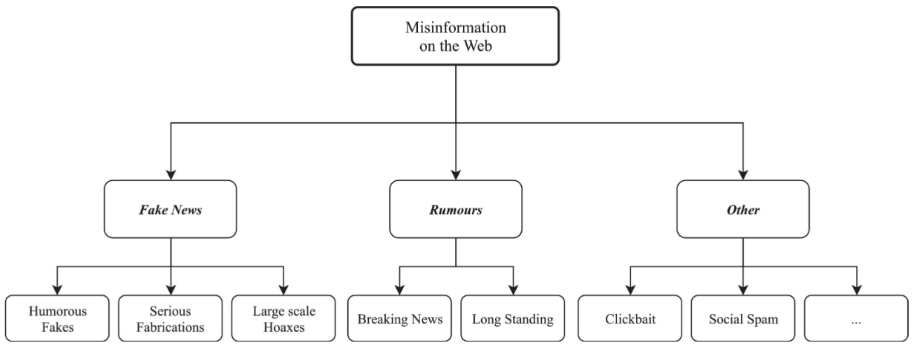


Figure 2.1: Hierarchy of types of misinformation [16]

Note from Figure 2.1 that fake news is a subcategory of *misinformation*, which implies the author did not intend to mislead but simply was ill-informed on the topic of information [20]. This is in contrast to *disinformation*, where the author intentionally forged the information to mislead. While *rumors* are defined as a subcategory of misinformation, it differs slightly because misinformation has previously been disproved, while rumors can turn out to be either true or false. Similar to fake news, the spread of false rumors can cause severe damage even in a short period.

Further dissection of the term fake news done by [21], separates fake news into three groups; *serious fabrications*, *large-scale hoaxes*, and *humorous fakes*. Serious fabrications are a quintessential type of fake news, where dishonest information written with antagonistic fashion are being spread, often through social media. Large-scale hoaxes are usually presented as legitimate news and are relatively complex and large-scale fabrications, which are meant to cause material loss or harm to the victim [22, p. 875]. Finally, humorous fakes which encapsulates satires, parodies, etc. portrayed as news. These stories can be read without being deceitful if the reader is aware of the humorous intent.

When looping back to the pragmatic definition of fake news, one can see that it is in close proximity to misinformation, which is why we henceforth will interchange between misinformation and fake news. The definition also does not consider whether the text is a parody or not. These terms design the task of fake news detection to not doing the detection of disinformation, but the detection of misinformation.

2.2 Fake News Detection

Most approaches to the detection of false information characterize the task as a classification problem, where the aim is to associate a spectrum of labels such as clickbait, satire, true or false with a particular document [16]. Although other perspectives will be discussed, this is the chosen perspective of this thesis as well. Disregarding the detection technique, results are generally better if the system is designed to detect only a select few labels such as $\{true, false\}$ for fake news. Restricting the domain, or topic, of the documents will also greatly impact the accuracy of the systems. General systems usually underperform compared to specialized systems that will have an increased insight within its domain. The following subsections will present both the process of deciding what information to base the analysis on, and the techniques available to one's disposal.

2.2.1 Feature extraction

Different detection techniques consider different kinds of information, or features, relevant for the analysis. At the fist level of abstraction we differentiate between *content-based features*; which are features directly extracted from a document, and *context-based features*; features from auxiliary information such as user's characteristics and network propagation features. Many approaches in the literature focus on using content features for classification. On the other hand, the number of approaches for fake news detection relying purely on context features are rare [23]. The most common approach is to use a mixture of both content and context features in the detection.

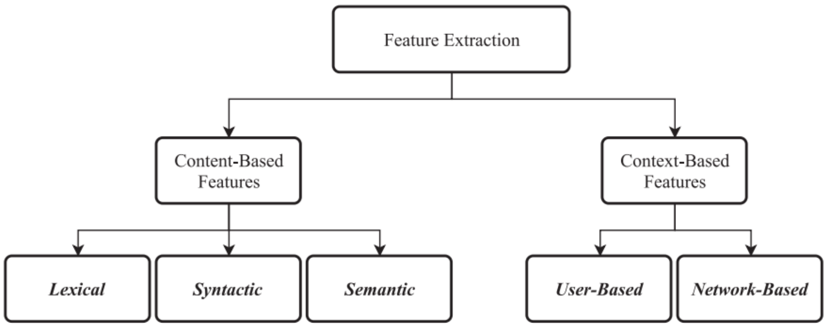


Figure 2.2: Types of features in fake news detection [16]

Content-Based Features

Content-based features are directly extracted from the news content. These features include any kind of content available in the document like text, images, video, or sound. Among non-textual content features, image analysis is one of the most commonly used. By examining the attached images, [24] reached an accuracy of 72.7% in assessing the credibility of online news. Most of the work on content-based features has been concerning the identification of linguistic cues in texts, as is also the approach to manual fake news detection used by professional journalists. By utilizing Natural Language Processing (NLP) tools, we can easily extract textual features such as tokens, phrase structure; parts-of-speech tags and phrasal categories, dependency structure, etc. to obtain a structured representation of a document to be used in the analysis. As shown in Figure 2.2, the textual content-based features can be identified at word-level (*lexical*), in the way sentences are structured (*syntactic*), and by analyzing the attributes of the underlying meaning conveyed (*semantic*) [25]. We now discuss the expressive elements within each of these levels and reference some existing methods that utilize them.

Lexical features concerns the actual words, n-grams, and phrases used in the document. [26] successfully determines the veracity of online information by analyzing the use of negations, abbreviations, word complexity, and word vulgarity.

Syntactic features concern aspects such as word counts, sentence length, parts-of-speech patterns, etc. The complexity of sentences can be utilized to indicate the reliability of information [26][27] and has been used as a feature in fake news detection [28].

Semantic features concern the underlying meaning that is being conveyed in a piece of text. Semantic features are identified by analyzing the larger meaning of the text (phrase, sentence, or paragraph) rather than only analyzing at word-level [25]. Distributional semantics techniques such as word embeddings are often used to extract these features. Several machine learning and deep learning approaches for fake news detection successfully utilize semantic feature analysis through word embeddings [29][30].

Context-Based Features

Context-based features concern auxiliary information such as user behavior, user network analysis, news sources, propagation structures, and news diffusion structure. As shown in Figure 2.2, we can define these features as either *user-based features* or *network-based features*. Among the benefits of contextual-based features is the fact that they can be applied to analysis regardless of the document language. This is especially beneficial when dealing with languages missing larger, cleaner, more readily available text resources, producing poor quality NLP systems that make textual features inconvenient.

User-based features concern characteristics of the social media user who produces or shares the news content. The features analyzed include; the number of posts, age of the account, amount of activity, number of connections or followers, and social circles. A common issue considering user-based features is availability [16]. This is mainly due to privacy constraints on different platforms, making information on the users and user interactions inaccessible. Further, consistency in the presence and representation of specific information is unusual across different online media platforms.

Network-based features concern modeling properties of the network where the news is shared. A majority of existing studies analyzing network-based features are limited to the use of statistics on diffusion patterns, such as the number of retweets and propagation times [31].

2.2.2 Detection Techniques

Fake news detection approaches can coarsely be grouped into classification and non-classification approaches. Our primary focus will be on classification, which can be further divided into approaches based on *Machine Learning* (ML) and *Deep Learning* (DL). We also give a brief introduction to some select non-classification approaches. Lastly, we present a special kind of approach, namely *computational-oriented fact-checking*. These approaches may utilize either ML, DL, or other techniques in the detection of fake news.

Machine Learning

Machine learning algorithms have proven to be extremely useful for solving complex tasks provided real-world data. Especially supervised machine learning techniques have been highly adopted in research concerning the detection of fake news.

Among the historically most used methods for classification are *Support Vector Machines* (SVMs) [32]. SVMs are discriminative classifiers formally defined by a separating hyperplane. Given labeled training data, the algorithm outputs an optimal hyperplane to classify new data examples. This hyperplane is calculated by finding the divider that minimizes the noise sensitivity by maximizing the margin of the training data. SVMs have been successfully applied to the task of fake news detection and have been shown to outperform many other supervised ML approaches [33].

Another proposed algorithm is the *Decision Tree* [34]. It adopts a divide-and-conquer approach to classification. Decision trees are generated from data with algorithms such as *C4.5* [35]. In general, each internal node in a decision tree represents an attribute, and the edges are marked by the condition on its parent node. Each data attribute value is compared to these conditions as it flows down the decision tree. The most significant attribute is detected and designated as the root node of the tree. Further, the tree is split into multiple subsets with new decision nodes. The leaf nodes are called terminal nodes and occur when all instances at this point have the same class. The adaption of decision tree algorithms has shown particularly useful in the rumor analysis task [16].

Random Forests [36] are ensembles of decision tree predictors created from randomly selected subsets of the training data. To decide the final class of the input data, predictions from the different decision trees are aggregated. The aggregation of many decision trees helps reduce the effect of noise, yielding more accurate classification. An interesting quality of regression trees is that they do not overfit, following the *Law of Large Numbers*. This algorithm has shown promising results and has been employed in a number of works on fake news detection and rumor analysis [27][37].

Hidden Markov Models (HMMs) [38] are tools for representing probability distributions over sequences of observations. The sequence of observable variables X is generated by a sequence of internal hidden states Z, which can not be directly observed. The transitions between the hidden states are assumed to have the form of a Markov chain. Three parameters can fully determine an HMM; a start probability vector Π, a transition probability matrix $\mathbf{A}$, and the emission probability of the observable variable θ_i, conditioned on hidden state i. HMMs are widely used for language modeling tasks, including fake news detection [39].

Deep Learning

Deep learning is a subset of machine learning based on artificial neural networks. It encapsulates all neural networks consisting of three or more layers. It removes some of the dependency on human experts in machine learning, where hidden representations require manually crafted features. Deep learning algorithms are able to learn these hidden representations, automating the feature extraction process. As most of the data preprocessing that is typically involved with machine learning is eliminated, deep learning approaches can ingest and process unstructured data, like text and images, in its raw form. Deep learning classifiers have seen an unprecedented rise in popularity in recent years due to promising results in a number of research fields.

The idea of *Recurrent Neural Networks* (RNN) have been around since the mid-1900s and have proven particularly effective for tasks including sequential data. This results from their feedback connections in the recurrent layer, providing "memory" to the network [40]. The "memory" means that the network outputs do not rely solely on the current input given to the network but are conditional on the recent context in the input sequence. The earliest adoption of RNNs in the context of fake news detection, specifically rumor detection, is reported in [30].

Convolutional Neural Networks (CNN) are composed of an input layer, an output layer, and a series of hidden layers, where a number of transforming operations are applied to the data by means of pooling and convolution operation. These operations are repeated over several hidden layers, with each layer learning to identify unique features specific to the training data. Though mainly applied to image recognition and processing [41], CNNs have recently gained traction in the NLP community [42]. CNNs utilizing word embeddings have been proposed for solving both stance and veracity classification of social media posts [43].

The *Transformer* architecture [44] has become immensely popular in the research field and has proven to be especially effective for common NLP tasks. Transformers are based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. They utilize the encoder-decoder structure; where the encoder maps an input sequence $(\vec{x_1}, \vec{x_2}, ..., \vec{x_n})$ to a sequence of continuous representations $\mathbf{z} = (\vec{z_1}, \vec{z_2}, ..., \vec{z_n})$. Then, one element at a time, the decoder generates an output sequence $(\vec{y_1}, \vec{y_2}, ..., \vec{y_m})$ of symbols given the provided $\mathbf{z}$. The model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. The Transformer architecture uses stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. Well-known transformer architectures, such as *Bidirectional Encoder Representations from Transformers* (BERT) [45], have been successfully adapted to the task of fake news detection [46] [47].

Non-classification Techniques

Research also exists on alternative approaches to classification, which can represent valuable alternatives by possibly exploiting different characteristics of false information. These approaches exploit models such as clustering and vector space models in order to identify and explain the properties of fake news and rumors [16].

The *Gini Coefficient* [48] is used to measure the inequality among values of a frequency distribution. It has been used to analyze online social networks to identify metrics to infer cues of deception. One such example is [49], where they use the Gini coefficient to evaluate the credibility of tweets by measuring disparity in retweet behavior. First, a retweet graph is built using only credible tweets and their retweets. Then, to compute the score of acceptability for any given tweet, the *PageRank* algorithm [50] is used. The information provided by this approach is not used in classification per se but presented to users accommodating a more informed evaluation of the tweet.

Tensor Decomposition [51] has recently been proposed as a tool to detect fake news. Tensors and their decompositions appeared as early as 1927 but remained unused in computer science until the late 20th century [52]. For further explanation of tensor decomposition, see subsection 2.7.2. Recently, numerous approaches to perform unsupervised or semi-supervised fake news detection have been proposed, utilizing tensor decomposition as their primary mechanism. Tensors can be used in order to separate documents that are similar and the outliers which may represent fake news. This approach was taken in [53], where news articles were decomposed into tensors able to model spatial relations between terms in a document via *CANDECOMP/PARAFAC decomposition* [54]. Then, co-clustering was performed to identify latent groups of articles that fall under coherent categories of false news.

Computational-Oriented Fact-Checking

Some approaches to fake news detection rely on extracting textual content features and comparing them to a knowledge base. These approaches are instances of *knowledge-based* detection techniques and adopt a process known as computational-oriented fact-checking [23], also referred to as automatic fact-checking [55].

Automatic fact-checking approaches aim to use external sources to fact-check the truthfulness of the claims in news content. Fact-checking can be utilized as a tool in both classification and non-classification approaches. As seen in Figure 2.3, these approaches can generally be divided into two stages;

1. Identifying and extracting check-worthy claims from the document
2. Discriminating these claims based on their estimated veracity

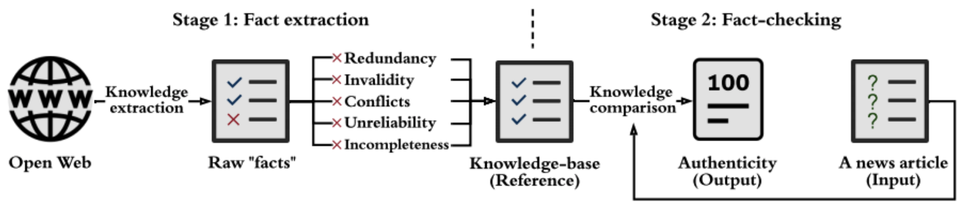


Figure 2.3: Automatic news fact-checking process [55]

To identify check-worthy claims, factual claims in news content first need to be extracted. This is a complex task consisting of many subtasks. The process of claim extraction is further explained in section 2.5.

Several strategies have been proposed to estimate the veracity of claims. The most widely used technique is through the exploitation of knowledge graphs. For an introduction to knowledge graphs, see section 2.4. In the context of knowledge bases and graphs, claims are often referred to as triples. Approaches vary but generally use some algorithm in order to check claims against the knowledge graph. The knowledge graph is used to provide the ground truth in fact-checking, i.e., it is assumed that every existing triple in the knowledge graph represent true claims. However, for non-existing triples, their authenticity relies on what assumption is made. There are mainly two possible assumptions stemming from database integrity theory on completeness [56] [57]:

- *Closed-world assumption*: assumes a triple not observed in the graph as false.
- *Open-world assumption*: assumes a triple not observed in the graph as unknown, i.e., the corresponding relationship could be either true or false.

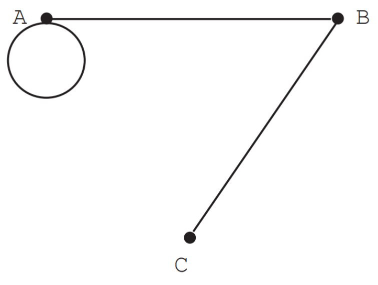
2.3 Graph Theory

The first paper on graph theory was published in 1736 by Leonhard Euler on the Seven Bridges of Königsberg problem[58][59]. Since then, the field has been expanded, and there are various types of graphs, each with its distinct definition. A general trait for graphs is that they contain a set of objects and their relations. Typically, a graph is depicted with a diagram where dots represent objects and lines between the circles represent relations, as shown in Figure 2.4a. In order to reason about and specify our domain, we define the simplest form of graphs.

Definition 2 (Graph)**.** *A graph G is a set $G = (V, E)$ where;*

- *V is a finite set, called vertices of G*
- *E is a finite set, called the edges of G*
- *$\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ is an adjacency matrix, mapping nodes $u \in V$ and $v \in V$ as edges $(u, v) \in E$.*

In undirected graphs, $\mathbf{A} = \mathbf{A}^{-1}$ and the presence of a edge between u and v is denoted by $\mathbf{A}[u, v] = 1$ otherwise $\mathbf{A}[u, v] = 0$.



(a) A graph with loop [60]

$$\begin{array}{c|c||c|c|c}
 & & \multicolumn{3}{c}{\overbrace{\hphantom{\mathbf{A}\quad\mathbf{B}\quad\mathbf{C}}}^{u}} \\
\hline
 & & \mathbf{A} & \mathbf{B} & \mathbf{C} \\
\hline
 & \mathbf{A} & 1 & 1 & 0 \\
v & \mathbf{B} & 1 & 0 & 1 \\
 & \mathbf{C} & 0 & 1 & 0
\end{array}$$

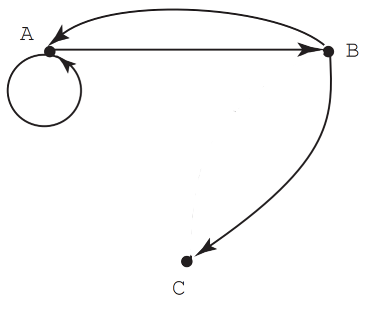
(b) Corresponding adjacency matrix

Figure 2.4: Simple graph with loop and adjacency matrix

In other words, a graph is a nonempty finite set of vertices V, and a finite set E of two-element or one-element subsets of V. Two-element subsets denote an edge from a vertex x to a different vertex y. Single-element subsets denote an edge from a vertex x to itself, often referred to as a *self-loop*. The definition above is an *undirected graph*; hence all edges are bidirectional, meaning nodes have a mutual relationship.

2.3.1 Directed Graph

A directed graph, also known as a *digraph*, follows the definition from undirected graphs section 2.4, except its edges have orientations. In the field of mathematics this is called a *binary relation*. A binary relation on a set V is simply a subset of $V \times V$[60]. This means that the adjacency matrix is no longer necessarily symmetric and can have one-way relationships.



(a) A directed graph with loop [60]

$$\begin{array}{c|c||c|c|c}
 & & \multicolumn{3}{c}{\overbrace{\hphantom{\mathbf{A}\quad\mathbf{B}\quad\mathbf{C}}}^{u}} \\
\hline
 & & \mathbf{A} & \mathbf{B} & \mathbf{C} \\
\hline
 & \mathbf{A} & 1 & 1 & 0 \\
v & \mathbf{B} & 1 & 0 & 1 \\
 & \mathbf{C} & 0 & 0 & 0
\end{array}$$

(b) Corresponding adjacency matrix

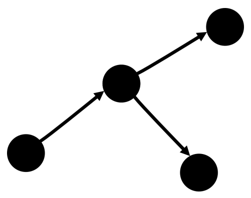
Figure 2.5: Directed graph with loop and adjacency matrix

In directed graphs, an arc going from vertex x to y denotes that x is the *tail* of the relationship while y is the *head*. For example, in Figure 2.5a, one would say that there is an arc going from the tail B to the head C.

2.3.2 Multi-Relational Graphs

Beyond the distinction between directed and undirected graphs, one also considers graphs with edges of different types. In these cases, we extend the edge notation to include an edge or relation type τ, e.g. $(u, \tau, v) \in E$, and we can define one adjacency matrix $\mathbf{A}_\tau$ per edge type. These graphs are called *multi-relational* graphs. Since we now have multiple adjacency matrices for each relation, we order them in a third dimension by an adjacency tensor $\mathcal{A} \in \mathbb{R}^{|V|\times|R|\times|V|}$, where R is the set of relations[61].

Graphs can be either *homogeneous* or *heterogeneous*. The graphs from the previous subsections have been homogeneous graphs, where the vertices in V represent instances of a single type, and all the edges in E represent relations of a single type. The graphs can also be viewed as having the absence of types, as the types do not add any structural information to the graph. An example of a homogeneous graph is a social network where all vertices represent "authors", and all edges represent "citations". If the social network is extended to be multi-relational, e.g., having the two edge types "citations" and "collaborators", the graph becomes heterogeneous. The nodes may have multiple types, also making a graph heterogeneous. This can be formalized with graph labeling, where we define R as the set of relation types, and $\mathcal{E}$ as the set of vertex types. A homogeneous graph then has $R = 1$ and $\mathcal{E} = 1$, while a heterogeneous graph has either $R = N$, $\mathcal{E} = M$, or both, where $N, M > 1$.


(a) Depiction of a homogeneous graph

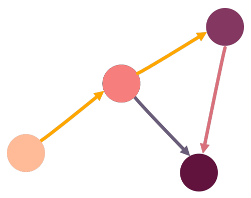

(b) Depiction of a heterogeneous graph

Figure 2.6: Juxtaposition of naive illustrations of a homogeneous graph and heterogeneous graph

2.3.3 Multi-graph

Graphs that are permitted to have multiple edges (also called *parallel edges*) connect vertices multiple times to other vertices. These graphs are called *multi-graphs*. To avoid ambiguity, multi-graphs that allow loops are normally called *pseudo-graphs*; however, we will use the term multi-graph as a synonym. Multi-graphs allows for vertices to be connected by several types of edges, thus creating a rich semantic network which can describe complex systems.

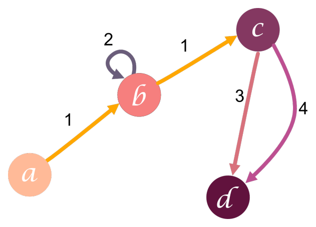


Figure 2.7: A labeled directed heterogeneous multi-graph permitting loops

2.3.4 Network Graph

When applying graphs to real-world applications, one often talks about *graph networks*. A tremendous benefit of graph networks is that many real-world datasets can be represented as graphs. Researchers have applied this technique to a wide variety of domains, including geosciences [62], decision support systems [63], bioinformatics [64], neuroscience [65] [66], cheminformatics [67], social networks [68], recommender systems [12], epidemiology [69], cybersecurity [70], and more. The reason is that real-world objects can be represented as vertices by, e.g., their name, and be extended through other nodes, as seen in Figure 2.8. This is a great way to capture the interconnectivity that so many datasets possess. The network in Figure 2.8 is a particular type of network called *Knowledge Graph*.

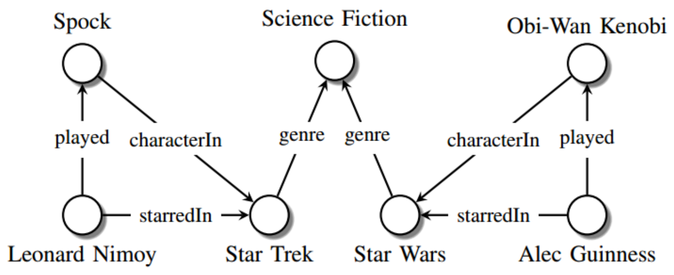


Figure 2.8: Representation of knowledge graph network [71]

2.4 Knowledge Graphs

We can represent statements about semantic data in the form of (subject, predicate, object) triples, where the subject and the object are named entities that have some predicate between them representing the relation [72]. These triples have many names, such as semantic triple, RDF-triple, or simply triple. A set of such triples is known as a *knowledge base*. They can be connected to form a multigraph where nodes represent the entities, and directed edges represent the relations. This results in a *heterogeneous semantic network*, often referred to as a *knowledge graph*. In this thesis we will borrow a slightly modified definition from knowledge graphs from [11].

Definition 3 (Knowledge Graph). *A knowledge graph is a set $KG = (\mathcal{E}, \mathcal{R}, \mathcal{G})$ where;*

- *$\mathcal{E}$: a set of nodes representing entities;*
- *$\mathcal{R}$: a set of labels representing relations;*
- *$\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$: a set of facts represented as edges connecting pairs of entities. Each fact is a triple $\langle h, r, t \rangle$, where h is the head (subject), r is the relation(predicate), and t is the tail (object).*

A large amount of world knowledge has been accumulated in several publicly available knowledge graphs, constructed in one of two ways, either manually or automatically [73]. Knowledge graphs such as DBpedia[1] or Freebase (succeeded by WikiData[2]) are authored by humans, making the database have little or no noisy facts. Automatically curated knowledge graphs such as Google KG uses machine learning tools to populate the graph, which requires much less effort from humans, but can introduce errors such as including wrong information[3]. A downside of manual construction methods is that they do not scale well compared to the automated approaches. Furthermore, knowledge graphs can be differentiated on whether they employ a fixed or open lexicon of entities and relations [71].

Schema-based	*Schema-based* knowledge graphs use globally unique identifiers for entities and relations. Here, all valid relations are predefined in a fixed vocabulary. Freebase is an example of a schema-based, manually created knowledge graph. In this knowledge graph, we might represent the fact that Hawaii is the birthplace of Barack Obama using the triple $\langle /m/02mjmr,\ /people/person/born-in,\ /m/03gh4 \rangle$, where $/m/02mjmr$ is the unique id for the named entity $Barack\ Obama$.
Schema-free	*Schema-free* approaches use open information extraction (OpenIE) techniques to identify entities and relations. These entities and relations are represented via normalized but not disambiguated strings. As a result, OpenIE knowledge graphs may contain multiple triples with the same semantic meaning, e.g., $\langle Obama,\ born\ in,\ Hawaii \rangle$ and $\langle Barack\ Obama,\ place\ of\ birth,\ Honolulu \rangle$. The main disadvantage of OpenIE systems is that this representation does not clarify whether the first triple refers to the same person as the second triple, nor whether "born in" has the same semantic meaning as "place of birth".

[1] https://wiki.dbpedia.org/

[2] https://www.wikidata.org/wiki/Wikidata:Main_Page

[3] https://www.theatlantic.com/technology/archive/2019/09/googles-knowledge-panels-are-magnifying-disinformation/598474/

Despite being widely used for research purposes, the Freebase is missing data on the place of birth for over 70% of all people included, even though this is a mandatory property of its schema [74]. This issue is not specific to Freebase, as other knowledge graphs are similarly incomplete. Consequently, efforts have been made to utilize the underlying properties of networks to infer missing facts, or links between entities; also referred to as link prediction, or knowledge graph completion. See section 2.7 for an in-depth description of the task. This procedure is necessary to improve upon the quality of the knowledge graphs being built. Generally, knowledge graphs excel due to the inherent capturing of context. They break the standard "store and retrieve" pattern of NoSQL databases by allowing for the utilization of connections in the dataset. Every time data is added to a knowledge graph; the entire data ecosystem is enriched because the new entity or relation is connected to everything else. In general, the more data available, the more context.

2.5 Triple Extraction

Triple Extraction is a subset of *information extraction* [75], where the goal is mining triples from a text, using various forms of natural language processing and data science techniques. The triples produced follow the schema-based format described in section 2.4. As with the construction of knowledge graphs one needs to decide whether to employ a fixed or open lexicon for entities and relations. Some of the tools and techniques commonly used will be highlighted below, manifested in chapter 4, and finally realized in chapter 5.

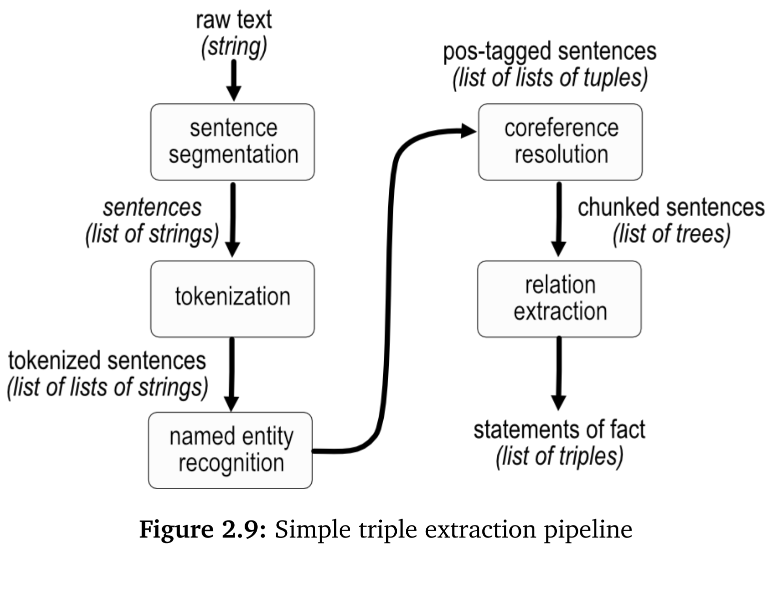


Figure 2.9: Simple triple extraction pipeline

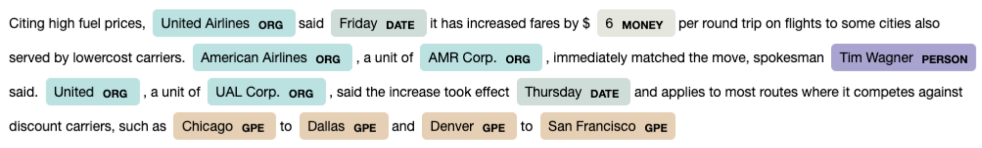
By using the architecture in Figure 2.9 as a baseline, one can illustrate most information extraction pipelines. The raw text is split into an array of sentences where each sentence is further subdivided into individual words, often called *tokens* [76, Chapter 7.1]. The tokens are given part-of-speech tags in *named entity recognition* (NER) which will be used in the next step, named *coreference resolution*. Next, potentially redundant entities are identified and replaced. At the end of the pipeline, *relation extraction* discovers likely relations between the newly tagged entities.

In the context of knowledge graph construction, the raw texts used as input to the pipeline in Figure 2.9, are often gathered from online sources. The texts can be from a single reliable source, such as Wikipedia, making it relatively efficient, but will suffer from knowledge base incompleteness [55]. On the other hand, there is open-source triple extraction, which aims to combine knowledge from several distinct sources. This makes it less efficient but leads to a more complete knowledge graph. Some knowledge graphs, such as NELL [77], uses a combination of both. The facts extracted from a single source must pass some confidence threshold, while candidate facts found from multiple sources can pass with lower confidence thresholds.

The following subsections present the main components and techniques of a triple extraction framework in more detail.

2.5.1 Named Entity Recognition

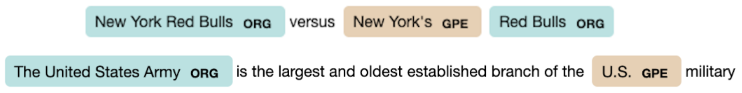
The NER task consists of finding each span of document that constitutes a named entity and then classifying the type of that entity. Named entities are singular identifiable and separate objects; in other words, anything we can refer to with a proper noun [78, p. 148]. Common types of entities that can be extracted are people, organizations, and places. In addition, we extend the definition to include temporal and numerical expressions such as dates, numbers, etc.



Citing high fuel prices, United Airlines ORG said Friday DATE it has increased fares by $ 6 MONEY per round trip on flights to some cities also served by lowercost carriers. American Airlines ORG , a unit of AMR Corp. ORG , immediately matched the move, spokesman Tim Wagner PERSON said. United ORG , a unit of UAL Corp. ORG , said the increase took effect Thursday DATE and applies to most routes where it competes against discount carriers, such as Chicago GPE to Dallas GPE and Denver GPE to San Francisco GPE

Figure 2.10: Named entity annotated document with entity type labels

There are several factors making the NER task challenging. First, multiple entities extracted from the document may refer to the same real-world entity, such as "The United States" and "U.S", or "William Shakespeare" and "He". To handle this, a procedure named coreference resolution is conducted. This procedure is explained in the following subsection. Further, other problems occur due to ambiguity, such as *type ambiguity* and *segmentation ambiguity*. Type ambiguity refers to the issue of the same mention possibly referring to several different named entities. Segmentation ambiguity refers to difficulties in defining the boundaries of the span of document that constitutes the named entity. Figure 2.11 and Figure 2.12 present examples of segmentation and type ambiguity.


New York Red Bulls ORG versus New York's GPE Red Bulls ORG

The United States Army ORG is the largest and oldest established branch of the U.S. GPE military

Figure 2.11: Segmentation ambiguity

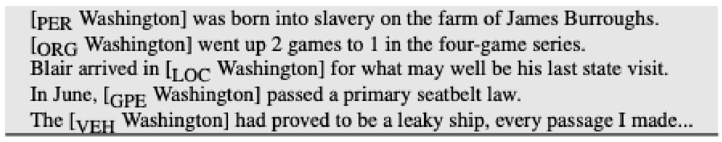
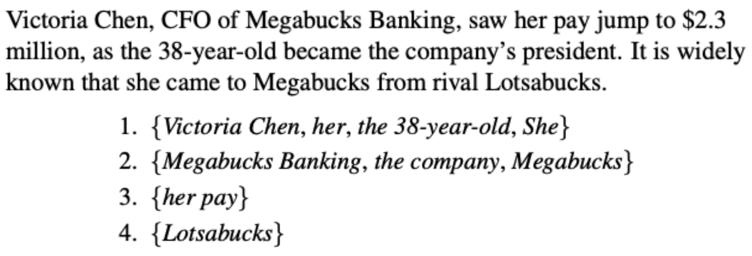

[PER Washington] was born into slavery on the farm of James Burroughs.
[ORG Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [LOC Washington] for what may well be his last state visit.
In June, [GPE Washington] passed a primary seatbelt law.
The [VEH Washington] had proved to be a leaky ship, every passage I made...

Figure 2.12: Type ambiguity for the word *Washington*, from [78, p. 154]

NER can be formulated as a task of *sequential token labeling* [78, p. 148]. Much like the task of part-of-speech tagging and chunking, the assigned label encapsulates both the token type and its context. Common approaches vary from the utilization of linear statistical models to neural network models. Some examples of approaches used to conduct NER are; Hidden Markov Models [79], Maximum entropy Markov models [80], Conditional Random Fields [81], and Bidirectional LSTMs [82].

2.5.2 Coreference Resolution

An imperative part of named entity recognition is the task of identifying mentions or referring expressions and their discourse entity [78, p. 415]. If several referring expressions refer to the same discourse entity, each of these is said to *corefer*. To determine whether two mentions corefer is referred to as coreference resolution. This is often done in two steps. First by identifying the mentions, and then by clustering them into coreference chains, i.e., the set of corefering expressions.


Victoria Chen, CFO of Megabucks Banking, saw her pay jump to $2.3 million, as the 38-year-old became the company's president. It is widely known that she came to Megabucks from rival Lotsabucks.

1. {*Victoria Chen*, *her*, *the 38-year-old*, *She*}
2. {*Megabucks Banking*, *the company*, *Megabucks*}
3. {*her pay*}
4. {*Lotsabucks*}

Figure 2.13: Coreference chains from document, from [78, p. 416]

Referring expressions can be of several different types. *Anaphoric references* occur when a referring expression points back to a referent [78, p. 416], while *cataphoric references* occur when a referring expression comes before their referent [78, p. 418]. When mentions are referring to a prior mention, we call it an *antecendent*.

Another kind of coreference task is that of event coreference, which is used to tell whether two event mentions refer to the same event. Coreference resolution is often done in combination with the task of mapping entities to some real-world individual, a task called *entity linking*. Approaches for coreference resolution vary from using rule-based dependency parse trees, to neural networks trained with word embeddings and distance between mentions as features [78, Chapter 21].

2.5.3 Relation extraction

Relation extraction is the task of determining the predicate between two given entities from a text to a knowledge graph [83]. For example, in the sentence "Barack Obama was born in Honolulu, Hawaii," a relation extraction system aims to infer the semantic relationship, "bornInCity" between the entities "Barack Obama" and "Honolulu". The relation extraction approaches that exist today mainly rely on the multi-instance, and distant learning paradigms [84].

Multi-instance relation extraction looks at previous occurrences of entity pairs when given a sentence, then predicts a target relation [85]. While this method can significantly help a relation extraction system determining suitable ties, it can also add noise by using realistic but wrong relations and thus hurt the overall performance [86]. On the other hand, *sentential relation extraction* [87] does not consider previous mentions and make predictions on a sentence level, therefore relying on local features or context. Relation extraction is the critical component for building relational knowledge graphs, and it is of crucial significance to NLP applications such as structured search, sentiment analysis, question answering, and summarization [88].

2.5.4 Post extraction

There may still be some required steps after the various forms of natural language processing and data science techniques are applied to extract triples from the texts. A recurring issue is the varying quality of the extracted triples. Some common issues were unveiled in [55], and presented here.

Redundancy A triple is said to be redundant if it gives no extra information to the knowledge graph. An example of this is having the presence of the triples *(Joseph R. Biden Jr., profession, President)* and *(Joe Biden, profession, President)* as the heads in both triples map to the same entity. Redundancy reduction is sometimes referred to as entity resolution [89], which is also known as deduplication or record linkage [90]. Ideally, coreference resolution should combat this issue, however implementations of coreference resolution are not perfect, and therefore might produce duplicates.

Invalidity Some triples may be dependent on temporal validity, i.e. being valid in only some point in time. This is especially important when working with news, which often covers new information or change of currently available information. The correctness of some facts might therefore be dated at the time of extraction. For example, *(USA, join, Paris Agreement)* has been the subject of temporal invalidity from 2017 to 2021. A solution to this is to allow facts to have a beginning and ending date as suggested by [91], or one can reify current facts by adding extra assertions to them [92].

Conflicts Triples extracted might also have conflicting knowledge. A topical example for fake news would be *(Barack Obama, bornIn, Hawaii)* and *(Barack Obama, bornIn, Kenya)*. In automatic triple extraction these types of conflicts might be resolved by multi-criteria decision-making methods [93][94][95][96].

Unreliability The source of triple extraction is an important factor. A triple extracted from The Onion[4] a satire news organization, will not be as reliable as triples extracted from i.e. The New York Times[5]. To avoid using web sites with unreliable content one can use credibility tools to filter the sources used. Expert systems such as NewsGuard or sites as Media-rank[6] highlights the credibility of websites.

[4]`https://www.theonion.com`
[5]`https://www.nytimes.com`
[6]`https://media-rank.com`

2.6 Distributional Hypothesis

"*You shall know a word by the company it keeps*"

— J.R. Firth (1957)

The *distributional hypothesis* relies on the assumption that there is a correlation between distributional similarity and semantic similarity. This assumption implies that we can analyze the distributional similarity of elements in the order to estimate their semantic similarity. The distributional hypothesis forms the basis for *statistical semantics*; the statistical study of meanings of words and their frequency and order of recurrence [97]. It has been a key contributing factor for the advancements in the NLP field. As most ML models require input to be represented as a fixed-length feature vector, many methods of creating such representations of words have been introduced. Vector representations are often, and will in this thesis, be denoted using arrow oversets, e.g $\vec{x}$. Some NLP systems and techniques disregard the notion of similarity between words by representing them as sparse, long vectors with dimensions corresponding to the words in the vocabulary. These models are often referred to as *bag-of-word models* (BOW) and have historically had great success for some problems but rendered insufficient in most.

2.6.1 Word Embedding

The disregard of semantic similarity in BOW models has resulted in a shift towards more advanced techniques. Among these techniques is the notion of distributed representations of words, also called *word embeddings*. This technique aims to represent words through lower-dimensional, fixed-sized, dense vectors, omitting the dimensionality problem of the previous approach. The fixed-size vector of words are mapped to a multidimensional semantic space, where vectors closer to each other carry more semantic similarities. This is illustrated in Figure 2.14, where neutral, positive, and negative words are located in distinct areas of the projected space. Research has shown that dense vectors perform better than sparse vectors for every NLP task [78].

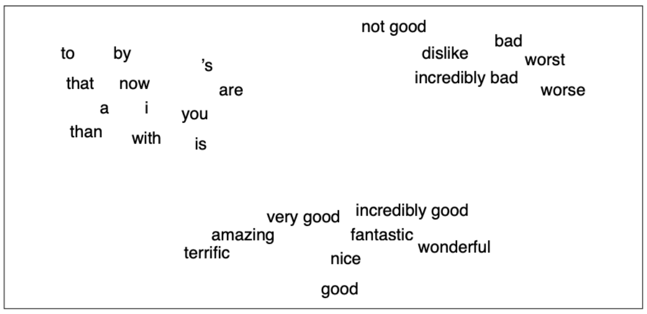


Figure 2.14: Two-dimensional projection of 60-dimensional embeddings trained for sentiment analysis [78]

The most common way of defining the similarity between two embeddings is by calculating the cosine of the angle between their vectors projected in a multi-dimensional space. The cosine is based on the dot product operator from linear algebra normalized for the vector length. This is done by dividing the dot product by the lengths of both vectors. For example, given two vectors $\vec{v}$ and $\vec{w}$, their *cosine similarity metric* is defined as:

$$cosine(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N}\vec{v}_i\vec{w}_i}{\sqrt{\sum_{i=1}^{N}\vec{v}_i^2}\sqrt{\sum_{i=1}^{N}\vec{w}_i^2}} \tag{2.1}$$

Word2Vec

A number of procedures have been proposed to encapsulate semantic similarities of words through word embeddings. Among the most popular procedures is the *Skip-Gram with Negative Sampling* (SGNS) algorithm [98], often loosely referred to as *word2vec*. This procedure trains a classifier on the binary prediction task:

"Is word w likely to show up near word x?"

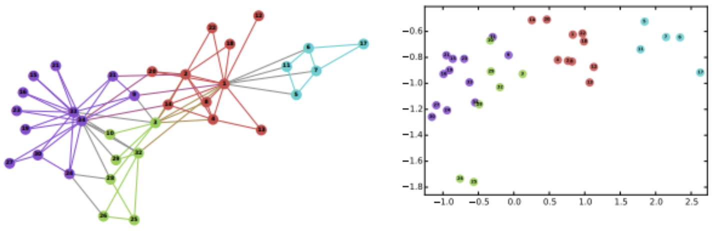
A sliding window size is selected, and based on this window size the algorithm attempts to identify the conditional probability of observing the output word based on the surrounding words within the window. The classifier learns statistics from the number of times each pairing of words are observed. In lieu of keeping focus on the prediction task, the learned classifier weights are extracted and used as word embeddings. This model omits the need for labeled datasets as it can use the text corpora as implicitly supervised training data for the classifier. A word s from our corpora that occurs near the target word x acts as the label to the classifier's prediction task.

2.6.2 Graph Embedding

Graph embedding is an approach that is used to transform nodes, edges, or the entire graph into fixed-size dense feature vectors. The aim of graph embeddings is to efficiently encapsulate the graph's topological properties, as well as the connectivity and attributes of nodes and edges [99]. Finding the optimal dimension for the feature vectors is challenging. Choosing a higher dimensionality may increase the reconstruction precision, but will also result in high time and space complexity. Multiple approaches for graph embedding have been proposed following the idea of distributed representation learning and the previous success of word embeddings. We define three categories among node embedding approaches; *factorization approaches*, *random walk approaches*, and *deep approaches*. Next, we present two of the most popular graph embedding algorithms.

DeepWalk

DeepWalk [100] utilizes a combination of the SGNS model and random walk process in the graph embedding process. It uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences. The procedure utilizes short random walks to omit the need for global recomputation on small changes in the graph. The model can then be iteratively updated with new random walks from the changed region in time sublinear to the entire graph.



(a) Input: Karate Graph (b) Output: Representation

Figure 2.15: DeepWalk used to generate a latent representation of the Karate network [101] in $\mathbb{R}^2$

The DeepWalk algorithm consists of two main components; a random walk generator followed by an update procedure. The random walk W_v starts by selecting any node v as the root of the random walk. The walk then samples a random neighboring node $w \in N(v)$ and repeats the process for a defined number of steps t. For each node v_i, we generate a random walk $|W_{v_i}| = t$, and then use a skip-gram network with *hierarchical softmax* to approximate the probability distribution in order to update representation vectors. A constant γ defines the number of times this procedure is done for each node.

Node2Vec

The *node2vec* algorithm [102] introduces a semi-supervised approach for scalable feature learning in graphs. The algorithm differs from the approach described in DeepWalk by introducing *Breadth-Fast-Sampling* (BFS) and *Depth-First-Sampling* (DFS) to control the random behavior of the walk component. In node2vec a flexible neighbor sampling strategy is employed by smoothly interpolating between BFS and DFS. The intuition is to enable modeling of the two kinds of graph similarities: *homophily* and *structural equivalence*. This is motivated by the findings that real-world networks exhibit both behaviors [102]. Homophily refers to the notion that nodes that are highly interconnected and belong to similar network clusters should be embedded closely together. Structural equivalence refers to the notion that nodes with similar structural roles in networks should be embedded closely together.

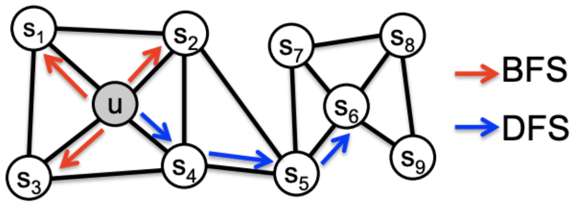


Figure 2.16: BFS and DFS search strategies from node u (k = 3) [102]

The node2vec algorithm consists of three distinct phases, executed sequentially. First, preprocessing is done to compute transition probabilities. Then, r random walks of fixed length l starting from every node are conducted. This is done to offset the implicit bias due to the choice of the start node u. Lastly, the objective is optimized using *stochastic gradient descent* with context size k. To provide flexibility in the random behavior of the walk component, node2vec defines a 2^{nd} order random walk with two parameters p and q to guide it. The in-out parameter q allows the search to differentiate between "inward" and "outward" nodes. Choosing smaller values for q tends to produce random walks with DFS-like exploration behaviors. The return parameter p controls the likelihood of immediately revisiting a node in the walk. Low values for p promote "local" walks close to the starting node, while high values ensure that we are less likely to sample already visited nodes in the next two steps.

Given a walk that just traversed edge (u, v) to get to node v, the next step is chosen by evaluates the transition probabilities π_{vx} on edges (v, x) leading from v. The unnormalized transition probability is defined by $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where w_{vx} is the static edge weights, d_{tx} denotes the shortest path distance between nodes t and x. Search bias $\alpha_{pq}(t, x)$ is given by

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{2.2}$$

2.7 Link Prediction

The *Link Prediction* (LP) task aims to infer missing triples by analyzing those already present in the graph [103]. For instance, in a social network, one might only know a subset of people are friends, and some are not, and seek to predict which other people are likely to become friends. All link prediction tasks operate under the open world assumption, which does not presume that the knowledge of a domain is complete. Hence, any triple not observed in the graph remain possible. There are mainly two angles taken when researching this problem; graph structure or attributes of nodes and edges [104]. The structure of a graph encapsulates its topology and refers to the way nodes are interconnected. In general, the modeling process constitutes the extraction of local or global connectivity patterns between nodes. Prediction is carried out by utilizing these patterns to generalize the observed edges between a specific node and all others [105]. The other approach aims to utilize attribute information of nodes and edges to make link predictions. Most existing models are designed to operate on feature vector representation of the constituents (nodes and edges) of the graph [106].

The link prediction problem can be formalized in several ways for different problems. Liben-Nowell and Kleinberg proposed that the link prediction problem could describe how a network changed over time [107]. The problem has later been further divided into three different types; adding links to the graph, removing links from the graph, or doing both. Other applications have characterize link prediction as the task of predicting the correct entity that completes a triple in the graph [11]. Subsampling real-world networks is a typical way of obtaining the graph datasets employed in link prediction research.

For the link prediction problem, we define the graph:

Definition 4 (Graph). *A graph G is a set $G = (\mathcal{E}, \mathcal{R}, \mathcal{G})$ where;*

- *$\mathcal{E}$: set of nodes representing entities*
- *$\mathcal{R}$: set of edges representing relations*
- *$\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$: set of triples $\langle h, r, t \rangle$*

In the graph, two nodes and the edge between them collectively represent a triple. As for most ML approaches, the set of triples $\mathcal{G}$ is divided into three disjoint subsets: a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{val}$, and a test set $\mathcal{G}_{test}$. Given a known node (source entity), an edge (relation), and an unknown node (target entity), we can conduct link prediction tasks. For directed graphs, we conduct either head $\langle ?, r, t \rangle$ or tail $\langle h, r, ? \rangle$ prediction. This distinction is redundant for undirected graphs. For ML approaches, the graph needs to be embedded to enable models to train on it. A scoring function ϕ is defined to estimate the plausibility of any triple $\langle h,\ r,\ t \rangle$ using their embeddings;

$$\phi(h, r, t) \tag{2.3}$$

This scoring function can be utilized to predict the most plausible node or edge to complete a triple in the graph:

$$h = \underset{e \in \mathcal{E}}{argmax}\ \phi(e, r, t) \tag{2.4}$$

$$t = \underset{e \in \mathcal{E}}{argmax}\ \phi(h, r, e) \tag{2.5}$$

$$r = \underset{\hat{r} \in \mathcal{R}}{argmax}\ \phi(h, \hat{r}, t) \tag{2.6}$$

Over time, many approaches to LP have been proposed. For the triple completion perspective of link prediction, [11] has proposed a simple taxonomy for separating the different approaches into three categories depending on the technique applied. This taxonomy is illustrated in Figure 2.17. The following subsections will describe the three categories of the proposed taxonomy. Refer to the original paper for a more thorough review of subgroups, constraints, space complexity, and so forth.

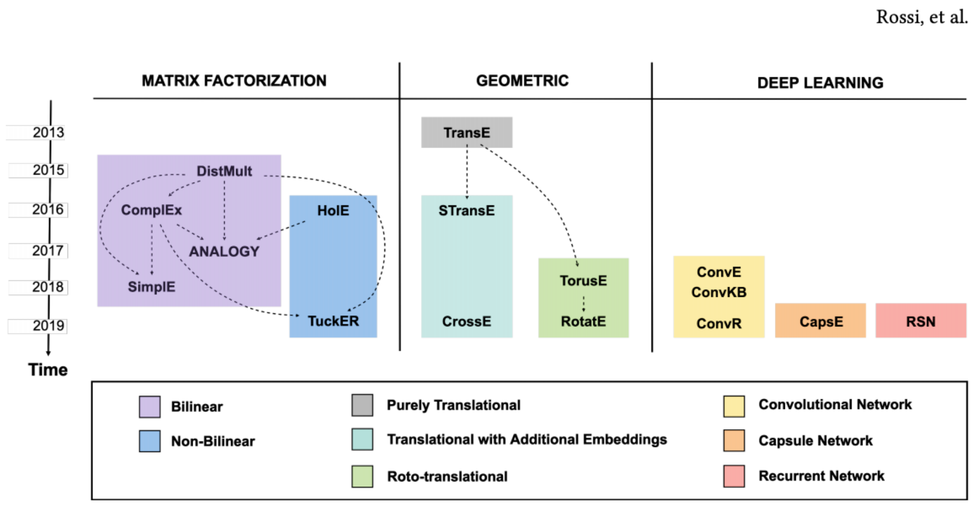


Figure 2.17: Proposed taxonomy for LP approaches [11]

2.7.1 Geometric Models

The *geometric models* are characterized by their view of graph edges as geometric transformations in the latent space. Given a triple $\langle h, r, t \rangle$, the embedding of the head node undergoes a spatial transformation τ that uses the values of the edge embedding r as parameters. The distance between the resulting vector and the tail node's vector represents the triple score. This distance is computed using a distance function δ. Thus, the scoring function ϕ is given by

$$\phi(h, r, t) = \delta(\tau(\vec{h}, \vec{r}), \vec{t}) \tag{2.7}$$

2.7.2 Tensor Decomposition Models

Tensors are often viewed as a generalization of vectors and matrices and are generally understood as multidimensional arrays. A tensor of order N in M-dimensional space is a mathematical object with n indices and M^n components [108]. Put in simple terms, a first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are called higher-order tensors.

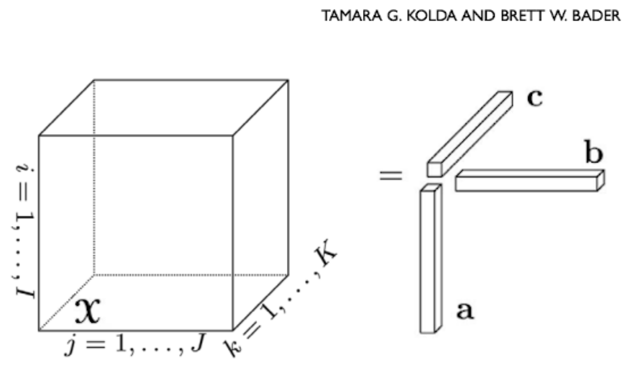


Figure 2.18: Third-order tensor: $\mathbf{X} = \vec{a} \cdot \vec{b} \cdot \vec{c} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J} \times \mathcal{K}}$, where the $(i,\ j,\ k)$ element of X is given by $x_{ijk} = a_i b_j c_k$ [108]

In *tensor decomposition models*, graphs are considered third-order tensors, also thought of as 3-dimensional adjacency matrices. This tensor can be decomposed into a combination of low m-dimensional vectors representing node and edge embeddings. To calculating the score of a triple, one can utilize an operation such as the bilinear product on the involved embeddings. During training of the LP model, we optimize the scoring function ϕ for all triples in $\mathcal{G}_{train}$. The idea is that the learned embeddings should generalize to the structure of the graph. Hence, the scoring function should assign high probability triples representing true statements, disregarding whether they have been observed in the graph or not. Equally, a low probability should be assigned to false triples.

2.7.3 Deep Learning Models

Deep Learning models are increasingly capable of solving predictive tasks on graphs, including the task of link prediction. Many types of neural network architectures have been utilized for the link prediction problem. These models learn parameters such as weights and biases, and combine them with the input data to recognize significant patterns. Deep networks organize parameters into separate layers, generally interspersed with non-linear activation functions. For deep learning based link prediction models, graph embeddings are learned jointly with the layer's weights and biases. Though [11] names three types, we extend this taxonomy to include a fourth: Convolutional Neural Networks, Capsule Neural Networks, Recurrent Neural Networks, and Graph Neural Networks. In section 3.2, the different neural network types are further explained in the context of LP. Graph Neural Networks are explained in detail in section 2.8.

2.8 Graph Neural Networks

The *Graph Neural Network* (GNN) is a general framework for defining deep neural networks on graph data. In graphs, the connections between data make the standard assumption of datapoints being *independent and identically distributed* (i.i.d.) invalid. This makes most of the standard machine learning models ineffective as their derivations are firmly based on the assumption that each datapoint is statistically independent from all the other datapoints. Hence, the need for different architectures for machine learning on graph-structured data with the ability to model their structural relationships, as well as any feature information that might be available. In general, GNN models take as input a graph G representing the data. This representation has in advance been embedded through creating distributed vector representations for each node in the graph. The output of a GNN model will be the same graph as the input, but for each node, the vector representation no longer represents the node itself but how this node belongs in the context of the graph.

Several variants of Graph Neural Networks have been suggested, either focused on differences in graph types, training methods, or method of the propagation step [109]. Among the first CNN-based architectures that could operate on graphs was the *Message Passing Neural Network* (MPNN) [67]. The author's implementation relied on message passing to extract valuable information from graph molecules. [110] again proposed using MPNN, which could generalize several Graph Neural Network and Graph Convolutional Network approaches.

The output representation from a GNN can be used in multiple tasks such as; *node classification*, *graph classification*, and link prediction. See section 2.7 for an in-depth description of the link prediction task. The two other tasks mentioned are described below:

Node Classification	The goal is to predict the label y_u, which could be the type, category, etc., associated with node $u \in V$, given a set of existing node labels [61]. We are only given the true labels of a subset of nodes $V_{train} \subset V$, and the task is to predict the unlabeled nodes using this information. Solutions to node classification tasks often rely on the exploitation of both homophily and structural equivalence. We want to exploit these two concepts and model the relationships between nodes, rather than simply treating nodes as independent datapoints.

Graph Classification Given a set of multiple different graphs G, instead of making predictions over the individual nodes and edges for a single graph, the goal is to predict the label y_g, which could be the type, category, etc., associated with the graph $g \in G$. In the graph clustering task is closely related, where the goal is to learn an unsupervised measure of similarity between the graphs in the provided set [61].

2.8.1 Neural Message Passing

As explained in [110], the *Message Passing Neural Network* (MPNN) approach aims to capture the dependence of the graph via vector message exchange between its nodes. This can be done by updating the hidden state h_v of nodes by a vertex update function U_t of the previous hidden state and the message of nodes neighbors given by the message function M_t. For simplicity, we explain only the phases related to the undirected graphs, but the extension to directed multigraphs should be trivial. We are given a graph G with node features $\vec{x}_v$ and edge features $\vec{e}_{vw}$. The approach is split into two phases; a message passing phase and a readout phase.

In the message passing phase, we define a number of timesteps T. For each node during this phase, $\vec{h}_v^t$ denotes the hidden state (distributed vector representation) at timestep t. This state is updated based on messages given:

$$\vec{h}_v^{t+1} = U_t(\vec{h}_v^t, \vec{m}_v^{t+1}) \tag{2.8}$$

where $\vec{m}_v^{t+1}$ is the message at time step $t+1$, given by

$$\vec{m}_v^{t+1} = \sum_{w \in N(v)} M_t(\vec{h}_v^t, \vec{h}_w^t, \vec{e}_{vw}) \tag{2.9}$$

where $N(v)$ denotes the neighbors of node v in G. For each timestep t, the hidden state of each node is updated simultaneously in parallel. After T timesteps have passed, context from nodes at leas T steps away from each node will be embedded in the hidden state $\vec{h}_v^T$.

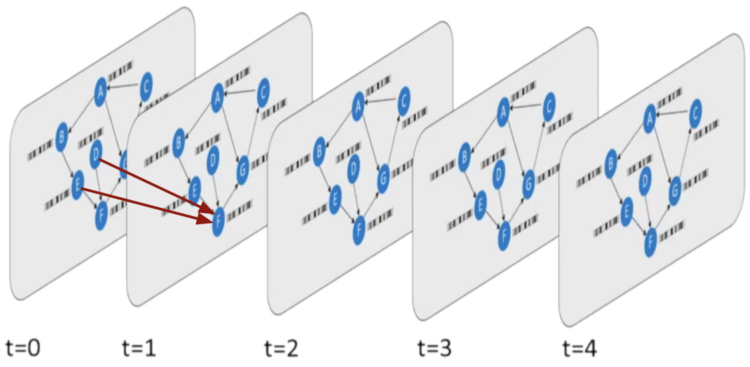


Figure 2.19: Message passing phase

Lastly, in the readout phase, we compute a feature vector for the entire graph using the readout function R:

$$\vec{\hat{y}} = R(\{\vec{h_v^T} | v \in G\}) \tag{2.10}$$

2.9 Attention Mechanism

The *attention mechanism* draws inspiration from the field of psychology, where attention is defined as the cognitive process of selectively concentrating on a discrete aspect of information while ignoring others. In ML, attention enables the modeling of global dependencies between input and output. This is done by not only including surrounding elements as context for the prediction, but introducing their relative importance as well [44]. Among the key benefits of attention mechanisms are the enabling of variable sized inputs. The introduction of attention mechanism has led to widespread success in many sequence-based tasks such as machine translation, image caption generation, video clip description, and speech recognition [44] [111].

2.9.1 Attention in graphs

Though graph-structured data has enabled the extraction of valuable knowledge in different applications through graph mining, there are still factors that make current approaches insufficient. Real-world graphs are often large, include many intricate patterns, and containing lots of noise. An effective way to deal with this issue is to incorporate attention into graph mining solutions. The attention mechanism helps the model make better decisions by only focusing on the graph's task-relevant parts.

Graph Attention Network (GAT) [112] represent a novel convolution-style neural network that operates on graph-structured data, incorporating attention mechanism in the propagation step. It has quickly become the benchmark approach for incorporating attention in Graph Neural Networks. This model utilizes a self-attention strategy, computing the hidden states of each node by attending over its neighboring nodes. In GAT, a *graph attentional layer* is defined, which can be stacked to produce arbitrary Graph Attention Networks. The input to the attentional layer is the F-dimensional hidden state of the N nodes in the graph G,

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

The layer produces a new set of node embeddings as its output. These new vector representations can but do not need to have the same cardinality as the originals. The output is given by;

$$\mathbf{h}' = \{\vec{h'}_1, \vec{h'}_2, ..., \vec{h'}_N\}, \vec{h'}_i \in \mathbb{R}^{F'}$$

In order to transform the input vector representation of nodes to a higher dimensional vector, they apply a shared linear transformation, parametrized by a weight matrix, $\mathbf{W} \in \mathbb{R}^{F' \times F}$, to every node as an initial step. Next, they perform self-attention through a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ on the nodes to compute *attention coefficient* e_{ij} expressing the importance of node j's embedding to node i. This coefficient is computed as;

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \tag{2.11}$$

The graph structure is injected into the mechanism by only computing e_{ij} for nodes $j \in N(i)$, where N(i) denotes the neighbors of node i in G. This is referred to as *masked attention*. Then, coefficients are normalized across all choices of j using the softmax function. In GAT, the attention mechanism a is a single-layer FNN, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, and applying the LeakyReLU nonlinearity (with negative input slope $= 0.2$). The normalized attention coefficients α_{ij} of a node pair (i, j) is computed as;

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i || \mathbf{W}\vec{h}_j]))}{\sum_{k \in N(i)} \exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i || \mathbf{W}\vec{h}_k]))} \tag{2.12}$$

T represents transposition, and || is the concatenation operation. Normalized attention coefficients are then used in a linear combination of the embeddings corresponding to them to serve as the final output h'_i for every node;

$$\vec{h'}_i = \sigma(\sum_{j \in N(i)} \alpha_{ij} \mathbf{W}\vec{h}_j) \tag{2.13}$$

The implementation utilizes *multi-head attention* to stabilize the training process of self-attention. K independent attention mechanisms are employed and either concatenated or averaged to create the final output;

$$\vec{h'}_i = \big\Vert_{k=1}^{K} \sigma\Big(\sum_{j\in N(i)} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\Big) \tag{2.14}$$

$$\vec{h'}_i = \sigma\Big(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\Big) \tag{2.15}$$

The latter equation drops the concatenation in favor of averaging, before the non-linearity. This is used in the final layer of the network, where concatenation is no longer sensible. The aggregation process of a multi-head graph attentional layer is illustrated in Figure 2.20.

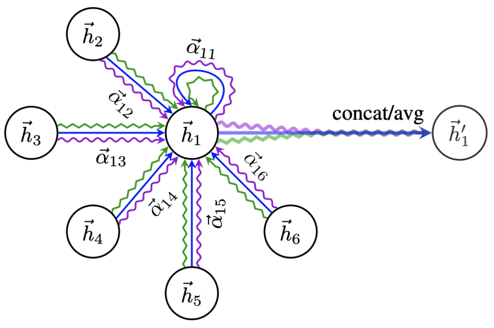


Figure 2.20: Visualization of multi-head attention on node 1. Three different attention mechanisms are employed, depicted with different arrow styles and colors [112]

Chapter 3

Related Work

This chapter presents a selection of previous works akin to the contributions of this thesis. It contains descriptions of some of the most prominent research concerning fake news detection and link prediction through knowledge graph embedding. All models used in the comparisons and evaluation of our contributions are presented in this chapter. Further, section 3.3 presents in detail the attention-based GNN architecture that forms the basis of our proposed model. On the grounds that this thesis proceeds the efforts and findings of our previous work [1], some sections may bear a close resemblance. More precisely, section 3.2 will present the same research as the referred work; in a revised format.

3.1 Fake News Detection

Following the rapid increase in attention to fake news, several approaches to automatic detection of fake news have been proposed. As described in section 2.2, approaches can rely on either: (i) content-based, or (ii) context-based features. Sections 3.1.1 - 3.1.3 will present some of the most common approaches within these. Keep in mind that most approaches use a mixture of content and context features to detect fake news. Section 3.1.3 describes some more uncommon techniques related to the proposed method of this thesis.

3.1.1 Detection based on Content Features

In [28], the authors show that fake and real news articles are notably distinguishable by their textual features. These textual features include complexity features, psychology features, and general stylistic features. How the features differ between the categories of news is found using *one-way ANOVA* and *Wilcoxon rank-sum tests*. This approach successfully separated fake news and satire from real news with an accuracy between 71% and 91%. Conclusively, the authors argue the persuasion in fake news is achieved through heuristics rather than the strength of arguments, which is the case for real news.

Based on a rich set of textual features, [113] propose a language-independent approach for automatic fake news detection. They utilize logistic regression for classification with *L-BFGS optimization* and *elastic net regularization* to distinguishing the fake and credible news. The selected features include *linguistic features* such as; n-gram counts and vocabulary richness, *credibility features* such as; article length, URLs, capitalization metrics and pronoun types, and lastly *semantic features*; modeling of the semantics of the article through averaging the word2vec embeddings of the non-stop word tokens.

HAN [114] propose a hierarchical attention network framework for fake news classification. It is based on content features of the documents and applies two levels of attention mechanisms, one on word-level and one on sentence-level. The model progressively builds a document vector by aggregating important words into sentence vectors and then aggregating important sentence vectors to document vectors.

Rhetorical Structure Theory (RST) [115], is an analysis that captures the coherence of a story in terms of functional relations among different meaningful textual units, and describes a hierarchical structure for each story. It builds a tree structure to represent rhetorical relations among the words in the text. RST is able to extract news style features by mapping frequencies of rhetorical relations to a vector space.

Linguistic Inquiry and Word Count (LIWC) [116] was developed in order to provide an efficient and effective method for studying the various emotional, cognitive, and structural components in textual documents. It has been widely adopted to extract the lexicons falling into psycholinguistic categories. It relies on an internal default dictionary that defines which words should be counted in the target text files. The internal default dictionary is used to learn a feature vector from the psychology and deception perspective.

TextCNN[1] is a simplified implementation of the described use of CNNs for sentence classification tasks from [117]. First, the model is used to classify news contents by capturing different granularity of text features through multiple convolution filters. Next, they max-pool the result of the convolutional layer into a long feature vector, conduct dropout regularization, and classify the result using a softmax layer.

Among more cutting-edge research, *dEFEND* [118] employs an attention-based deep learning approach. The authors propose a sentence-comment co-attention sub-network to jointly capture explainable top-k check-worthy sentences and user comments for fake news detection. In addition to outperforming several state-of-the-art approaches, it outperforms the baselines by 30.7% in precision at success-

[1]`https://github.com/dennybritz/cnn-text-classification-tf`

fully identifying the top-k user comments that explain why a document is fake.

3.1.2 Detection based on Context Features

By taking advantage of “wisdom of crowds”, the authors of [119] improve news verification by mining conflicting viewpoints of users on social media. They leverage a topic model to identify conflicting viewpoints and build a network of these viewpoints linked by supporting and opposing relations. Furthermore, by formulating credibility propagation on their network as a graph optimization problem, they provide an iterative optimal solution that outperformed the baselines.

User-Based

In [120], user opinions are mined from engagement hierarchies in social media. The credibility of users and authenticity of news are calculated using a *Bayesian network model*, which yields a probability of their trustworthiness. These new variables are treated as latent random variables, such that a probabilistic graphical model can be built to capture the complete generative spectrum using a *collapsed Gibbs sampling* approach. It is noteworthy that this whole process is unsupervised and outperforms the unsupervised benchmarks.

Network-Based

The authors of [121] introduce a *network-based pattern-driven model*. They inspect how patterns of fake news spread in social networks, which include both the spreaders of the news and the relationships among them. By integrating empirical studies and social psychological theories, they represent patterns on multiple network levels to detect fake news. Their approach enhances the explainability of fake news feature engineering and outperforms state-of-the-art approaches. However, as opposed to content-based methods, their model can not detect fake news before it has been propagated on a social network.

3.1.3 Computational-Oriented Fact-Checking

Several approaches have been proposed utilizing knowledge-based information retrieval methods to determine the veracity of news articles. As an example, [122] presents a web-based fact-checking framework, using web queries to calculate the trustworthiness of textual documents. This approach starts by extracting key facts, or statements, from the document. Thereon, web search is used to estimate the web support of these statements. Finally, the individual support for each statement is aggregated to produce an overall score indicating the trustworthiness of the document.

An alternative approach to the task of computational-oriented fact-checking was presented in [123]. Here, the authors cast fact-checking as a link prediction task in knowledge graphs. Given a knowledge graph G, and a statement S on the format $\langle subject, predicate, object \rangle$. Checking statement $S = \langle s, p, o \rangle$ is equivalent to predicting the existence of edge $s \xrightarrow{p} o$ in G. The fact-checking is extended to include alternative paths P between entities of the same type as s and o through the knowledge graph G. By extracting discriminative paths from the KG, they validate the truthfulness of any given statement S. This approach yielded state-of-the-art performance on fact-checking while additionally accommodating interpretability by providing sensible reasons for the predictions.

In [15] the task of computational oriented fact-checking is solved using a novel approach combining the construction of knowledge graphs and use of knowledge graph embedding models. First, three KGs are constructed from different article bases containing; verified fake news, news from reliable news agencies, and an independent open KG such as FB15k and DBpedia. Then, to detect whether the news article is true or fake, TransE [105] models are utilized to build entity and relation embedding in low-dimensional vector space for each KG. Next, the authors compare the performance of single TransE models, a binary-TransE model trained on the fake and true news corpora, and lastly, a hybrid approach combining the feature vectors from different models. Results show that external open KGs yield the best performance and that binary models generally outperform single models.

3.2 Link Prediction

There exists a large assortment of approaches and architectures proposed to solve the task of LP. We will focus on a group of approaches based on KG embedding. As previously described in section 2.6, these approaches can generally be defined as either: (i) geometric, (ii) tensor decomposition, (iii) or deep learning based models. The following subsections will present a handful of novel and state-of-the-art models within these categories.

3.2.1 Geometric models

TransE [105], an LP model inspired by the word2vec algorithm, was the first of its kind using a geometric interpretation of the embedding space. It considers the translation operation between head and tail nodes for edges. Given a selected distance function, TransE requires that the tail embedding lies close to the sum of the embedded head and edge. Due to the nature of translation, this model has shown to struggle with one-to-many and many-to-one relations, as well as symmetric and transitive relations. The observed limitation in TransE's ability to satisfy the translational constraint resulted from the regularization technique forcing embeddings to lie on a hypersphere. Due to this limitation, *TorusE* [124] was proposed. The TorusE model works by projecting each point $\vec{x}$ of the traditional open manifold $\mathbb{R}^d$ into a $[\vec{x}]$ point on a torus $\mathbb{T}^d$.

CrossE [125] is among the more recent geometric models proposed. It introduces an additional edge-specific embedding $\mathbf{c}_r$ for each edge in the graph. These embeddings are learned during the training of the model. Triple-specific embeddings are produced by taking element-wise products to combine $\mathbf{c}_r$ with the head and edge embeddings. The triple-specific embeddings are then used in the translation operation.

3.2.2 Tensor Decomposition models

DistMult [126] utilizes a special case of the bilinear objective used in the geometric model TransE. Whereas TransE is based on element-wise subtraction with a bias, DistMult uses weighted element-wise dot product. Given a triple $\langle h, r, t \rangle$, the scoring function of DistMult is defined as the bilinear product:

$$\phi(h,r,t) = \vec{h}^T \cdot \mathbf{R} \cdot \vec{t}$$

where $\vec{h} \in \mathbb{R}^d$ denotes the head embedding, $\vec{t} \in \mathbb{R}^d$ the tail embedding, and $\mathbf{R} \in \mathbb{R}^{d \times d}$ is the relation embedding represented by a bidimensional matrix. The bidimentional matrices representing relations are restricted to diagonal matrices. As this makes the scoring function commutative, it results in the model treating relations as symmetric $\phi(h,r,t) = \phi(t,r,h)$. Despite this weakness, it has been shown that a carefully tuned model can still reach state-of-the-art performance [127]. This weakness was addressed in *ComplEx* [128]. It generalizes DistMult by extending the embeddings in the complex space: $\vec{h} \in \mathbb{C}^d$, $\vec{t} \in \mathbb{C}^d$, $\mathbf{R} \in \mathbb{C}^{d \times d}$, and hence using *Hermitian dot products* to model entity edges. By replacing the embedding $\vec{t}$ by the *Hermitian transposed* $\vec{t}^H$, the approach enables modeling of asymmetric relations.

Another auspicious model based on tensor decomposition is *TuckER* [129]. It has proven a great success for the LP tasks and reached state-of-the-art performance on several benchmark datasets [11]. The model is based on the notion of *Tucker Decomposition*, factorizing tensors into a set of vectors and a smaller shared core $\mathcal{W}$. TuckER allows for flexibility in the dimension size of node and edge embeddings $\vec{v} \in \mathbb{R}^{d_v}$, $\vec{e} \in \mathbb{R}^{d_e}$. The shared core $\mathcal{W} \in \mathbb{R}^{d_v \times d_e \times d_v}$ is learned jointly with the graph's embeddings and is what enables TuckER in modeling the asymmetry of relations.

3.2.3 Deep Learning models

Following the rapid increase in research on deep learning, lots of attention has been put on utilizing it for link prediction. Several types of neural architectures have been employed for the LP task, with the hope that these networks will be able to encapsulate global topology, connectivity, and attributes of a graph's nodes and edges. This subsection will cover approaches within four architectural categories: Convolutional, Capsule, Recurrent, and Graph Neural Networks.

CNN-based

ConvKB [130] employs a Convolutional Neural Network to enable the capturing of global relationships and transitional characteristics of the graph. It composes the embeddings of each triple $\langle h, r, t\rangle$ into a 3-column input matrix $\mathbf{A} = [\vec{h}, \vec{r}, \vec{t}] \in \mathbb{R}^{k\times 3}$, where k denotes the cardinality of the node and edge embeddings. The input matrix is fed to a convolution layer where different feature maps are generated by applying a set $\mathbf{\Omega}$ of filters. Then, a single feature vector $\vec{f}$ representing the input triple is generated by concatenating the feature maps. The feature vector is multiplied with a weight vector $\vec{w}$ via dot product to return a triple score representing its validity.

ConvE [131] is another CNN-based approach, characterized by a large reduction in parameters compared to other models. While yielding the same performance as both DistMult and R-GCN, it has 8x and 17x fewer parameters, respectively. The model predicts links using convolution over 2D shaped embeddings $[\vec{h}; \vec{r}] \in \mathbb{R}^{d_m\times d_n}$. A 2D embedding is let through a convolutional layer with a set ω of $m \times n$ filters, followed by a dense layer with d neurons and a set of weights $\mathbf{W}$. Finally, the score for triple $\langle h, r, t\rangle$ is calculated by combining the output with the tail embedding $\vec{t}$ using the dot product.

CapsNet-based

Recent efforts have been made in employing *Capsule Neural Networks* (CapsNet) to the LP task. These networks seek to omit the fundamental drawbacks of CNN's lack of consideration of important spatial hierarchies between simple and complex objects, as well as the issue of pooling operations leading to the loss of valuable information. This notion was brought to light by Geoffrey Hinton, widely considered one of the founders of deep learning.

> "*The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster*"
>
> — G. Hinton [132]

Hinton was among the authors proposing the *CapsNet* architecture [133]. This architecture incorporates dynamic routing algorithms to estimate features, i.e., position, size, orientation, of specific objects. Capsules are composed of groups of neurons whose activity vector represents the instantiation parameters of a specific type of object or object part. Compared to normal neurons, capsules differ in the way they perform computations on their inputs. Where neurons produce scalar quantities, capsules encode specific input features to produce a vector of highly informative outputs. The length of the output vector of a capsule represents the probability that the specific object it represents is present in the current input, and its orientation represents the instantiation parameters.

CapsE [134] presents a model for LP which, much like ConvKB, represents each triple $\langle h, r, t \rangle$ with k-dimensional embeddings as a 3-column input matrix $\mathbf{A} = [\vec{h}, \vec{r}, \vec{t}\,] \in \mathbb{R}^{k \times 3}$. Each column vector in $\mathbf{A}$ represent the embedding of an element in the triple, where we assume that different embeddings encode homologous aspects in the same positions. The matrix is then fed to a convolution layer where different feature maps $\mathbf{q} = [\vec{q_1}, \vec{q_2}, ..., \vec{q_k}] \in \mathbb{R}^k$ are generated by applying a set of filters $\mathbf{\Omega}$. The feature maps are reconstructed into corresponding capsules that are then routed to another capsule, where a non-linear squashing function is used to produce a vector output $\vec{e}$. This vector's length is used as a score to measure the validity of the triple $\langle h, r, t \rangle$.

RNN-based

Approaches based on Recurrent Neural Networks remove the restriction of processing each triple individually and enables the analysis to include variable-length sequences of triples. Despite RNN's proficiency on various sequential data prediction tasks, they introduce certain limitations when used to model relational paths. Traditional RNN architectures treat each input as the same type of element, which works well for tasks including words and numbers. However, the relational paths in graphs alter between two types of elements, nodes, and edges, making RNNs less effective in capturing semantic information in relational paths. Another issue with RNNs for link prediction is that the model is only passed one of the previous elements at each time step. Hence, in the tail prediction of a triple RNNs are only passed its relation, but not the head.

Recurrent Skipping Networks (RSNs) [135] have been proposed as a solution to RNN's struggles in modeling relational paths. These models refine the RNN architecture by introducing a simple yet effective skipping mechanism. RSNs use a conditional function to update the hidden state. Given a relational path $(\vec{x}_1, \vec{x}_2, ..., \vec{x}_T)$, at any time step t, if the input is a relation, the hidden state is updated re-using the triple head too;

$$\vec{h'}_t = \begin{cases} \vec{h}_t & \vec{x}_t \in \mathcal{E} \\ \mathbf{S}_1 \vec{h}_t + \mathbf{S}_2 \vec{x}_{t-1} & \vec{x}_t \in \mathcal{R} \end{cases},$$

where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of nodes and edges, and $\mathbf{S}_1$, $\mathbf{S}_2$ are the weight matrices whose parameters are shared at different time steps. The output vector and target embedding are multiplied via dot product to compute the score of a triple. During training, biased random walk sampling is used to learn relation paths built from the train triples. In evaluation, RSNs employ an optimized loss function using a type-based *noise contrastive estimation* (NCE).

GNN-based

The use of Graph Neural Networks introduce a promising category of approaches to link prediction. This is due to their inherent ability to model structural relationships of graphs. For further description of the GNN architecture, please refer to section 2.8.

Relational Graph Convolutional Network (R-GCN) [136] presents a GNN-based model for the task of LP. The authors propose a encoder-decoder architecture, as shown in Figure 3.1, employing the R-GCN network as encoder, and DistMult as the decoder. For an explanation of the decoder, please see subsection 3.2.2.

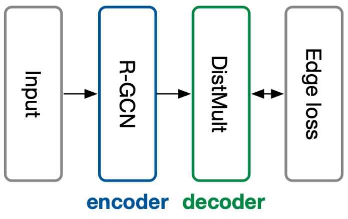


Figure 3.1: LP model with an R-GCN encoder and a DistMult decoder [136]

The architecture is motivated by the MPNN architecture, but introduces relation-specific transformations to accommodate the highly multi-relational data characteristic of real-world knowledge bases. R-GCN define the following model for calculating the forward-pass update of a node v_i in a directed heterogeneous multi-graph:

$$\vec{h}_i^{(l+1)} = \sigma\Big(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}^r(i)} \frac{1}{c_{i,r}} \mathbf{W}_r^{(l)} \vec{h}_j^{(l)} + \mathbf{W}_0^{(l)} \vec{h}_i^{(l)}\Big), \tag{3.1}$$

where $c_{i,r}$ is a problem-specific normalization constant, and $\mathcal{N}^r(i)$ denotes the neighbouring nodes of node i under relation r. A neural network layer update consists of evaluating equation 3.1 for every node in the graph. The update is conducted for each node in parallel. The model introduces basis and block-diagonal decomposition of weight matrices $\mathbf{W}_r^{(l)}$. Basis decomposition accommodates effective weight sharing between different relation types, while block-diagonal decomposition enforces a sparsity constraint on each $\mathbf{W}_r^{(l)}$. Applying these methods result in a large reduction of parameters needed to model multi-graphs with a high number of relation types. Additionally, overfitting on rare relations is alleviated through the shared parameter updates between both rare and more frequent relations.

3.3 GraphStar

GraphStar [137] is a novel and unified Graph Neural Net architecture that utilizes a message-passing relay and attention mechanism for multiple prediction tasks, such as; graph classification, node classification, and link prediction. The model has been shown to achieve non-local representation without increasing the model depth nor bearing high computational costs. According to the authors, GraphStar outperforms several state-of-the-art models by 2% — 5% on multiple key benchmarks. The architecture uses *star nodes* to learn graph-data representation. Inspired by GAT [112], *Non-local Neural Network* [138], and *Star-Transformer* [139], they have managed to capture the global state with fully-connected attention. The model extends the information boundary with the help of relay nodes, which aggregates global information. GraphStar creates an internal data representation via information relay, which is trained to:

1. Perform inductive tasks on previously unseen graph data
2. Aggregate both local and long-range information, making the model globally aware, extracting high-level abstraction typically not represented in individual node features
3. The relays serve as a hierarchical representation for the graphs and can be directly used for graph classification tasks.

The model is described in greater detail as it will be the basis of a modified version used in our experiments. Modifications are presented in section 4.1.

3.3.1 Model Architecture

GraphStar's architecture is divided into three steps. These steps revolve around (i) the initializing star nodes, (ii) updating real nodes, and finally (iii) updating the stars.

The model input is a graph $G(\mathbf{A},\mathbf{F})$, where $\mathbf{A} \in \mathbb{R}^{N_b \times N_b}$ is an adjacency matrix and $\mathbf{F} \in \mathbb{R}^{N_b \times d}$ is a feature matrix, where each node has d features, and N_b being the number of nodes in the graph. Furthermore, the input to each layer of the GraphStar model describe the graph with a set of learned features $\mathbf{H}^t = (\vec{h_1}, \vec{h_2}, ..., \vec{h_{N_b}})$, $\vec{h_i^t} \in \mathbb{R}^{d'}$, where i being the index of the node, and t being the layer number. Note that the dimensionality of d can change between layers, hence d' denotes a *possible* change of dimensionality. The initial representation of graph is a special case where $t = 0$ and $d' = d$, denoted as $\mathbf{F} = (\vec{f_1}, \vec{f_2}, ..., \vec{f_{N_b}})$, $\vec{f_i} \in \mathbb{R}^d$.

Step 1: Initial Representation of the Star

Let $\vec{F}_{mean} = mean(F^0), F \in \mathbb{R}^d$. We have the initial star representation

$$\vec{S^{(0)}} = \sigma(\sum_{i \in N_b} \alpha_{init,i} \mathbf{W}_V^{init} \vec{f}_i) \tag{3.2}$$

where

$$\alpha_{init,i} = \frac{\exp(\langle \mathbf{W}_Q^{init} \vec{F}_{mean}, \mathbf{W}_K^{init} \vec{f}_i \rangle)}{\sum_{k \in N_b} \exp(\langle \mathbf{W}_Q^{init} \vec{F}_{mean}, \mathbf{W}_K^{init} \vec{f}_k \rangle)} \tag{3.3}$$

$\langle \vec{a}, \vec{b} \rangle$ is the dot product of $\vec{a}$, $\vec{b}$,σ is the non-linear activation function and $\mathbf{W}_Q^{init}, \mathbf{W}_K^{init}, \mathbf{W}_V^{init}$ follow the standard Transformer setup [44].

Step 2: Real Node Update

For real node update, the authors use multi-head attention to conduct real node update [112]. Inspired by R-GCN [136], GraphStar defines three internal types of relation which control the importance of nodes under a relation in r with attention coefficients:

1. **Node-to-self**, or self-loop, controlled by $\mathbf{W}_{V0}$.
2. **Node-to-neighborhood** controlled by $\mathbf{W}_{V1}$.
3. **Node-to-star**, or self-star, controlled by $\mathbf{W}_{VS}$.

Adding self-loops, also called *renormalization trick* [136], have been proven to effectively shirk the underlying graph spectrum. Node-to-neighborhood corresponds to the number of relations in the graph, and node-to-star a virtual message-passing relay.

The node update can be represented by

$$\vec{h}_i^{(t+1)} = \overset{N_{Head}}{\underset{m}{\Vert}} \sigma(\sum_{r \in R} \sum_{j \in N_i^r} \alpha_{ijr}^m \mathbf{W}_{rV1}^{m(t)} \vec{h}_j^{(t)} + \alpha_{is,r=s}^m \mathbf{W}_{VS}^{m(t)} S^t + \alpha_{i0,r=0}^m \mathbf{W}_{V0}^{m(t)} \vec{h}_i^t), \tag{3.4}$$

where $\Vert$ represents the concatenation of all multi-heads, N_i^r is the neighbors of node i under relation $r \in R$. N_{Head} is the number of heads in the multi-head attention setting. The multi-head outputs are then concatenated as the representation of $t+1$ layer. Here we define

$$\begin{aligned}\sum^m = \sum_{r \in R} \sum_{j \in N_i^r} & \exp(\langle \mathbf{W}_Q^{m(t)} \vec{h}_i^{(t)}, \mathbf{W}_{rV1}^{m(t)} \vec{h}_k^{(t)} \rangle) \\ & + \exp(\langle \mathbf{W}_Q^{m(t)} \vec{h}_i^{(t)}, \mathbf{W}_{VS}^{m(t)} S^{(t)} \rangle) \\ & + \exp(\langle \mathbf{W}_Q^{m(t)} \vec{h}_i^{(t)}, \mathbf{W}_{V0}^{m(t)} \vec{h}_i^{(t)} \rangle),\end{aligned} \tag{3.5}$$

and the attention coefficients can thus be represented by

$$\alpha_{ijr}^{m} = \frac{\exp(\langle \mathbf{W}_{Q}^{m(t)} \overset{\rightarrow(t)}{h_i}, \mathbf{W}_{rV1}^{m(t)} \overset{\rightarrow(t)}{h_j} \rangle)}{\overset{m}{\sum}}, \tag{3.6}$$

$$\alpha_{is,r=s}^{m} = \frac{\exp(\langle \mathbf{W}_{Q}^{m(t)} \overset{\rightarrow(t)}{h_i}, \mathbf{W}_{VS}^{m(t)} S^{(t)} \rangle)}{\overset{m}{\sum}}, \tag{3.7}$$

$$\alpha_{i0,r=0}^{m} = \frac{\exp(\langle \mathbf{W}_{Q}^{m(t)} \overset{\rightarrow(t)}{h_i}, \mathbf{W}_{V0}^{m(t)} \overset{\rightarrow(t)}{h_i} \rangle)}{\overset{m}{\sum}}. \tag{3.8}$$

It should be noted that for all relations, GraphStar adopts a parameter sharing scheme $\mathbf{W}_{K*} = \mathbf{W}_{V*}$.

Step 3: Star Update

After we have the real node representation at layer $t+1$, one can update the star nodes via

$$S^{(t+1)} = \overset{N_{Head}}{\underset{m}{\|}} \sigma(\sum_{j \in N_b} \alpha_{s,j}^{m} \mathbf{W}_{V}^{m(t)} \overset{\rightarrow(t+1)}{h_j} + \alpha_{s,s}^{m} \mathbf{W}_{V}^{m(t)} S^{t}). \tag{3.9}$$

We define

$$\sum_{s}^{m} = \sum_{k \in N_b} \exp(\langle \mathbf{W}_{Q}^{m} S^{t}, \mathbf{W}_{K}^{m} \overset{\rightarrow(t+1)}{h_k} \rangle) + \exp(\langle \mathbf{W}_{Q}^{m(t)} S^{t}, \mathbf{W}_{K}^{m} S^{t} \rangle), \tag{3.10}$$

and the attention coefficients are thus

$$\alpha_{s,j}^{m} = \frac{\exp(\langle \mathbf{W}_{Q}^{m} S^{(t)}, \mathbf{W}_{K}^{m} \overset{\rightarrow(t+1)}{h_j} \rangle)}{\overset{m}{\sum}}, \tag{3.11}$$

$$\alpha_{s,s}^{m} = \frac{\exp(\langle \mathbf{W}_{Q}^{m} S^{(t)}, \mathbf{W}_{K}^{m} S^{(t)} \rangle)}{\overset{m}{\sum}}. \tag{3.12}$$

Loss Function

The purpose of the training is to find embeddings and shared parameters that maximize the plausibility of true triples and, at the same time, minimize the plausibility of false triples. As the LP datasets contain only positive triples, these are often corrupted to generate a set of presumably false triples during run-time. The corrupted triple set, G_{corr}, is included in the loss function with the goal of minimizing their score. G_{corr} consist of *anti-patterns* of the true triples, i.e., triples that do not exist in the training set. The problem then becomes a binary classification problem where the model uses the loss function, *Binary Cross-Entropy*, defined as

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{3.13}$$

where N is the number of triples including the corrupted triples, y_i is the label for each triple, and $p(y_i)$ is the predicted probability of triple i being true. The predicted probability is calculated with the help of a scoring function.

Scoring Function

In principle, the model can be adapted to rely on any scoring function, the most straightforward being DistMult is used. DistMult is known to perform well on standard link prediction benchmarks when it is used on its own. The drawback is that it can not model asymmetry, which causes a triple $\langle h, r, t \rangle$ to be scored equally as a triple $\langle t, r, h \rangle$. The scoring function is defined as

$$\phi(h, r, t) = \vec{h_h}^{T} \cdot \mathbf{R}_r \cdot \vec{h_t} \tag{3.14}$$

where $\vec{h_i}$ is the embedding of entity i, and $\mathbf{R}_i$ being an embedding of relation i.

3.3.2 Limitations

The work of GraphStar mainly addresses many previous limitations, such as GAT's inability to model deep-neighborhood representation, which lessened the model's capability to understand the context. However, there are still significant limitations to the proposed architecture. The model is based on the special case $R = 1$, making it a single-type relation link prediction. The special case implies that it is not applicable to heterogeneous graphs, i.e., knowledge graphs. Further, DistMult is utilized as the decoder of the network. As a result, GraphStar is not a fully expressive model, treating all relations as symmetrical.

Chapter 4

Method

This chapter will introduce the models and frameworks used to conduct the experiments of the thesis. The objective is to present our method clearly and concisely and to provide enough information so that, together with chapter 5, the results of our thesis can be replicated.

4.1 mrGraphStar

Among the objectives of this thesis was the implementation and evaluation of an attention-based Graph Neural Network for link prediction in knowledge graphs. At the inception of this research, no such architecture had been proposed. Consequently, this implied the extension of one such architecture to incorporate edge features in its predictions. Through a comprehensive literature review, we identified a select number of relevant studies corresponding to our inclusion criteria. The collection of research to base our approach on was based on several factors such as reported performance, implied difficulties in extending, and the existence of open-source code for implementation.

This thesis proposes a new link prediction model named *mrGraphStar*[1] (Multi-relational GraphStar), or simply MrGS. It is based on the architecture of GraphStar, explained in section 3.3. Numerous architectural modifications have been made to the original code base[2] to produce our model. These modifications ultimately enable the incorporation of vector representation of heterogeneous directed relations in the training and predictions of our model. This is done by introducing a new group of attributes to the data. Additionally we have created a procedure supporting a reproducible deterministic train, test, and validation split of the dataset. In essence, whereas GraphStar had only a zero-one relation type between nodes, i.e., *has relation* or *has no relation*, we have introduced a spectrum of relation types through a combination of label encoding and relation embedding.

[1]`https://github.com/oscarvik/multiRelational-GraphStar`

[2]`https://github.com/graph-star-team/graph_star`

Initial node representation

Before sending the adjacency matrix and feature vectors to the model, we embedded the nodes of the graph through node2vec. Each node is given a default representation $(\vec{n_1}, \vec{n_2}, ..., \vec{n_{N_b}})$, $\vec{n_i} \in \mathbb{R}^d$, where N_b is the number of nodes and d is the chosen cardinality of the vector. The now embedded nodes can be indexed using a simple label encoder on the original set of entities to encode the target labels with a value between 0 and $N_b - 1$. The embedded vector is ordered into a matrix by the index, creating a node type lookup table. This results in a matrix $\mathbf{F} \in \mathbb{R}^{N_b \times d}$. The new feature matrix must be passed to the model giving the graph $G(\mathbf{F}, \mathbf{R}, \mathbf{A})$. The choice of providing an initial embedding is motivated by the findings that GNN models with default node2vec embeddings outperform pure node2vec by large margins and that the joint learning often improves the performance of the GNN model as well [140].

Heterogeneous directed edges

Relations from FB15k and FB15k-237 follow a hierarchical structure, as shown in Figure 4.1, where the leftmost word is the most generic and the rightmost, the most precise. For instance, the relation *medicine/drug/legalstatus* shows the relationship between two entities, being in the domain of medicine, more specific drugs, and most specifically, the legal status of this drug. Motivated by the findings of [141], we investigate if this hierarchy can be taken advantage of through word embeddings which embed the meaning of each word to a vector. Their experiments showed that relation embeddings could be utilized to produce better results for baseline decoders such as DistMult.

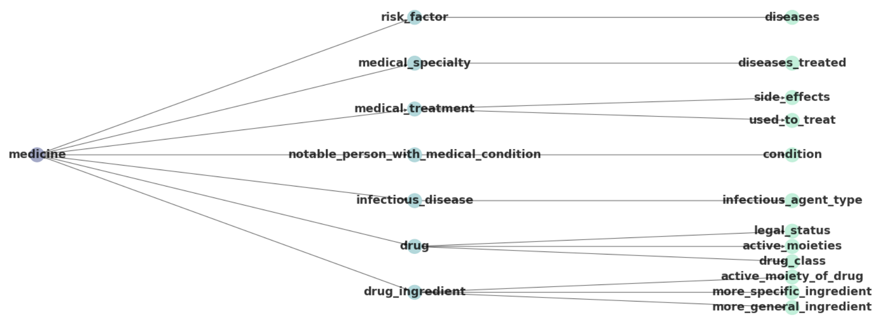


Figure 4.1: Hierarchical structure of the relations in Freebase

Our approach to the relation embeddings is as follows. Let r be a relation in R containing a sequence of words $\{w_1, w_2, ..., w_k\}$ and $\Psi(w)$ be a mapping function from the alphabetic domain to the numerical domain, $\mathbb{R}^{E_b}$, where E_b is the cardinality of the vector. The embedding $\Psi(r)$ then results in the set of vectors $\vec{r} \in \mathbb{R}^{k \times E_b}$. This group of vectors can then make a composite vector that encapsulates the relationship. The most trivial way to support this is by using the average of the vectors as the mapping, resulting in the vector $\vec{r} \in \mathbb{R}^{E_b}$. The now embedded relation can be indexed using a simple label encoder on the original set of relations to encode the target label with a value between 0 and $|R|-1$. The embedded vector is ordered into a matrix by the index, creating an edge type lookup table. Doing this for all relations R results in a matrix $\mathbf{R} \in \mathbb{R}^{R \times E_b}$. The new relation matrix must be passed to the model giving the graph $G(\mathbf{F}, \mathbf{R}, \mathbf{A})$.

Deterministic data split

When introducing heterogeneous graphs, thus multiple relation types for edges, we also introduce a more complex train, validation, and test split of our dataset. Whereas the original model had an absence of types, and a stochastic data split, we introduce the attributes new *train_edge_type*, *val_edge_type*, and *test_edge_type* which contains indexes of the node feature matrix **R**. After the train-val-test split, 3 sets of data are needed for each partition

$$\mathbf{F}_{tr} \in \mathbb{R}^{N_{tr} \times d}, \quad \mathbf{R}_{tr} \in \mathbb{R}^{R_{tr} \times E_b}, \quad \mathbf{A}_{tr} \in \mathbb{R}^{N_{tr} \times N_b} \tag{4.1}$$

$$\mathbf{F}_{va} \in \mathbb{R}^{N_{va} \times d}, \quad \mathbf{R}_{va} \in \mathbb{R}^{R_{va} \times E_b}, \quad \mathbf{A}_{va} \in \mathbb{R}^{N_{va} \times N_b} \tag{4.2}$$

$$\mathbf{F}_{te} \in \mathbb{R}^{N_{te} \times d}, \quad \mathbf{R}_{te} \in \mathbb{R}^{R_{te} \times E_b}, \quad \mathbf{A}_{te} \in \mathbb{R}^{N_{te} \times N_b} \tag{4.3}$$

where N_{tr}, N_{va}, N_{te} are partitions of the total node set, and R_{tr}, R_{va}, R_{te} are partitions of the total set of relations. From this point on, training set $\mathcal{G}_{train}$ will refer to Equation 4.1, validation set $\mathcal{G}_{val}$ will refer to Equation 4.2, and finally, testing set $\mathcal{G}_{test}$ will refer to Equation 4.3.

Loss Function

The loss function is kept from the original model Equation 3.13; however, we corrected a flaw where positive edges, but not negative edges, were symmetrized. As a result, each positive edge appeared with two copies, but each negative edge with only one copy, resulting in a biased performance. The approach chosen in the modified architecture is to symmetrize the positive edges, conduct corrupted triple sampling, then remove the newly created symmetric edges as well as half of the newly created corrupted triples. This process results in the dataset still being directed, having an equal number of real and corrupted triples, and having no corruption of randomly created real triples in the corrupted sampling set. The loss function is now defined as

$$\mathcal{L} = -\frac{1}{N} \sum_{\langle h,r,t \rangle \in \mathcal{G}_{train,corr}} y_{\langle h,r,t \rangle} log(\phi(h,r,t)) + (1 - y_{\langle h,r,t \rangle}) \cdot log(1 - \phi(h,r,t)) \tag{4.4}$$

where $\mathcal{G}_{train,corr}$ denotes the set containing triples from both the original and corrupted training set.

4.2 Fake News Detection

In this section, we present our design of a computational-oriented fact-checking framework for fake news detection. The framework includes the utilization of our proposed link prediction model mrGraphStar and the implementation of an RDF-triple extraction framework. An illustration of the suggested fake news detection framework is presented in Figure 4.2.

4.2.1 The Proposed Framework

Given a corpus of news articles $\mathcal{D}$, each containing a sequence $\{w_1, w_2, ..., w_k\}$ of words and a corresponding label $y \in \{0, 1\}$, where $y = 1$ indicates that the news is fake. We construct an RDF-triple corpus $\mathcal{D}_{RDF}$ by extracting each statement, or clause, from the original corpus. The new dataset now consists of a set of triples $\langle h, r, t\rangle$ with reference to their corresponding article in $\mathcal{D}$. Each triple consists of a source entity h which is the subject in a clause, a target entity t which is an object in a clause, and a predicate r between them.

Given a knowledge graph *G(***F***,* **R***,* **A***)*, where $\mathbf{F} \in \mathbb{R}^{N_b \times d}$ is a feature matrix representing the embeddings of each node, $\mathbf{R} \in \mathbb{R}^{E_b \times r}$ is a feature matrix representing the embeddings of each relation type, and $\mathbf{A} \in \mathbb{R}^{N_b \times N_b}$ is an adjacency matrix denoting the edges between pairs of nodes. We now train a model M on the task of link prediction. We adapt the mrGraphStar network, presented in section 4.1, to embed the nodes of G. For each node, its embedding vector is learned from its neighbors and their corresponding multi-type relations.

Further, the scoring function $\phi(\mathbf{h}, \mathbf{r}, \mathbf{t})$ is used on the learnt embeddings of model M to estimate the plausibility of each triple $\langle h, r, t\rangle \in \mathcal{D}_{RDF}$. Next, to conduct the classification of the news articles in the original corpus $\mathcal{D}$, the scoring algorithm assigns a predicted label $\hat{y} \in \{0, 1\}$ to each triple. A simple strategy for assigning false labels to documents including at least one false triple is selected.

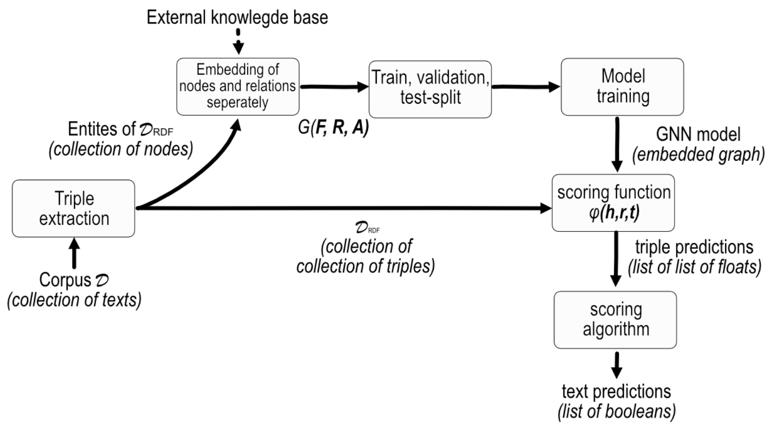

External knowlegde base
Embedding of nodes and relations seperately
Train, validation, test-split
Model training
Entites of $\mathcal{D}_{RDF}$ (collection of nodes)
G(F, R, A)
GNN model (embedded graph)
Triple extraction
scoring function φ(h,r,t)
$\mathcal{D}_{RDF}$ (collection of collection of triples)
triple predictions (list of list of floats)
Corpus $\mathcal{D}$ (collection of texts)
scoring algorithm
text predictions (list of booleans)

Figure 4.2: Flow chart of the proposed framework for fake news detection

4.2.2 Triple Extraction

The architecture of our triple extractor is a pipeline that uses raw news articles as input and produces objects in the form of Freebase RDF triples. The triple extractor is used to convert the original fake news dataset D into the triple extracted dataset $\mathcal{D}_{RDF}$. The pipeline consists of an external Stanford CoreNLP server, an algorithm utilizing NER labels for ranking and labeling, FB15k's relations as a supporting database, web-based lookups on several websites to reduce possible coreferences, a component for mapping Freebase ids to all entities in the Wikidata database, and finally an algorithm removing redundant triples. The pipeline illustrated in Figure 4.3 consists of the following components

Text cleaning	A component removing symbols and stopwords from the raw text, improving the performance of the external extractor.
Stanza CoreNLP	A client producing triples by communicating with a Stanford CoreNLP server.
NER filter	A filter removing triples that do not meet certain criteria.
Relation enhancement	A module parsing the sentence and finds the most likely FB15k relation for the triple.
Google lookup	A component checks the search result of the entity when typed into google. If the search produces an article from a reliable website, the entity is replaced by the string in the article's title.
Wikidata lookup	A module crawls the Wikidata website to find the Freebase ID of the entities in the triple.
Integrity module	An algorithm to verify the integrity of the triples, as well as removing duplicate triples.

For the task of triple extraction, we apply the Stanford CoreNLP server, which provides coreference resolution as well as dependency parsing, and lemmatization, to disambiguate triples. Stanza has integrated the original Stanford CoreNLP codebase to a python library, which initializes a Java server. This server then communicates with the python code through HTTP requests and ultimately produces triples.

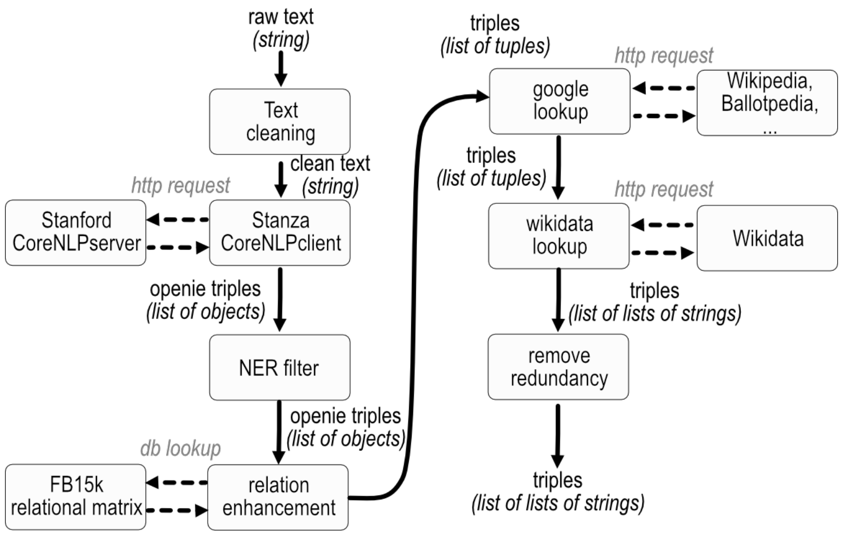


Figure 4.3: Triple extraction pipeline

The triples received from the server can have a varying degree of validity. To affirm fair evaluation, only the triples that give useful information from the text, i.e., could fit in a knowledge graph, should be considered. To support this, we apply a filter of semantic cleaning, where triples are assessed with regards to NER label priority, self-loops, and validity of NER labels. The relation enhancement uses a matrix of strings to find the most probable relation. The algorithm also computes a score for the proposed relation and removes triples not passing a threshold score. The lookup process is used to assure that the entities are atomic, as well as using the most common version of an entity's name. At the end of this process, the entities are sent through a Wikidata lookup function that obtains the entities' freebase id.

We assume that duplicates in the dataset are caused by sentences with redundant information found in the text. Therefore, it would not be wrong to keep these triples since they are real data coincidentally with identical values. However, we choose to look at the extracted triples as a knowledge graph of the text and remove duplicate triples as they do not bring any additional information.

Chapter 5

Experiments

In this chapter, we introduce the overarching strategy, rationale, and experimental setup of our research. Differing from chapter 4, it explains the implementation aspects in detail. The experimental setup includes description of tools, datasets, and hyperparameter selection for the LP model. Additionally, we present the various evaluation metrics selected for measuring the LP model's and fake news detection framework's performance.

5.1 Tools and libraries

Presented below is a list of the most important Python libraries used for the implementation of our model. We include a brief description of what each library was used for.

Pandas[1]	Pandas is an open-source data analysis and manipulation tool. Pandas is used in a wide variety of fields, including academia, finance, economics, statistics, analytics, etc. We have used pandas both in the analysis of the problem, i.e., visualization and rapid prototyping, as well as in the model for data loading and data manipulation.
Scikit-learn[2]	Sklearn is an open-source library for machine learning via a consistence interface in Python. It is built upon the SciPy (Scientific Python) language. Sklearn provides a selection of tools for machine learning and statistical modeling. This includes tools for classification, regression, clustering, and dimensionality reduction. We only utilized Sklearn's metric methods in our implementation.

[1] `https://pandas.pydata.org/`

[2] `https://scikit-learn.org/stable/index.html`

PyTorch[3]	Pytorch is an open-source machine learning framework for Python. This framework provides two main high-level features; tensor computing with strong GPU acceleration and the ability to build deep neural networks on tape-based autograd systems. We used various libraries from Pytorch ranging from storing data structures to computational heavy lifting.
Torch Geometric[4]	Torch Geometric is a geometric deep learning extension library built on top of PyTorch. It is also seen as a framework for Graph Neural Networks comes with a collection of well-implemented GNN models illustrated in various papers. The main benefit of this library is its speed, reporting performance several times faster than the most well-known GNN framework, DGL[5]. We have implemented our model using MessagePassing modules and use several utility libraries from softmax functions to graph self-loop methods.
Stanza[6]	The Stanza library is an NLP a collection of tools for extracting textual content based features in many languages. Most notably, Stanza provides the Stanford CoreNLP client, which supports syntactic analysis and entity recognition, making it easy to convert raw text to machine-interpretable features. We have used the Stanford CoreNLP client in the construction of our triple extracted dataset.
Gensim[7]	Gensim is an open-source Python library providing tools to represent textual documents as semantic vectors. The library gives access to well known unsupervised embedding algorithms such as word2vec. We have used Gensim to create initial vector representations for entities and relations.

[3]https://pytorch.org/

[4]https://pytorch-geometric.readthedocs.io/en/latest/

[5]https://www.dgl.ai/

[6]https://stanfordnlp.github.io/stanza/

[7]https://radimrehurek.com/gensim/

5.2 Datasets

This section presents the characteristics and motivation behind our selected datasets.

5.2.1 Link Prediction

The datasets used in this implementation are FB15k and FB15k-237. They are selected as they have been used extensively in prior work on LP. To this date, FB15k is one of the most commonly used benchmarks for LP [11]. This enables easy and comprehensive comparisons between our model and previous work. FB15k-237 promises a more realistic dataset, where simple observed features model can no longer substantially outperform latent feature models [142]. The characteristics of the chosen datasets are described below.

FB15k is a subset of the Freebase database containing roughly 15k entities, 1,345 different relations, yielding about 592k unique triples. The dataset was built by an open group of volunteers manually creating triples. It follows a schema-based approach using globally unique ids for entities and relations, where the relations are grouped hierarchically, e.g., */medicine/medical_treatment/used_to_treat*. Previous work has noted test leakage flaws in FB15k. This is due to the fact that near-identical and inverse relations from the $\mathcal{G}_{train}$ may be included in the $\mathcal{G}_{test}$.

FB15k-237 [142], a subset of FB15k, was built to omit the current issue of test leakage. This issue resulted in simple models based on observable features being able to reach state-of-the-art performance on FB15k. To build FB15k-237, the authors filtered the set of relations to keep only one of a set of inverse or duplicate relations, resulting in 237 relations. The training, validation, and testing sets were then limited to the set triples with these relations. Additionally, they ensured that none of the entities connected in training set $\mathcal{G}_{train}$ were also directly linked in $\mathcal{G}_{val}$ and $\mathcal{G}_{test}$.

Table 5.1: General properties of LP datasets used in our implementation

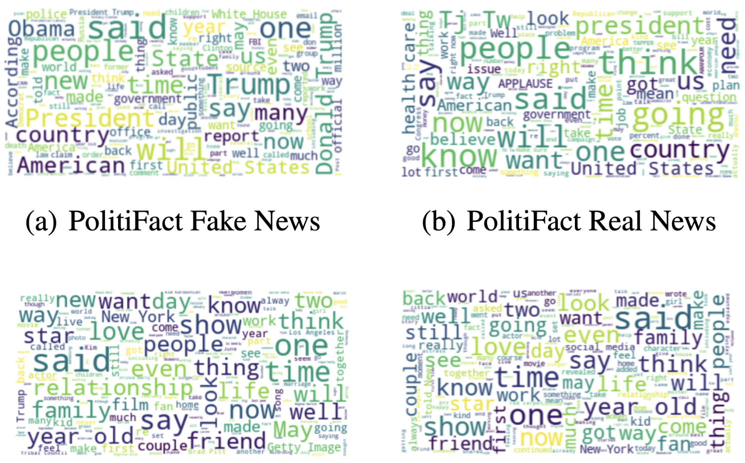
Dataset	**FB15k**	**FB15k-237**
Entities	14951	14541
Relations	1345	237
Train Triples	483142	272115
Val Triples	50000	17535
Test Triples	59071	20466
Test Leakage	Yes	No

5.2.2 Fake News Detection

The fake news dataset used in this implementation is the FakeNewsNet [143]. This dataset was constructed as a result of lacking qualities in other well-researched datasets, such as BuzzFeedNews[8], LIAR[9], and CREDBANK[10]. It has been selected due to having reliable labels annotated by journalists and domain experts with extensive multi-dimension information from news content, social context, and spatiotemporal information, enabling a variety of approaches to the fake news detection task. The dataset has been utilized as the basis for constructing both an automatically and manually annotated triple extracted version.

FakeNewsNet

FakeNewsNet [143] is a comprehensive dataset containing information from news articles, social context, and spatiotemporal information. The news articles are sourced from fact-checking websites such as *PolitiFact*[11] and *GossipCop*[12] and are labeled as either true or fake by journalists and domain experts. Since we have a purely content-based approach to fake news detection, we disregard the social context and spatiotemporal information. A known issue with the FakeNewsNet dataset is that it is highly imbalanced. Of the documents included, 75% belong to the " real" class, while only 25% belong to the "fake" class.



(a) PolitiFact Fake News (b) PolitiFact Real News

(c) GossipCop Fake News (d) GossipCop Real News

Figure 5.1: Textual content word-cloud for fake and real news from FakeNewsNet [143]

[8]https://github.com/BuzzFeedNews/2016-10-facebook-fact-check/tree/master/data

[9]https://www.aclweb.org/anthology/P17-2067/

[10]http://compsocial.github.io/CREDBANK-data/

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

[12]https://www.gossipcop.com/

Figure 5.1 presents an analysis of the topic distribution in fake and real news articles between the two sources. We observe from subfigures (a) and (b) that the fake and real news of the PolitiFact dataset is mostly related to the political campaign. From subfigures (c) and (d), the GossipCop dataset, we observe that the articles are mostly related to topics about relationships among celebrities. In Figure 5.2, one can see that the corpus, as whole, is heavily dominated by American politics. The most salient[13] one is Donald Trump, which is followed up by the 2016 presidential election contender Hillary Clinton.

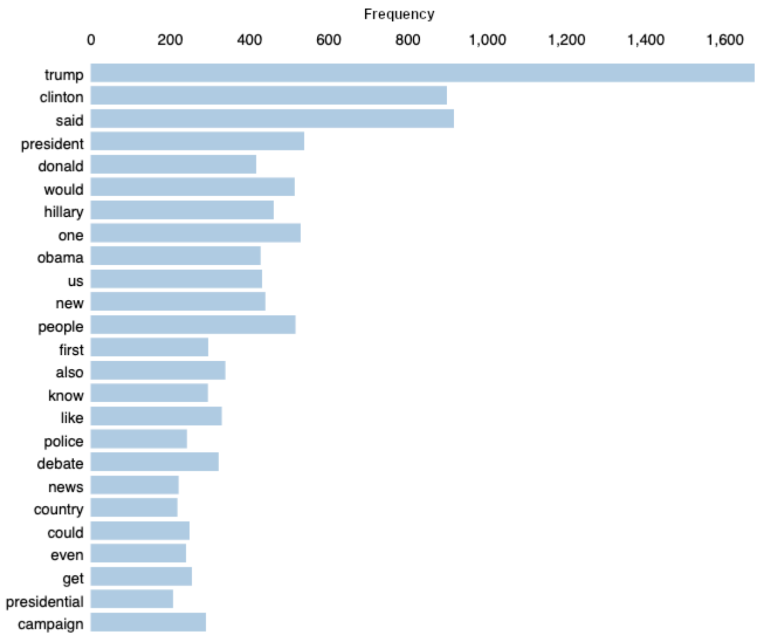


Figure 5.2: Top 25 most salient terms from FakeNewsNet

Triple Extracted FakeNewsNet

We produce both automatic and manually triple extracted versions of a subset of the FakeNewsNet dataset. This is done due to the time consumption of the triple extraction pipeline. To deal with the original imbalance issue, we construct triples from 200 real and 200 fake documents. From here on, the automatically generated dataset is referred to as FNN_{Auto}, while the manually constructed dataset FNN_{Manual}.

[13]saliency(term w) = frequency(w) * $[sum_t p(t|w) * log(p(t|w)/p(t))]$ for topics t

The construction of FNN_{Auto} follows an iterative process, with the following steps:

1. **Clean text** Component removing symbols not fitting into Latin-1[14], as well as removing non-alphanumerical characters.
2. **Stanford CoreNLP** The *CoreNLPClient* was configured to extract triples only if they consume the entire fragment. Thus ensuring that only logically warranted triples are extracted. It was also configured to resolve co-references, use 8GB of RAM, and time out at 1.5 minutes of run time.
3. **NER tag validation** By using a blacklist of NER tags that we are not interested in, such as *DATE* or *PERCENT*, we render triples as "invalid" if either the subject or object is entirely made up of tags in this blacklist.
4. **FB15K relation extraction** Using the FB15K relations as a source of relations, we use both the ner tags and the textual value of the subject, relation, and object to find the closest relation from the source. The relation with the highest similarity score, and surpassing a threshold, is selected as a new relation.
5. **Google lookup** By leveraging the fact that we now mostly have nouns as subject and object and that the dataset is heavily political, we do a google search or the textual value to see if we can find a Wikipedia or Ballotpedia article. If found, we cut the name from the article, and else we keep the former value.
6. **Wikidata lookup** In a similar fashion to the previous step, we send requests to Wikidata and scrape the Freebase id of the entities. If not found this, we insert a nil value and leave it to manual inspection.
7. **Manual inspection** To affirm that the dataset is fair and not lacking and values, we remove data that is wrong and fill in freebase ids (if possible).

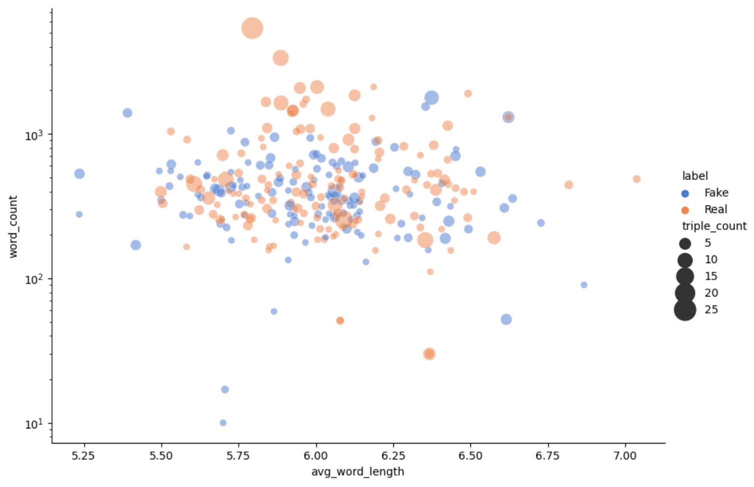


Figure 5.3: Content metrics of FakeNewsNet

[14]https://www.iso.org/standard/28245.html

Automatic vs Manual Triple Extraction

When inspecting the automatically extracted triples, a pattern emerges. Typically, a triple extracted contains a person and some organization, in which the relation between them is either a title or some position. From Figure 5.4, we can see that roughly 92 percent of the total relation set consists of two relation types, namely */people/.../employment_tenure/title* and */organization/.../leadership/person*. This is an indication that the triple extraction pipeline struggles with more complex relations and only extracts obvious ones. An ideal dataset would have a more even distribution of relations and relations that could be more useful in a fact-checking setting.

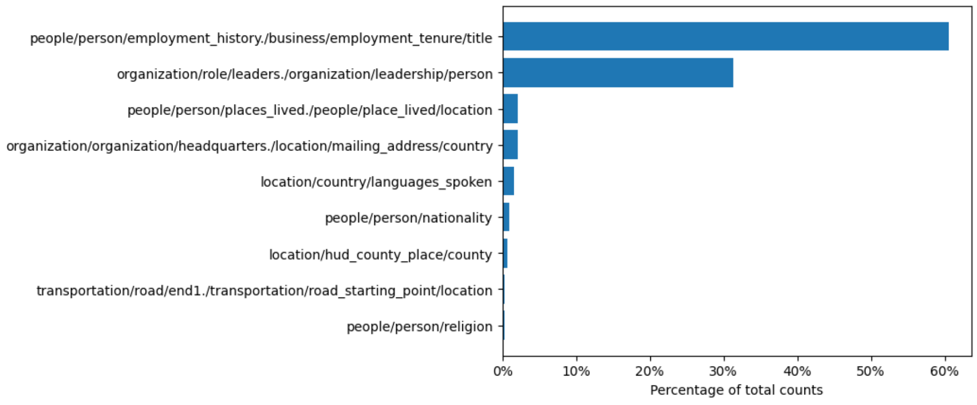


Figure 5.4: Distribution of relations in FNN_{Auto}

The manually annotated dataset FNN_{Manual} aims to contain triples that have a wide range of entities and relations and a high triple count per document. The dataset is created from the same source as the automatically extracted dataset and the motivation that it should be as close to an automatic triple extraction process as possible. The implication of this is that the raw text of the document is the only material eligible for entities and relations, i.e., no reading between the lines. It is also assumed that satirical texts should be interpreted in a literally sense. Manual co-reference resolution is not bound by the content of the text, e.g., *GOP* can be replaced with *The Republican Party*. Due to the time constraint of this thesis, the manually created dataset is significantly smaller than the automatically extracted dataset. Table 5.2 demonstrates the differences of the two datasets.

Table 5.2: General properties of automatically extracted triples vs manual

Dataset	FNN_{Auto}	FNN_{Manual}
Unique Entities	706	114
Unique Relations	9	60
No. Articles	280	20
Real Articles	149 (53.21%)	10 (50%)
Fake Articles	131 (46.79%)	10 (50%)
No. Triples	850	112
From Real Articles	519 (61.06%)	53 (47.32%)
From Fake Articles	331 (38.94%)	59 (52.68%)
Per article	3.04	5.6

The manually annotated dataset has substantially more even distribution of relations types with 36 relations in their top 92 percent, Figure 5.5, whereas the automatic only had two. The top two relations of the new dataset covers 21 percent of all relations, which is still somewhat high considering each relations should ideally be present in 3 percent of the triples, however this is a great improvement on the prior dataset.

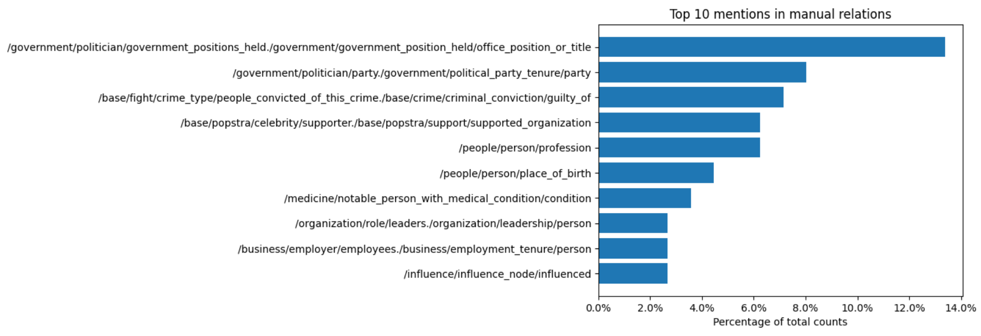


Figure 5.5: Distribution of relations in FNN_{Manual}

5.2.3 Link Prediction vs Fake News Detection

Due to the nature of this task, different datasets are used in training the link prediction model and in evaluating its performance in fake news detection. FB15k, chosen as the source of knowledge for our model, was created in 2013 [105]. On the other hand, FakeNewsNet is from 2018, with most of the articles being from 2016. While there is a good amount of overlapping domain knowledge, there are obvious limitations with this approach. Hence, we present a brief analysis of the overlap, or lack thereof, between these datasets. Although most entities extracted through the triple extraction framework are given a Freebase id, these entities may not be present in FB15k. Entities present in FB15k will from here on be referred to as *in-domain entities*, while the entities extracted from FakeNewsNet not present are referred to as *out-of-domain entities*.

As shown in Table 5.3, our model has only previously observed 20.52% of the entities in FNN_{Auto}, and 41.12% of entities in FNN_{Manual}. We will continue to investigate the effect of this discrepancy on the model's predictive abilities.

Table 5.3: Comparison and overlap of entities in the employed datasets

Dataset	Entities
$FB15k$	14951
FNN_{Auto}	682
FNN_{Manual}	107
$FB15k \cap FNN_{Auto}$	140
$FB15k \cap FNN_{Manual}$	44

5.3 Hyperparameter tuning

Hyperparameter tuning is very important in link prediction models [127], and is often needed for each individual dataset. This section will present the approach for hyperparameter tuning of the modified model through a grid search on *hidden layer count*, *number of neurons* per layer, *learning rate*, and *number of epochs*. The grid search is somewhat restrained as the model needs to train on a fairly large dataset, with shared resources on the Idun cluster. This causes the grid search to be limited in spectrum, and conducted in a greedy manner. The following values is showing in Table 5.4.

Table 5.4: Spectrum of grid search

Parameter	Values
Number of hidden layers	[1, 2, 3, 4, 5]
Number of neurons	[64, 128, 256, 512, 1024]
Epochs	[100, 200, 400, 800]
Learning rate	[0.0001, 0.001, 0.01, 0.1]

The grid search is conducted in an iterative fashion, where the first attributes are the hidden layers and corresponding neurons. The other attributes are set as default from the original model. The second iteration uses the permutation of epochs and learning rate, as well as the best optimum hidden size and neurons combination from the first iteration.

Neurons and Layers

As shown in Table 5.4, the model had five different configurations for number of neurons, and five configurations for number of layers. Hidden layer size dictates the depth of the GNN, while number of neurons dictate the width. Each play the part of memorization vs generalization. Aside from the fact that width can cause overfitting, increasing the width also means longer training time. To find the optimum configuration all permutations are trained for 100 epochs, with otherwise default settings.

Epochs and Learning Rate

Iteration two is done in the same fashion as iteration one, with four different configurations for learning rate, and four epoch checkpoints during training. Learning rate is a parameter for how much the embeddings should be updated each pass, and epochs is a parameter for how many passes the training process should do. The optimum parameters from the previous step is used, otherwise default settings.

5.3.1 Link Prediction Preprocessing

In order to train our LP model on the selected datasets, several preprocessing steps are required. As both datasets are subsets of Freebase, with the same schema, their preprocessing steps are nearly identical. The main task is to create vector representations for all nodes and relations. This is done by first label encoding the unique ids of nodes and relations, as well as creating an updated edge-index matrix. This is used to create a Torch Geometric data-object, which is then converted into a *networkx* graph. The following step consists of embedding the nodes and relations of the graph using the node2vec and word2vec algorithm respectively.

Node embedding

All nodes, with triples from $\mathcal{G}_{train}$ and $\mathcal{G}_{val}$ where embedded using node2vec, as described in section 2.6.2. Table 5.5 lists the selected parameters used in the implementation.

Table 5.5: Parameter selection of node2vec implementation

Parameter	Value
Embedding dimension	$e \in \mathbb{R}^{64}$
Random walk length	$l = 20$
Number of walks	$r = 50$
Return hyper parameter	$p = 1$
In-out parameter	$q = 1$
Context size	$k = 10$

Relation embedding

All valid relations present in the FB15k dataset where embedded using the word2vec algorithm. Each word in the hierarchical structure of a relation is embedded, and then averaged to create the embedding for the relation as a whole, as described in section 4.1. We have chosen embedding dimension $\vec{r} \in \mathbb{R}^{64}$ and set the minimum occurrence of words to 1, meaning we do not ignore any word in the relations.

5.4 Evaluation Metrics

This section presents the metrics used to evaluate the performance of our link prediction model and fake news detection framework. These metrics are selected as they provide various valuable insights into the strengths and weaknesses of the model. Additionally, the state-of-the-art models selected to provide a basis of comparison all provide these metrics in evaluation.

5.4.1 Link Prediction

Evaluation of the LP model is done by performing head and tail predictions on all triples in the test set $\mathcal{G}_{test}$. The model can be evaluated using multiple different metrics, some based on ranking while others on the binary task of deciding whether there should exist a link or not.

LP classification metrics

During training, LP performance is reported as the combined average of *Area Under the ROC Curve* and *Average Precision* scores. These metrics report on the simple task of determining whether there should be a link between two nodes in the graph or not. We define the metrics as:

AP Average Precision (AP) summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. The metric is computed as

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are the precision and recall at the nth threshold.

AUC Area under the ROC Curve (AUC) measures the entire two-dimensional area underneath the entire *ROC curve*. The ROC curve plots *true positive rate* vs. *false positive rate* at different classification thresholds. The AUC metric provides an aggregate measure of performance across all possible classification thresholds.

Rank Metrics

In rank metrics, for each prediction, we compute how the target entity ranks against all other entities. The goal is that the actual target entity yields the highest score. We differentiate between *raw ranks* and *filtered ranks* when considering rank computations [11]. These types differ in the way they view predictions that outscore the target entity. This is a consequence of one simple observation; a triple prediction may have several valid answers. A triple is considered valid as long as it is observed in $\mathcal{G} = \mathcal{G}_{train} \cup \mathcal{G}_{valid}$. Given that our model tries to predict the tail of $\langle The\ Beatles,\ Member,\ John\ Lennon \rangle$, it may associate a higher score to $Paul\ McCartney$ than $John\ Lennon$. The two different scenarios of rank computations are defined as:

Raw scenario In the raw scenario, we regard predicted entities not equal to the target as mistakes regardless if they are valid or not. The raw rank r_t of the target t in a tail prediction of triple $\langle h, r, t \rangle \in \mathcal{G}_{test}$ is given by

$$r_t = |\{e \in \mathcal{E}\ \{t\} : \phi(h, r, e) > \phi(h, r, t)\}| + 1$$

Filtered Scenario In the filtered scenario, prediction of valid entities outscoring the target one are not considered as mistakes. Thus, they are skipped when computing the rank. The filtered rank r_t of the target t in a tail prediction of triple $\langle h, r, t \rangle \in \mathcal{G}_{test}$ is given by

$$r_t = |\{e \in \mathcal{E}\ \{t\} : \phi(h, r, e) > \phi(h, r, t) \land \langle h, r, e \rangle \notin \mathcal{G}\}| + 1$$

We apply the filtered rank scenario in our implementation. To handle a prediction tie, i.e., when two or more predictions are associated with the same score, we use the *min-policy*. In this policy, the target is given the best (lowest) rank among the entities in the tie. We can use the list of ranks Q obtained from prediction to calculate our model's standard global metrics. Below, we describe the most common global rank-based metrics used for LP models.

MR *Mean Rank* (MR) denotes the arithmetic average of the positions of elements in the list of ranks Q. The metric is computed as:

$$MR = \frac{1}{|Q|} \sum_{q \in Q} q$$

MRR The *reciprocal rank* is the multiplicative inverse of the obtained ranks. The *mean reciprocal rank* (MRR) is the average of the reciprocal ranks in Q. The metric is computed as:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q}$$

Hits@K *Hits@K* reports the rate of correctly predicted entities in the top K entries for each instance list. Common values for K are 1, 3, 5, 10. This metric may exceed $Hits@K = 1.0$ if the average K-truncated list contains more than one true entity. The metric is computed as:

$$Hits@K = \frac{|\{q \in Q : q \leq K\}|}{|Q|}$$

5.4.2 Fake News Detection

As we characterize the task of fake news detection as a classification problem, where the aim is to associate the labels fake or true to a document, we introduce metrics common to the classification task. For any given class c_i, we define four categories of predictions. *True positives* (tp) is the group of correctly classified predictions as belonging to class c_i. *False positives* (fp) are the examples that are wrongly classified as belonging to the class c_i. *True negatives* (tn) are predictions that correctly classified as not belonging to c_i. Lastly, *false negatives* (fn) are the group of predictions classified as not belonging to the c_i when they actually do.

Classification models use a variety of metrics in evaluation, and different metrics generally give a slightly different insight of the model's performance. For classification of fake news, we have chosen to report on the four most common metrics in classification tasks; *accuracy*, *precision*, *recall*, and F_1 *score*.

Accuracy

Accuracy is the fraction of total correct predictions to the sum of all predictions. This metric is known to be misleading if used with imbalanced datasets. The metric is computed as:

$$A(c_i) = \frac{tp + tn}{tp + fp + tn + fn}$$

Precision

Precision is a measure of the proportion of true positives against the sum of true positives and predicted false positives. This metric is considered particularly useful when the cost of false positives is high. The metric is computed as:

$$P(c_i) = \frac{tp}{tp + fp}$$

Recall

The recall metric, also known as true positive rate, calculates the amount of actual examples in c_i our model predict as being in c_i. The metric is computed as:

$$R(c_i) = \frac{tp}{tp + fn}$$

F1 score

The F_β score combines recall and precision into a single metric. It is commonly used as it provides better metrics of incorrectly classified classes than the accuracy metric. The general case of this metric is computed as:

$$F_\beta = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R}$$

where β controls the balance between precision and recall in their impact on the metric. We commonly select $\beta = 1$ yielding the F_1 metric. This measure is termed as a harmonic mean of precision and recall, given by:

$$F_1 = \frac{2 \times P \times R}{P + R}$$

Chapter 6

Results

This chapter will present the results gathered from the experiments preformed. The dataset, experiments and evaluation metrics used are detailed in chapter 5.

6.1 Link Prediction

In Table 6.1, we evaluate mrGraphStar on the test set of FB15k and FB15k-237 and compare the results with other common LP models. If not explicitly cited, the evaluation metrics are taken from [11]. We provide results using three commonly used evaluation metrics: *MR*, *MRR*, and $Hits@K$. Plots for training and validation performance is reported using the combined average of AUC and AP scores.

Table 6.1: Best experimental results on FB15k and FB15k-237 test set

	FB15k				**FB15k − 237**			
			$Hits@K$				$Hits@K$	
	MR	MRR	@1	@10	MR	MRR	@1	@10
TransE	45	0.628	0.494	0.847	209	0.310	0.217	0.497
TorusE	143	0.746	0.689	0.839	211	0.281	0.196	0.447
CrossE	136	0.702	0.601	0.862	227	0.298	0.212	0.471
DistMult	173	0.784	0.736	0.863	199	0.313	0.224	0.490
ComplEx	**34**	**0.848**	**0.816**	**0.905**	202	0.349	0.257	0.529
TuckER	39	0.788	0.729	0.889	**162**	**0.352**	0.259	**0.536**
ConvE	51	0.688	0.595	0.849	281	0.305	0.219	0.476
ConvKB	324	0.211	0.114	0.408	309	0.230	0.139	0.415
CapsE	610	0.087	0.019	0.219	405	0.160	0.073	0.356
RSN	51	0.777	0.723	0.870	248	0.280	0.198	0.444
R-GCN [136]	*	0.651	0.541	0.825	*	0.248	0.153	0.414
MrGS	245	0.787	0.773	0.811	214	0.345	**0.329**	0.381

Graphstar Comparison

Unfortunately, investigating the relationships of the performance of our model and Graphstar is challenging. This is due to the inherent difference between the task of predicting zero-one relation types, versus a spectrum of different relation types through label encoding. We can however present the same metrics as in the original paper. Table 6.2 shows the combined average AUP and AP scores on $\mathcal{G}_{test}$.

Table 6.2: Combined average of AUC and AP scores. $\overline{CPC}$ denotes the average performance on the three datasets; Cora, Pubmed, and Citseer [137]

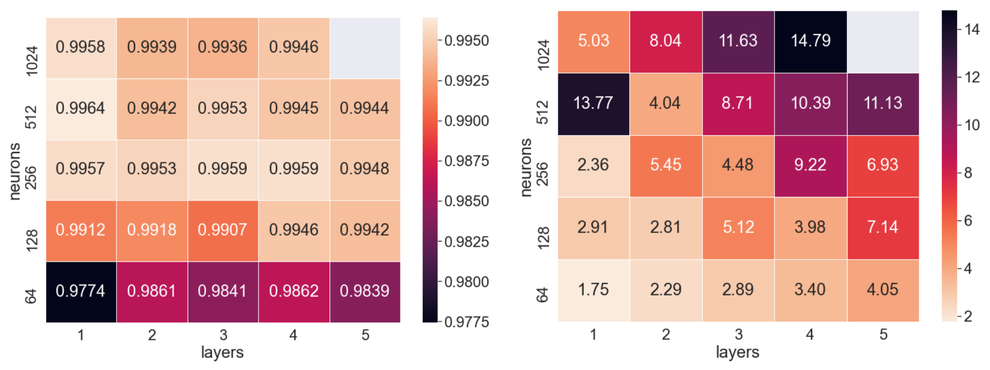
	$\overline{\textbf{CPC}}$	**FB15k**	**FB15k-237**
GraphStar	0.969		
MrGS		0.996	0.997

6.1.1 Hyperparameter tuning

This section will present the results of the hyperparameter tuning presented in section 5.3. The grid search is split into two sections, before the final settings are presented.

Number of Neurons and Layers

The performance of all permutations of iteration one on the link prediction task can be seen in Figure 6.1a and their corresponding runtimes in Figure 6.1b. The hardware used limits the size of the model to have an upper ceiling of 4 layers with 1024 neurons each.



(a) Performance of test set in average precision

(b) Runtime in hours of training

Figure 6.1: Heatmaps of the performance and runtime by neurons and layers

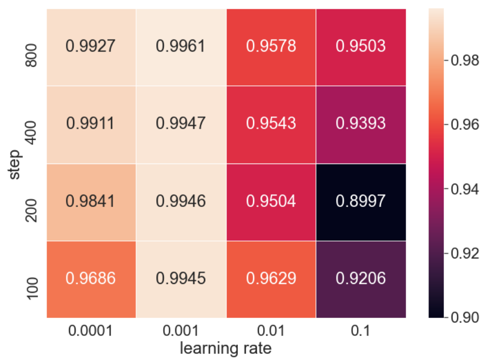
The performance of the link prediction task should be a good indicator of which model to choose, however, for fake news detection other metrics play an important part. In Table 6.3 the model achieving the best rank metrics has 512 neurons and 4 layers.

Table 6.3: Metrics of grid search iteration 1.
The best scores are in **bold** and the second best scores are <u>underlined</u>

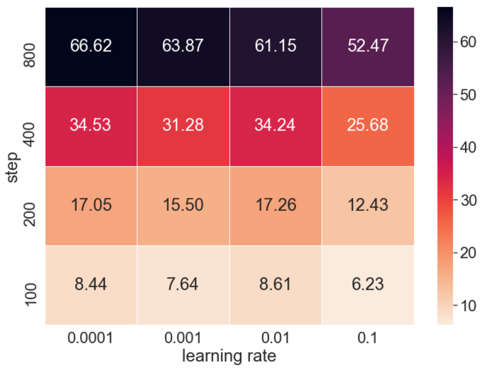
Neurons	Layers	Time *(hh:mm:ss)*	MRR	Hits@1	Hits@10
64	1	01:45:16	0.197	0.172	0.238
	2	02:17:35	0.274	0.250	0.313
	3	02:53:06	0.161	0.130	0.210
	4	03:23:48	0.174	0.149	0.214
	5	04:02:44	0.172	0.144	0.219
128	1	02:54:46	0.377	0.352	0.416
	2	02:48:34	0.343	0.316	0.389
	3	05:07:09	0.357	0.334	0.394
	4	03:59:00	0.339	0.308	0.393
	5	07:08:38	0.395	0.369	0.439
256	1	02:21:47	0.543	0.523	0.576
	2	05:26:51	0.366	0.339	0.413
	3	04:28:57	0.401	0.374	0.448
	4	09:13:13	0.456	0.430	0.501
	5	06:55:42	0.401	0.380	0.435
512	1	13:46:18	<u>0.624</u>	<u>0.607</u>	<u>0.651</u>
	2	04:02:09	0.554	0.539	0.581
	3	08:42:41	0.557	0.544	0.580
	4	10:23:08	**0.787**	**0.773**	**0.811**
	5	11:07:59	0.522	0.505	0.549
1024	1	05:01:33	0.604	0.589	0.628
	2	08:02:24	0.428	0.408	0.464
	3	11:37:57	0.326	0.302	0.366
	4	14:47:18	0.537	0.522	0.560

Epochs and Learning rate

The performance of all permutations of iteration one on the link prediction task can be seen in Figure 6.2a and their corresponding runtimes in Figure 6.2b. The upper bound of epochs is set to 800.



(a) Performance of test set in average precision



(b) Runtime in hours of training

Figure 6.2: Heatmaps of the performance and runtime by epochs and learning rate

In Table 6.3 the model achieving the best rank metrics has 0.0001 learning rate at the 100th epoch mark.

Table 6.4: Metrics of grid search iteration 2

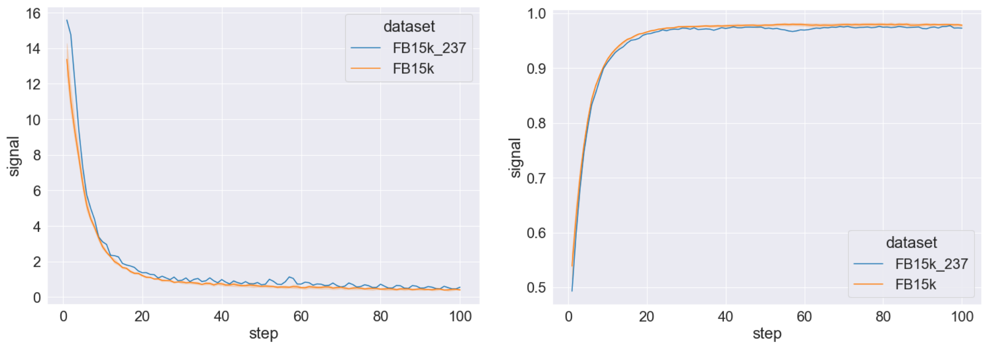
Epochs	Learning rate	Time *(hh:mm:ss)*	MRR	HIT@1	HIT@10
100	0.0001	07:51:39	**0.787**	**0.773**	**0.811**
	0.0010	06:19:01	0.771	0.763	0.784
	0.0100	08:23:36	0.077	0.037	0.146
	0.1000	09:23:13	0.061	0.053	0.077
200	0.0001	17:05:42	0.779	0.760	0.802
	0.0010	16:02:29	0.753	0.745	0.767
	0.0100	17:49:56	0.085	0.046	0.156
	0.1000	15:08:00	0.054	0.045	0.072
400	0.0001	28:59:08	0.725	0.701	0.742
	0.0010	25:02:53	0.708	0.699	0.723
	0.0100	24:55:50	0.096	0.061	0.161
	0.1000	28:05:29	0.126	0.124	0.128
800	0.0001	58:55:44	0.712	0.697	0.728
	0.0010	54:14:02	0.691	0.679	0.708
	0.0100	59:33:40	0.083	0.051	0.139
	0.1000	52:35:49	0.053	0.044	0.068

Model Parameters

The final settings of the best performing model were applied on both link prediction datasets. The number of attention head $N_{head} = 4$, the number of layers is 4, the initial learning rate is 0.0001, and the L2 regularization setting is 0.0005. The attention coefficient dropout is 0, and the hidden layer dropout rate is 0. The number of neurons per layer is 512, and finally the number of epochs is set to 100.

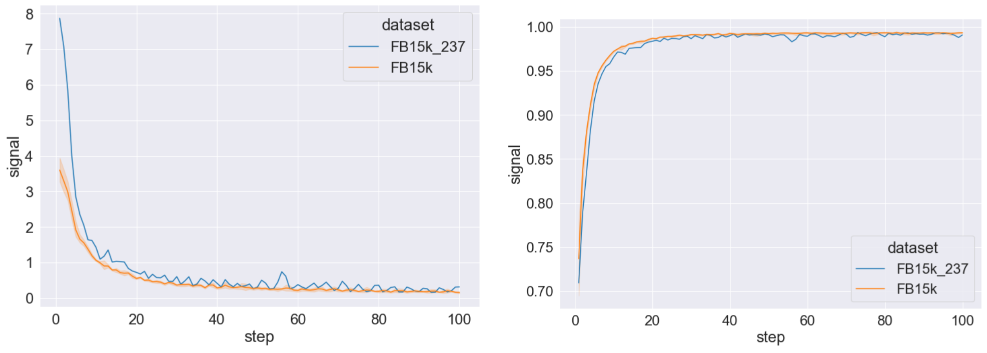
6.1.2 Training

In Figure 6.3, Figure 6.4, we report on the optimization and performance learning curves on both the training set $\mathcal{G}_{train}$ and test set $\mathcal{G}_{test}$.



(a) Loss during training **(b)** Average precision during training

Figure 6.3: Train loss and average precision curves over 100 epochs



(a) Loss during testing **(b)** Average precision during testing

Figure 6.4: Test loss and average precision curves over 100 epochs

The calculation of rank metrics is much more computationally costly than LP classification metrics, therefore we inspected a single run where rank metrcis where computed every 10 epoch.

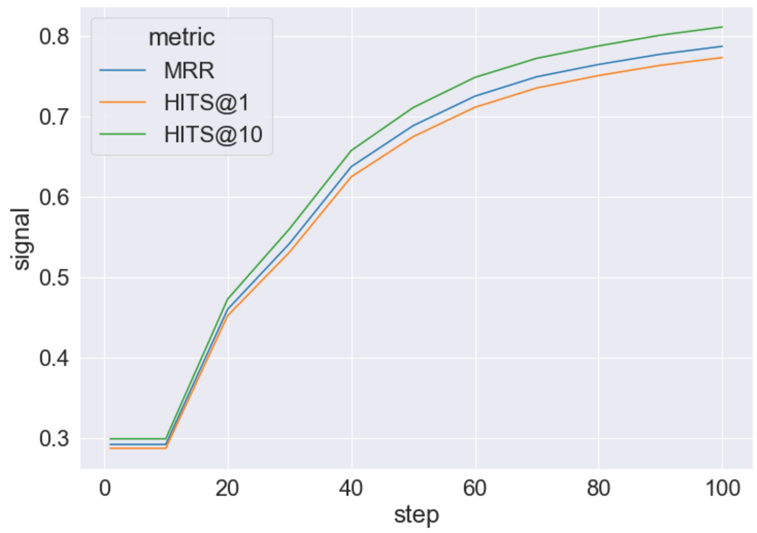


Figure 6.5: Rank metrics computed in 10 epoch interval with optimum settings

6.1.3 Loss Function

We also analyze the effect of our proposed way of generating corrupted triples for models using scoring functions only supporting symmetric triples. Figure 6.6 presents a comparison of the original and newly proposed modification.

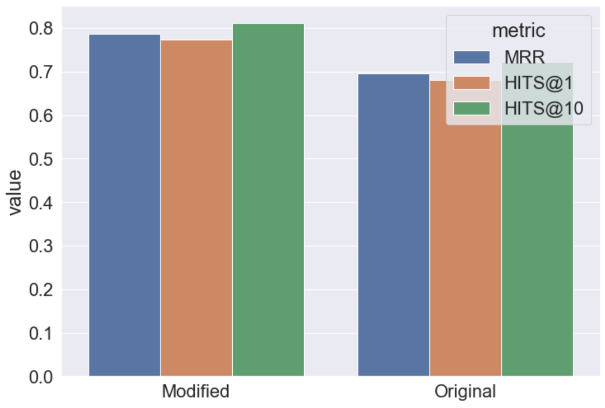


Figure 6.6: Effect of corrupted triple generation on LP performance

6.1.4 Preprocessing

To illustrate the result of our preprocessing of nodes, Table 6.5 present the most similar nodes to a specified node based on their embeddings. We validate the efficiency of the node-embedding by analyzing cosine similarity.

Table 6.5: node2vec; cosine similarity predictions to: *Blood Diamond*

Nodes		Cosine Similarity
1.	*Collateral*	0.755
2.	*The Last Samurai*	0.728
3.	*Rules of Engagement*	0.690
4.	*A Simple Plan*	0.668
5.	*J. Edgar*	0.664
6.	*Water for Elephants*	0.664
7.	*Master and Commander: The Far Side of the World*	0.639
8.	*Road to Perdition*	0.629

As with the nodes of the graph, we can validate the efficiency of the relations embeddings by analyzing relation similarity. Table 6.6 presents one such example listing embedings most similar to the relation $/people/person/ethnicity$.

Table 6.6: Relation cosine similarity predictions to: */people/person/ethnicity*

Relations		Cosine Similarity
1.	*/people/person/place_of_birth*	0.991
2.	*/people/person/nationality*	0.978
3.	*/people/person/profession*	0.975
4.	*/people/person/gender*	0.967
5.	*/people/person/spouse_s/../spouse*	0.961
6.	*/people/person/religion*	0.958
7.	*/people/person/places_lived/../location*	0.956
8.	*/people/person/spouse_s/../location_of_ceremony*	0.954

Next, we analyze the effect of the preprocessing steps on the link prediction performance. Figure 6.7 presents a comparison of four different configurations; using only label encoded values, using only node embedding, only relation embedding, or both embedding approaches.

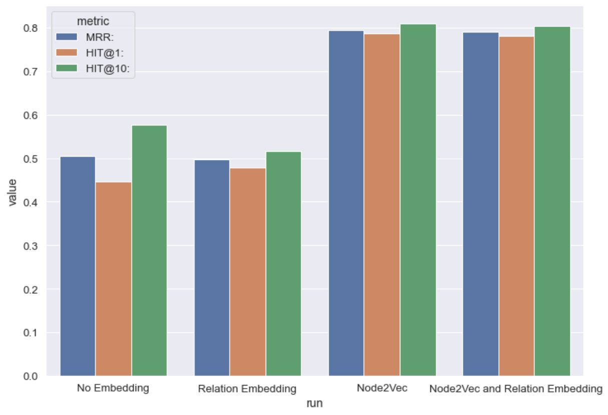


Figure 6.7: Preprocessing effect on LP performance

6.1.5 LP Examples

We use an example from the testing set $(Jack\,Bruce, /music/artist/genre, hard\,rock)$ to inspect our models predictive abilities. We do both head and tail prediction on this triple, rank the head and tail probabilities, and show the top 10 heads and tails.

$\underset{e\in\mathcal{E}}{top}\ 10\ \phi(e,\ music\,genre,\ hard\,rock)$		$\underset{e\in\mathcal{E}}{top}\ 10\ \phi(Jack\,Bruce,\ music\,genre,\ e)$	
1.	*Alan Parsons*	1.	*blues rock*
2.	*Gregg Allman*	2.	*experimental rock*
3.	*Roger Waters*	3.	*art rock*
4.	*Jeff Lynne*	4.	*baroque pop*
5.	*Jack Bruce*	5.	*power pop*
6.	*Steve Howe*	6.	*latin pop*
7.	*John Cale*	7.	*funk rock*
8.	*Maurice Gibb*	8.	*jazz fusion*
9.	*Steve Hackett*	9.	*heartland rock*
10.	*Rick Wakeman*	10.	*post-grunge*

Table 6.7: Head and tail predictions of the triple (*Jack Bruce*, */music/artist/genre*, *hard rock*).

6.2 Fake News Detection

The mrGraphStar model, trained on the FB15k dataset, has been utilized to classify the veracity of the FakeNewsNet dataset. We present both the results on the automatically extracted triples from the entire dataset FNN_{Auto}, and the manually annotated subset FNN_{Manual}. The results of our model will be discussed in section 7.2. The classification performance is measured through various metrics, such as; accuracy, precision, recall, and F_1 score. While we have not utilized the FakeNewsNet dataset like other algorithms to train the classification model, we still present a performance comparison to state-of-the-art models. The results of the compared models are gathered from [118].

Table 6.8: Experimental results on FakeNewsNet

	PolitiFact				**GossipCop**			
	$A(c_i)$	$P(c_i)$	$R(c_i)$	F_1	$A(c_i)$	$P(c_i)$	$R(c_i)$	F_1
RST	0.607	0.625	0.523	0.569	0.531	0.534	0.492	0.512
LIWC	0.769	<u>0.843</u>	0.794	0.818	0.736	**0.756**	0.461	0.572
TextCNN	0.653	0.678	0.863	0.760	0.739	0.707	0.477	0.569
HAN	<u>0.837</u>	0.824	<u>0.896</u>	<u>0.860</u>	<u>0.742</u>	0.655	0.689	<u>0.672</u>
dFEND	**0.904**	**0.902**	**0.956**	**0.928**	**0.808**	<u>0.729</u>	<u>0.782</u>	**0.755**
MrGS	0.410	0.443	0.846	0.581	0.471	0.456	**0.981**	0.623

Due to the lack of quality in triples extracted from FakeNewsNet in FNN_{Auto}, we also compare the performance on the manually annotated subeset FNN_{Manual}. The results are presented in Table 6.9.

Table 6.9: Comparison of experimental classification results on triple datasets

	FNN_{Auto}	FNN_{Manual}
Accuracy	0.436	0.500
Precision	0.449	0.506
Recall	0.901	1.000
F1	0.599	0.667

In-domain vs out-of-domain entities

We suspect issues to arise due to the lack of overlap in entities extracted from the fake news dataset and the entities the LP model has been trained on. The effect of this finding on fake news detection performance is presented in Table 6.10.

Table 6.10: Experimental classification results on in-domain vs out-of-domain entities from FNN_{Manual}

	In-domain	Out-of-domain
Accuracy	0.667	0.385
Precision	0.714	0.385
Recall	0.714	1.000
F1	0.714	0.556

6.2.1 Triple classification

In addition to evaluating the performance of our fake news detection classifier, we also tested mrGraphStar's ability to predict whether a triple is true or not. Thus, a label indicating the veracity of each triple $\langle h, r, t \rangle \in FNN_{Manual}$ was added. The triple classification performance is reported in Table 6.11.

Table 6.11: Experimental triple classification results

	FNN_{Manual}
Accuracy	0.505
Precision	0.923
Recall	0.316
F1	0.471

In-domain vs out-of-domain entities

We suspect issues to arise due to the lack of overlap in entities extracted from the fake news dataset and the entities the LP model has been trained on. The effect on in-domain vs out-of-domain entities on the model's ability to predict whether a triple is true or not is shown in Table 6.12.

Table 6.12: Experimental triple classification results on in-domain vs out-of-domain entities from FNN_{Manual}

	In-domain	Out-of-domain
Accuracy	0.931	0.182
Precision	0.895	0.333
Recall	1.000	0.059
F1	0.944	0.100

Chapter 7

Discussion

In this chapter we present insight as to what has been learnt in this research. The chapter will present a discussion of the results presented in chapter 6, with the aim of reasoning about the lacks and strengths of the presented approach, explaining the underlying difficulties of the task. Results of the link prediction model and fake news detection framework are discussed separately in section 7.1 and section 7.2, respectively. Lastly, section 7.3 presents a discussion regarding the manner in which we conducted the research, including research strategy, choice of datasets, and technical challenges.

7.1 Link Prediction

Though based on an architecture showing advancements in state-of-the-art performance on many graph-related tasks, mrGraphStar does currently not pose a threat to the best performing link prediction models. However, with regards to the LP classification and the rank metrics, our model shows results in close proximity, and further development might enable mrGraphStar to surpass the compared models. Referring to Table 6.1, out of the ten most probable entities in a head or tail prediction, about 81% of the predicted entities are correct. Furthermore, the correct entity is predicted as most probable in about 77% of the cases. The potential reasoning for these numbers will be further investigated in the following subsections.

7.1.1 Scoring Function

The strong result of ComplEx and TuckER highlights the contribution of inverse relation pairs to the performance of solutions on FB15k. We suspect this is due to the fact that they are the only models to explicitly model asymmetry in relations. Like most models, mrGraphStar is not fully expressive. This is a consequence of the utilization of DistMult as its decoder. When comparing to FB15k-237, where inverse relation pairs have been removed, we see that the same models still perform best, but there is a significantly more even distribution. While we only investigate latent feature models for LP, it has been shown that observed feature models do not perform well on this new dataset in contrast to the original FB15k dataset, explaining the low performance despite the significant drop in relations to model.

A notable difference in our model compared to the others in Table 6.1 is that the $Hits@K$ metrics are very close. There is a discrepancy of about 10% to 20% for most models, while only about 4% in mrGraphStar. We suspect that the model might be scoring triples very close to each other, pushing triples to either end of the probability space. By inspecting the scores given to the triples, we discover that many triples are given the same score, thus achieving high $Hits@1$, but only barely better performance at $Hits@10$. With min-policy used to determine ranks might therefore give an artificially high score on the rank metrics. This might explain the reason we achieve the leading $Hits@1$ metric on the FB15k-237 dataset while simultaneously being second to last in $Hits@10$.

7.1.2 Loss Function

As LP models work under the Open World Assumption, any unseen triples, even the ones obtained with corruption, cannot be considered false with certainty. Increasing the number of generated corrupted triples per triple in the training set has shown a positive correlation with LP models' performance on FB15k [128]. As a result, we balance the benefits of increasing this number with the risk of including true triples in the corrupted triples set. Since our dataset includes asymmetric relations, the construction of corrupted triples could mean that a corrupted triple can be the same as a true triple, with the head and tail switched. This has a negative effect on our scoring function, DistMult, as it only supports symmetric relations and will give both triples the same score. To alleviate this problem, we make the conscious choice of producing an intermittent symmetric dataset. This dataset reduces the chance of having randomly generated positives in the corrupted sampling set. Before training, the newly created symmetric edges are removed. In addition, half of the newly created corrupted triples are removed to have an equal number of triples in both sets. The results of our proposed generation of corrupted samples are illustrated in Figure 6.6, showing that the model benefits from this approach.

The mrGraphStar model uses the same loss function as the original architecture; Binary Cross-Entropy. Even though this loss function, a version of logistic losses, has been shown to work best for scoring functions based on tensor decomposition such as DistMult [128], a downside is that it is based on the binary task of prediction. This means that the loss function forces the model to predict values very close to either 0 or 1 and could therefore be one reason our model produces similar prediction scores for triples. Thus, an interesting comparison would be to investigate the effect of adopting a rank-sensitive loss function. One such example is the *Pairwise Ranking loss*, defined as;

$$\mathcal{L} = \sum_{\langle h,r,t\rangle \in \mathcal{G}_{train}} \sum_{\langle h',r,t'\rangle \in \mathcal{G}^{corr}_{train}[\langle h,r,t\rangle]} [\lambda - \phi(h,r,t) + \phi(h',r,t')]_+ \tag{7.1}$$

where λ is a margin hyperparameter, $\mathcal{G}^{corr}_{train}[\langle h,r,t\rangle]$ is the set of constructed corrupted triples of $\langle h,r,t\rangle$, and $[x]_+ = max(0,x)$. Instead of forcing corrupted triples to be assigned the value 0, this loss just enforces negative samples to have lesser scores than positive ones. As noted by [144], this, in turn, will better adhere to the Open World Assumption.

7.1.3 Training

The model hyperparameters were inspected through grid search to find the optimum settings. The lower and upper bounds for each component in the grid search were selected based on commonly used values and promising values used during the creation of the model. Deciding on the number of epochs was a little more complicated as we experienced declining rank metrics, but increasing LP classification metrics. We chose to prioritize rank metrics as it reflects the performance of our task best, thus setting the epochs to 100. This decision is backed by the curves shown in Figure 6.5, where rank metrics seem to plateau at about 100 epochs.

The shape and dynamics of a learning curve can be used to diagnose our LP model's behavior and perhaps suggest the type of configuration changes that may be made to improve learning, performance, or both. As shown in Figure 6.3 and Figure 6.4, we achieve higher AP scores on $\mathcal{G}_{test}$ compared to $\mathcal{G}_{train}$, indicating our model is generalizing well. This implies that our model would yield accurate estimates on new observations that were not part of the original training dataset. On the other hand, there are reasons to suspect the model is trained for too long, which again could lead to the issue of overfitting. This may be supported by looking at the plot of training loss, which is not stabilizing but continuing to decrease slightly with experience. The justification for the high choice of epochs, even though the model is only barely improving on the AP training objective, comes from, as mentioned above, the finding that rank metric scores continue to increase.

The runtime displays a somewhat stochastic depiction of the training, both in time and performance, as shown in Table 6.3. The reasons for this randomness might be caused by the method used in the corrupted sampling, which creates an arbitrary set of corrupted triples each time. Another reason might be the runtime environment as the Idun cluster assigns GPUs of varying type for each job. The inherent randomness in the model might also contribute to the somewhat unintuitive results. Furthermore, the runtime and performance was somewhat stochastic as the Idun cluster assigns GPUs of varying type for each job. It should therefore have been conducted multiple identical grid searches, where the average of the values would give a more accurate result.

7.1.4 Preprocessing

The findings of our experiments would seem to demonstrate that the embeddings produced by node2vec have successfully encapsulated the topological properties of the graph, as well as the connectivity and attributes of nodes and edges. From Table 6.5, we observe that the cosine measure of the node embedding of *Blood Diamond* is closest to nodes all representing movies. All of the predictions are in the same movie genre *Drama* with language *English*. There are also other similarities such as; two of the top eight predictions where also released by *Warner Bros*, *Leonardo DiCaprio* also stars in two of the predictions, three predictions also have music by *James Newton Howard*. The quality of the preproccesing embeddings lead us to believe we can attain the benefit of joint learning in GNN models [140], yielding better results for our LP model. Figure 6.7 clearly demonstrates this. We observe a significant increase in performance on the models trained with optimal parameters for 100 epochs.

As the relation embeddings are not updated during the training of the GNN, we anticipated that providing vector representation encapsulating their semantic meaning would enable the network to better model the topology of the graph during training. As shown in [141], the hierarchical relation structure of relations, such as in FB15k, can be utilized to produce better results. Our experiments do, however, not support this finding. The relation embedding shows little to no effect and actually seems to result in slightly worse rank metrics.

7.1.5 Concluding Remarks

Our LP model represents a successful implementation of an attention-based GNN model to conduct link prediction in knowledge graphs. The extension of the GraphStar architecture to incorporate edge features in prediction has shown ability to reach performance in close proximity to that of state-of-the-art models, on both FB15k and FB15k-237. Examples shown in Table 6.7 seems to reassure the capabilities of the model by providing solid head and tail predictions. However, some issues remain, and concerns about the inability to model asymmetric relations, similar prediction scores, and thus the choice of loss function.

7.2 Fake News Detection

As shown in Table 6.8, our framework underperforms compared to state-of-the-art classification models. While it achieves a high recall, accuracy and precision scores are low due to the fact of predicting a large number of false positives. Although the classification results may be discouraging, the results on in-domain triple classification in Table 6.12 indicate promising potential of the framework. The following subsections will discuss these classification results, as well as the strengths and weaknesses of the proposed detection framework.

7.2.1 The Proposed Framework

Our approach differs from other fact-checking frameworks in that the knowledge of the model is solely based on an external source. This is done as we ultimately want the model's knowledge to be based on knowledge graphs authored by humans, making the database have little or no noisy facts. Further, most other approaches utilize the dataset tested as either the primary or additional training data. In [123], the model is not tested on actual news corpora but tested exclusively on the LP performance on the knowledge graphs used for training. In [15], the authors try multiple strategies of training the LP model on a combination of triple extracted news corpora and an external knowledge graph. Due to the low quality of state-of-the-art triple extraction frameworks, we decide to avoid this. As we cannot guarantee the veracity of automatically extracted claims, we do not want to use this as training data for our model. Additionally, [15] only uses the titles and the first two sentences of each article to produce the summaries used for triple extraction. This is a considerable weakness as it overlooks most of the content of the text, which may contain false claims. Our framework makes a point out of extracting claims from the entirety of the text.

There are, nevertheless, some issues with our framework and knowledge-based approaches in general. These approaches only consider the features of knowledge graphs, omitting global semantic features of news, which also provide critical information for fake news detection. They exclusively consider features from the text, ignoring all non-textual content features as well as contextual features. These approaches only detect false claims revolving around two entities and the relation between them. This may be an issue, as documents contain many claims involving the assignment of attributes of an entity, e.g., "The earth is flat" or "Barack Obama is 4 years old".

Another underlying issue in utilizing an LP model for fake news detection is the discrepancy between its and a human's view of semantic similarity. While the distinction between, e.g., 'Islam' and 'Protestantism' is substantial to a reader, our LP model will assign a relatively high probability to any head representing a person, and the predicate $/people/person/religion$.

As a last note on the framework in itself, reducing the rate of false positives can be done by deciding a higher confidence requirement, or threshold, to call a triple fake. We have decided not to attempt this strategy, and the reason for it is twofold. Firstly, this decision will change the fundamental objective of the LP model, which in itself should be able to decide whether a triple is true or not. Further, due to the strong results of in-domain triple classification, we argue the issue does not lie with the threshold but rather with the dataset domain.

7.2.2 Domain overlap

One of the main takeaways from analyzing the effect of domain overlap is that our framework is deficient when the entities or relations extracted do not exist in a knowledge base. Although we made the conscious choice of including these entities as input to the model, their embeddings are not updated during the message passing as they are not connected to the graph. Consequently, triples including entities not originally a part of the knowledge graph are exceedingly likely to be predicted as false. As a large number of entities extracted from the fake news dataset are not present in FB15k, this, in turn, leads to poor classification performance.

Throughout the experimental results, we notice a significant increase in performance on triples that include in-domain entities. While the fake news classification performance is still low, we argue this to be a fact of the triples containing the false information possibly being filtered out. The large discrepancy in triple classification performance further supports this intuition. Yielding an F_1 score of 0.944 on in-domain triples indicates that given a more extensive domain overlap, classification results would improve significantly. One possible solution to this problem, suggested in R-GCN, is to adopt an entity encoder in the link prediction model. Whereas our approach use a single, real-valued vector $\vec{e}_i$ for every node $v_i \in \mathcal{V}$, they compute representations through an R-GCN encoder with $\vec{e}_i = \vec{h}_i^{(L)}$. This could enable our framework to better predict out-of-domain triples.

Further, an issue that remains is the temporal aspect of the model's knowledge. For such a system to be successful in real-world applications, one would need to maintain the knowledge graph and model continuously. As noted in [118], research that attempted to use external sources to fact-check the claims in news articles may not be able to check newly emerging events.

7.2.3 Triple Extraction

Among the most obvious limitations of our framework is the triple extraction framework's inability to reliably model the content of a text. In addition to extracting only a few of the stated claims of a text, the quality of the ones extracted is mediocre. As shown in Figure 5.4, one of the issues is that a few select relations dominate the constructed dataset FNN_{Auto}. The steps of coreference resolution and entity linking are also lacking, further explaining the reason for the low number of triples. Moreover, the framework is unable to adequately model negations. Quotes and irony also represent an issue, where the model disregards these notions and treat every word in the document literally.

If given satisfactory performance from the triple extraction framework, there will still be some underlying issues remaining. Triples extracted from fake documents may only include true triples. The fake claim in a document may not be extracted, either because of the lacking performance of state-of-the-art models or simply because the false information is not in the form of a claim about a relation between two entities. Lastly, there are issues related to the time consumption of the triple extraction framework. The extraction process takes about one minute per document. While not a big issue when testing in a research setting, this could prove a problem when applied in a real-world scenario.

7.2.4 Interpretability

Over time, the complexity of applications using ML systems has skyrocketed, shifting the interest from not only the task performance of a system but also the other essential criteria such as interpretability of a system's decisions. However, unlike performance measures such as accuracy, interpretability often cannot be completely quantified, hence the need for discussion.

Black boxes typically outperform standard approaches for creating interpretable models, creating a tradeoff between model performance and interpretability. In our case, the suggested fake news detection framework is capable to justify its reasoning by pointing out which claims are deemed false. Thus, having the potential of enabling the users of the framework to verify whether that reasoning is sound or not. In this setting, the link prediction model works as a black box, while the fake news detection system utilizing it accommodates interpretable predictions.

Our framework is utilized in a classification setting, so the information of *why* documents are deemed false might be hidden to a potential end-user. A potential solution is to adopt the model in a non-classification setting, highlighting the claims in a text predicted to be false. This will ultimately create a tool to which empowers the user to make their own conclusion about the content of the text.

7.2.5 Concluding Remarks

While the efforts of adopting mrGraphStar in a fake news detection framework did not yield state-of-the-art classification performance, we have identified multiple reasons explaining why the approach could still bear fruitful results. The proposed framework is strongly reliant on advancements in state-of-the-art triple extraction performance. Additionally, while challenging to ensure, the proposed framework depends heavily on overlapping domains between the knowledge graph and the tested news articles. We argue that a sufficient domain overlap would lead to a significant increase in classification performance. Although tested on a small dataset, we have shown our model's ability to perform triple prediction, reaching an F_1 score of 0.94. Currently, the introduction of out-of-domain entities has led to the frequent occurrence of false positives. This is unfortunate as we contemplate that the cost of false positives is high for fake news detection systems. Further, as interpretability is one of the criteria motivating our research, we also see potential in using our link prediction model in a non-classification setting. As a final thought, with all knowledge-based solutions, the trustworthiness and reliability of the external knowledge remain an issue. One cannot ensure the users of the framework will trust the information contained in the selected knowledge graph.

7.3 Work Practice

This section will discuss the different aspects surrounding the conduction of our research. Although most of the choices made have been explained in the previous sections and chapters, this section will serve as a supplement by further explaining our thought process and reasons for the final decisions. Additionally, we include some discussion and ideas around aspects of this research that could have been done differently.

7.3.1 LP Model

Given the perspective of a fake news detection framework, one could argue that this thesis has put too much focus on the creation of a novel architecture for link prediction in heterogeneous graphs. By choosing to utilize a preexisting model, a significant increase in focus could be put on investigating the potential of our proposed fake news framework. However, on the grounds that this thesis proceeds the efforts and findings of our previous work [1], and that other research exists on similar tasks, we decided the creation and evaluation of an attention-based GNN model for this purpose would serve as a valuable contribution to the field of study. In hindsight, we might have underestimated the time and effort required to ensure our link prediction model achieved satisfactory performance.

7.3.2 Domain and Dataset

There were undoubtedly difficulties in selecting the proper datasets to use in this research. While we wanted to select datasets accommodating comparison within the specific tasks, cross-domain performance was also an essential factor. In order to utilize our model for fake news detection, we researched the existence of preexisting triple extracted datasets, but none were found. As a consequence, we were forced to include the task of constructing such dataset, introducing a new significant source of error. Due to the difficulties in extending the link prediction model, this work was started later than desired. As a result, after investigating the quality of automatic triple extraction, our manually annotated dataset is significantly smaller than we would have liked. A larger manually constructed dataset would have led to more representative results, making it easier to draw conclusions. One weakness that should have been addressed, but was not due to time constraints, is testing multiple external knowledge graphs. Testing on WN18 or DBpedia could have yielded interesting results, both on link prediction and fake news detection performance.

7.3.3 Technical Challenges

We encountered many technical difficulties while conducting the work of this thesis. First and foremost, the field of semantic networks, GNNs and link prediction proved to be a new and challenging domain of research. Lots of time went into gathering a sufficient overview of these research fields. Another challenge arose due to the inherent difficulties of adapting a preexisting codebase. While the authors provided some documentation and basic functionality for LP in undirected single-relation datasets, a lot of changes in the original codebase had to be made. In addition to functional changes, a multitude of third party dependencies where outdated, and needed to be updated in order to support our new needs.

Early efforts in utilizing larger knowledge graphs quickly illuminated the issue of space complexity in LP models. In the current implementation, where we use full-batch gradient descent, memory requirement grows linearly in the size of the dataset. This forced us to adopt smaller datasets in the training evaluation of our model.

Grid searching on a shared cluster with multiple different GPU's proved to be a challenge as well. Sometimes the GPU could handle bigger model sizes, and other times it would just crash. When conducting a grid search across several hyperparameter settings, with runtimes up to six days, it became time consuming to verify which setup was most successful. The grid search had to be restarted three times, before the setup was changed to run on separate instances, leaving an unorganized folder of finished runs. Luckily the runs could be formatted and ordered in python code to give a more readable overview.

Chapter 8

Conclusion

As deliberately designated by the title, the overall topic of this thesis is to leverage attention-based GNNs and knowledge graphs to detect fake news. Like several others before us, we have cast computational-oriented fact-checking as a link prediction task in knowledge graphs. The decision to utilize fact-checking to detect fake news was motivated by the finding that, even though recent advancements in detection techniques have been profound, most approaches do not accommodate interpretability in their predictions. Furthermore, the graph attention network architecture was selected as it currently serves as the most cutting-edge approach for many graph-related tasks. If able to surmount the many barriers present in this task, automatic fact-checking systems built on external knowledge graphs could help mitigate the spread of news content having no significant third-party filtering, fact-checking, or editorial judgment.

Throughout this thesis, we have presented the study, creation and implementation of a novel link prediction model and a framework to automatically extract and verify claims in order to classify news documents. The proposed process enables the model to explain its reasoning by highlighting what claims are deemed false. Users of the framework can therefore verify whether the reasoning is sound or not. The aim of our thesis has three parts: (i) explore the potential for attention-based Graph Neural Networks to reach state-of-the-art link prediction performance in knowledge graphs; (ii) investigate this model's ability to achieve state-of-the-art fake news detection performance; and (iii) evaluate the effectiveness of claim extraction in the context of fact-checking.

The remainder of this chapter will present the conclusion and summarization of contributions in section 8.1. Further, the several compelling venues for further work will be concluded in section 8.2.

8.1 Summary of Contributions

After exploring and gaining a considerable amount of both theoretical and practical knowledge on the domain of fake news, knowledge graphs, and Graph Neural Networks, it is time to look at the results with regards to the research questions. This section will attempt to provide short but concise answers, complementing the findings of this thesis.

RQ1 *Can Graph Attention Networks reach state-of-the-art link prediction performance on multi-relational datasets?*

We have introduced a successful extension of a Graph Attention Network to conduct link prediction in multi-relational knowledge graphs. After a literature review of relevant studies corresponding to our inclusion criteria, the choice was made to base our implementation on the architecture GraphStar. Our experiments have shown the models ability to reach performance in close proximity to that of state-of-the-art models, on both FB15k and FB15k-237. We argue, utilizing a more suitable loss function and decoder, graph attention networks may overtake the currently best performing link prediction models.

RQ2 *Can the link prediction model from RQ1, trained on an external knowledge graph, be used to accurately detect fake news documents?*

The link prediction model, used as part of our framework, is currently underperforming compared to state-of-the-art classification models. Despite achieving a high recall score, the accuracy and precision scores are low due to predicting a high number of false positives. Through investigation we have found the main issues to stem from lacking domain overlap between the external knowledge graph and fake news dataset. Although the classification results may be discouraging, the results of in-domain triple classification indicate promising potential. By creating a manually annotated dataset for fake news detection and triple classification, we have revealed the importance of domain overlap. We argue that a sufficient domain overlap would lead to a significant increase in document classification performance. Although tested on a small dataset, have shown our models ability to perform triple prediction, reaching an F_1 score of 0.94.

RQ3 *Can the link prediction model from RQ2 be extended by integrating a claim extraction system for automatic fake news detection?*

We have proposed a framework combining an information extraction pipeline, which uses a textual input to produce triples, and an adaption of a Graph Attention Network for link prediction in knowledge graphs. Though not extensively tested, the utilization of several state-of-the-art models in our triple extraction pipeline displayed an evident inability to reliably model the content of a text.

Both the amount, quality, and variety of triples are severely lacking. In addition, the pipeline is ineffective due to high time consumption, which may deem the frameworks using it nonviable in real-world applications. Hence, any system that plan on both extracting and verifying statements from text depend strongly on advancements in state-of-the-art triple extraction performance.

8.2 Future Work

We have identified and will now account for several interesting venues for further work on our proposed solutions. This section will attempt to conclude the findings, previously discussed in chapter 7, briefly and concisely.

8.2.1 Link Prediction

There are mainly three venues left unexplored in our current implementation, mr-GraphStar. Firstly, the choice of decoder, or scoring function, makes our model not fully expressive. This entails that it is incapable of modelling asymmetric relations. There are many possible solutions, such as implementing the approach described in ComplEx. This algorithm is able to model asymmetry by utilizing the hermitian product. This would be a good substitution for the current scoring function, albeit introducing the complex plane. Further, the evaluation of a rank-sensitive loss function could yield interesting results. By omitting the current downside of forcing predictions very close to either 0 or 1, we might alleviate the issue of similar triple scores and artificially high $Hits@1$ metrics. Lastly, a particularly interesting research direction would be to analyze the learned attentional weights to perform a thorough analysis of the model interpretability.

As a closing remark, there is room for more exploration in the evaluation of our current implementation. As of now, it has only been tested on a single dataset. By evaluating the model on several other benchmark datasets, one may discover currently hidden strengths or weaknesses. Some datasets to consider include; YAGO3-10[1], WN18[2], and WN18RR2[0].

8.2.2 Fake News Detection

The current state of the art triple extraction systems does not produce triples of sufficient quality, leading to a "garbage in, garbage out" scenario. We have shown that the triples extracted also needs to have a acceptable overlap with the external knowledge graph, although our framework has only been tested after training on a single KG. An interesting next step would be to test the performance of the model on different domains outside of the political spectrum, e.g., society in the

[1]https://yago-knowledge.org

[2]https://github.com/villmow/datasets_knowledge_embedding

Fakeddit dataset[3] or health in the CoAID dataset[4]. Ideally we would also have liked to test the framework using a link prediction model trained on a more extensive knowledge graph.

Due to the lack of a dataset containing text and triples we had to manually create our own. This dataset is small and a more extensive manually annotated dataset could be of great benefit. Furthermore, the naive approach of classifying documents as fake when containing at least one false claim might not be the best approach, testing a more sophisticated scoring algorithm would be of great interest. Finally, a qualitative analysis of the system in a non-classification setting as a tool for supporting fact-checkers could give great insight in both the value of such a system, and the possible limitations it may have.

[3]https://github.com/entitize/fakeddit
[4]https://github.com/cuilimeng/CoAID

Bibliography

[1] O. G. Aspaas and O. C. Vik, "Knowledge Graph Completion through attention-based GNNs," 2020.

[2] K. Rodgers and N. Massac, "Misinformation: A threat to the public's health and the public health system," *Journal of Public Health Management and Practice*, vol. 26, pp. 294–296, May 2020. DOI: `10.1097/PHH.0000000000001163`.

[3] J. Chen, D. Liu, L. Liu, P. Liu, Q. Xu, L. Xia, Y. Ling, D. Huang, S. Song, D. Zhang, *et al.*, "A pilot study of hydroxychloroquine in treatment of patients with common coronavirus disease-19 (covid-19)," *Journal of Zhejiang University (Medical Science)*, vol. 49, no. 1, pp. 0–0, 2020.

[4] E. Shearer and A. Mitchell. (2020). "News use across social media platforms in 2020," [Online]. Available: `https://www.journalism.org/2021/01/12/news-use-across-social-media-platforms-in-2020/` (visited on 04/08/2020).

[5] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018, ISSN: 0036-8075. DOI: `10.1126/science.aap9559`. eprint: `https://science.sciencemag.org/content/359/6380/1146.full.pdf`. [Online]. Available: `https://science.sciencemag.org/content/359/6380/1146`.

[6] V. Carrieri, L. Madio, and F. Principe, "Vaccine hesitancy and (fake) news: Quasi-experimental evidence from italy," *Health Economics*, vol. 28, Aug. 2019. DOI: `10.1002/hec.3937`.

[7] N. Johnson, N. Velasquez, N. Restrepo, R. Leahy, N. Gabriel, S. Oud, M. Zheng, P. Manrique, S. Wuchty, and Y. Lupu, "The online competition between pro- and anti-vaccination views," *Nature*, vol. 582, Jun. 2020. DOI: `10.1038/s41586-020-2281-1`.

[8] T. S. Ulen, "Democracy and the internet: Cass r. sunstein, republic. com. princeton, nj. princeton university press. pp. 224. 2001," 2001.

[9] J. Piacenza. (). "News media credibility rating falls to a new low," [Online]. Available: `https://morningconsult.com/2020/04/22/media-credibility-cable-news-poll/` (visited on 03/21/2021).

[10] Cisco. (). “Cisco annual internet report (2018–2023),” [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf (visited on 11/25/2020).

[11] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, “Knowledge graph embedding for link prediction: A comparative analysis,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 2, pp. 1–49, 2021.

[12] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ser. KDD ’18, London, United Kingdom: Association for Computing Machinery, 2018, pp. 974–983, ISBN: 9781450355520. DOI: 10.1145/3219819.3219890. [Online]. Available: https://doi.org/10.1145/3219819.3219890.

[13] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017, pp. 6348–6358. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/d9896106ca98d3d05b8cbdf4fd8b13a1-Paper.pdf.

[14] Y. Han, S. Karunasekera, and C. Leckie, *Graph neural networks with continual learning for fake news detection from social media*, 2020. arXiv: 2007.03316 [cs.SI].

[15] J. Pan, S. Pavlova, C. Li, N. Li, Y. Li, and J. Liu, “Content based fake news detection using knowledge graphs: 17th international semantic web conference, monterey, ca, usa, october 8–12, 2018, proceedings, part i,” in. Jan. 2018, pp. 669–683, ISBN: 978-3-030-00670-9. DOI: 10.1007/978-3-030-00671-6_39.

[16] A. Bondielli and F. Marcelloni, “A survey on fake news and rumour detection techniques,” *Information Sciences*, vol. 497, pp. 38–55, 2019, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.05.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519304372.

[17] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.

[18] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020.

[19] N. Higdon, *The Anatomy of Fake News: A Critical News Literacy Education*, 1st ed. University of California Press, 2020, ISBN: 9780520347878. [Online]. Available: `http://www.jstor.org/stable/j.ctv1503gc8`.

[20] *Misinformation, n.* In *OED Online*, Oxford University Press, Jun. 2020. [Online]. Available: `https://www.oed.com/view/Entry/119699?redirectedFrom=misinformation` (visited on 04/22/2021).

[21] V. L. Rubin, Y. Chen, and N. K. Conroy, "Deception detection for news: Three types of fakes," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[22] J. H. Brunvand, *American folklore: An encyclopedia*. Routledge, 2006, vol. 1551.

[23] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter,* vol. 19, no. 1, pp. 22–36, 2017.

[24] S. Elkasrawi, A. Dengel, A. Abdelsamad, and S. S. Bukhari, "What you see is what you get? automatic image verification for online news content," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, IEEE, 2016, pp. 114–119.

[25] G. Verma and B. V. Srinivasan, "A lexical, syntactic, and semantic perspective for understanding style in text," *arXiv preprint arXiv:1909.08349*, 2019.

[26] S. Vosoughi, M. N. Mohsenvand, and D. Roy, "Rumor gauge: Predicting the veracity of rumors on twitter," *ACM transactions on knowledge discovery from data (TKDD)*, vol. 11, no. 4, pp. 1–36, 2017.

[27] E. J. Briscoe, D. S. Appling, and H. Hayes, "Cues to deception in social media communications," in *2014 47th Hawaii international conference on system sciences*, IEEE, 2014, pp. 1435–1443.

[28] B. Horne and S. Adali, "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*, vol. 11, 2017.

[29] N. Ruchansky, S. Seo, and Y. Liu, "Csi: A hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 797–806.

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

[31] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning," Association for Computational Linguistics, 2017.

[32] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[33] H. Zhang, Z. Fan, J. Zheng, and Q. Liu, "An improving deception detection method in computer-mediated communication," *Journal of Networks*, vol. 7, no. 11, p. 1811, 2012.

[34] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[35] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[36] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[37] L. Zeng, K. Starbird, and E. Spiro, "# unconfirmed: Classifying rumor stance in crisis-related social media messages," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 10, 2016.

[38] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[39] S. Dungs, A. Aker, N. Fuhr, and K. Bontcheva, "Can rumour stance alone predict veracity?" In *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 3360–3370.

[40] N. K. Manaswi, "Rnn and lstm," in *Deep Learning with Applications Using Python : Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras*. Berkeley, CA: Apress, 2018, pp. 115–126, ISBN: 978-1-4842-3516-4. DOI: 10.1007/978-1-4842-3516-4_9. [Online]. Available: https://doi.org/10.1007/978-1-4842-3516-4_9.

[41] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems*, IEEE, 2010, pp. 253–256.

[42] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," *arXiv preprint arXiv:1809.08037*, 2018.

[43] Y.-C. Chen, Z.-Y. Liu, and H.-Y. Kao, "Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification," in *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, 2017, pp. 465–469.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

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

[46] H. Jwa, D. Oh, K. Park, J. M. Kang, and H. Lim, "Exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert)," *Applied Sciences*, vol. 9, no. 19, p. 4062, 2019.

[47] C. Liu, X. Wu, M. Yu, G. Li, J. Jiang, W. Huang, and X. Lu, "A two-stage model based on bert for short fake news detection," in *International Conference on Knowledge Science, Engineering and Management*, Springer, 2019, pp. 172–183.

[48] J. Morgan, "The anatomy of income distribution," *The review of economics and statistics*, pp. 270–283, 1962.

[49] K. K. Kumar and G. Geethakumari, "Detecting misinformation in online social networks using cognitive psychology," *Human-centric Computing and Information Sciences*, vol. 4, no. 1, pp. 1–22, 2014.

[50] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999.

[51] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[52] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[53] S. Hosseinimotlagh and E. E. Papalexakis, "Unsupervised content-based identification of fake news articles with tensor decomposition ensembles," in *Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*, 2018.

[54] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[55] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–40, 2020.

[56] A. Motro, "Integrity= validity+ completeness," *ACM Transactions on Database Systems (TODS)*, vol. 14, no. 4, pp. 480–502, 1989.

[57] R. Reiter, "On closed world data bases," in *Logic and Data Bases*, 1978, pp. 55–76.

[58] N. L. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph theory 1736-1936*, English, London: Clarendon Press; Oxford University Press. X, 239 p. £9.50 (1976). 1976.

[59] B. Hopkins and R. J. Wilson, “The truth about königsberg,” *The College Mathematics Journal*, vol. 35, no. 3, pp. 198–207, 2004. DOI: 10.1080/07468342.2004.11922073. eprint: https://doi.org/10.1080/07468342.2004.11922073. [Online]. Available: https://doi.org/10.1080/07468342.2004.11922073.

[60] E. Williamson, *Lists, Decisions and Graphs*. S. Gill Williamson. [Online]. Available: https://books.google.no/books?id=vaXv%5C_yhefG8C.

[61] W. L. Hamilton, “Graph representation learning,” *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[62] J. D. Phillips, W. Schwanghart, and T. Heckmann, “Graph theory in the geosciences,” *Earth-Science Reviews*, vol. 143, pp. 147–160, 2015, ISSN: 0012-8252. DOI: https://doi.org/10.1016/j.earscirev.2015.02.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0012825215000239.

[63] C. T. Arsene, B. Gabrys, and D. Al-Dabass, “Decision support system for water distribution systems based on neural networks and graphs theory for leakage detection,” *Expert Systems with Applications*, vol. 39, no. 18, pp. 13 214–13 224, 2012, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2012.05.080. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412007968.

[64] M. Tsubaki, K. Tomii, and J. Sese, “Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences,” *Bioinformatics*, vol. 35, no. 2, pp. 309–318, Jul. 2018, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty535. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/2/309/27497010/bty535.pdf. [Online]. Available: https://doi.org/10.1093/bioinformatics/bty535.

[65] R. Anirudh and J. J. Thiagarajan, “Bootstrapping graph convolutional neural networks for autism spectrum disorder classification,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3197–3201. DOI: 10.1109/ICASSP.2019.8683547.

[66] O. Sporns, “From simple graphs to the connectome: Networks in neuroimaging,” *NeuroImage*, vol. 62, no. 2, pp. 881–886, 2012, 20 YEARS OF fMRI, ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2011.08.085. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811911010172.

[67] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R.

Garnett, Eds., vol. 28, Curran Associates, Inc., 2015, pp. 2224–2232. [Online]. Available: `https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf`.

[68] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and analysis of online social networks,” ser. IMC ’07, San Diego, California, USA: Association for Computing Machinery, 2007, pp. 29–42, ISBN: 9781595939081. DOI: `10.1145/1298306.1298311`. [Online]. Available: `https://doi.org/10.1145/1298306.1298311`.

[69] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O’Banion, *Examining covid-19 forecasting using spatio-temporal graph neural networks*, 2020. arXiv: `2007.03113 [cs.LG]`.

[70] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” ser. KDD ’18, London, United Kingdom: Association for Computing Machinery, 2018, pp. 2847–2856, ISBN: 9781450355520. DOI: `10.1145/3219819.3220078`. [Online]. Available: `https://doi.org/10.1145/3219819.3220078`.

[71] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[72] O. Lassila, R. R. Swick, *et al.*, “Resource description framework (rdf) model and syntax specification,” 1998.

[73] N. Aggarwal, S. Shekarpour, S. Bhatia, and A. Sheth, “Knowledge graphs: In theory and practice,” in *Conference on Information and Knowledge Management*, vol. 17, 2017.

[74] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, “Knowledge base completion via search-based question answering,” in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 515–526.

[75] H. Cunningham, “Information extraction, automatic,” *Encyclopedia of language and linguistics,*, vol. 3, no. 8, p. 10, 2005.

[76] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st. O’Reilly Media, Inc., 2009, ISBN: 0596516495.

[77] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell, “Toward an architecture for never-ending language learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010.

[78] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Oct. 2019, vol. 3.

[79] G. Zhou and J. Su, “Named entity recognition using an hmm-based chunk tagger,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 473–480.

[80] F. P. Andrew McCallum Dayne Freitag, “Maximum entropy markov models for information extraction and segmentation,” 2000. [Online]. Available: http://www.ai.mit.edu/courses/6.891-nlp/READINGS/maxent.pdf.

[81] A. McCallum and W. Li, “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons,” 2003.

[82] K. Y. Zhiheng Huang Wei Xu. (2015). “Bidirectional lstm-crf models for sequence tagging,” [Online]. Available: https://arxiv.org/pdf/1508.01991.pdf (visited on 11/15/2020).

[83] S. Wu, K. Fan, and Q. Zhang, “Improving distantly supervised relation extraction with neural noise converter and conditional optimal selector,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7273–7280, Jul. 2019. DOI: 10.1609/aaai.v33i01.33017273. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4713.

[84] A. Smirnova and P. Cudré-Mauroux, “Relation extraction using distant supervision: A survey,” *ACM Comput. Surv.*, vol. 51, no. 5, Nov. 2018, ISSN: 0360-0300. DOI: 10.1145/3241741. [Online]. Available: https://doi.org/10.1145/3241741.

[85] S. Vashishth, R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar, *Reside: Improving distantly-supervised neural relation extraction using side information*, 2019. arXiv: 1812.04361 [cs.CL].

[86] T. Liu, K. Wang, B. Chang, and Z. Sui, “A soft-label method for noise-tolerant distantly supervised relation extraction,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1790–1795. DOI: 10.18653/v1/D17-1189. [Online]. Available: https://www.aclweb.org/anthology/D17-1189.

[87] D. Sorokin and I. Gurevych, “Context-aware representations for knowledge base relation extraction,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1784–1789. DOI: 10.18653/v1/D17-1188. [Online]. Available: https://www.aclweb.org/anthology/D17-1188.

[88] Y. Y. Huang and W. Y. Wang, *Deep residual learning for weakly-supervised relation extraction*, 2017. arXiv: 1707.08866 [cs.CL].

[89] L. Getoor and A. Machanavajjhala, “Entity resolution: Theory, practice amp; open challenges,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2018–2019, Aug. 2012, ISSN: 2150-8097. DOI: 10.14778/2367502.2367564. [Online]. Available: https://doi.org/10.14778/2367502.2367564.

[90] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, Jan. 2016, ISSN: 1558-2256. DOI: `10.1109/jproc.2015.2483592`. [Online]. Available: `http://dx.doi.org/10.1109/JPROC.2015.2483592`.

[91] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’08, Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1247–1250, ISBN: 9781605581026. DOI: `10.1145/1376616.1376746`. [Online]. Available: `https://doi.org/10.1145/1376616.1376746`.

[92] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, “Yago2: A spatially and temporally enhanced knowledge base from wikipedia,” *Artificial Intelligence*, vol. 194, pp. 28–61, 2013, Artificial Intelligence, Wikipedia and Semi-Structured Resources, ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2012.06.001`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0004370212000719`.

[93] Y. Deng, “Generalized evidence theory,” *Applied Intelligence*, vol. 43, no. 3, pp. 530–543, 2015.

[94] B. Kang and Y. Deng, “The maximum deng entropy,” *IEEE Access*, vol. 7, pp. 120 758–120 765, 2019.

[95] G. Pasi, M. Viviani, and A. Carton, “A multi-criteria decision making approach based on the choquet integral for assessing the credibility of user-generated content,” *Information Sciences*, vol. 503, pp. 574–588, 2019.

[96] M. Viviani and G. Pasi, “A multi-criteria decision making approach for the assessment of information credibility in social media,” in *International Workshop on Fuzzy Logic and Applications*, Springer, 2016, pp. 197–207.

[97] E. Delavenay and K. M. Delavenay, *An introduction to machine translation*. Thames and Hudson London, 1960.

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

[99] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[100] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[101] W. W. Zachary, “An information flow model for conflict and fission in small groups,” *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[102] A. Grover and J. Leskovec, “Node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[103] K. Miller, M. Jordan, and T. Griffiths, “Nonparametric latent feature models for link prediction,” *Advances in neural information processing systems*, vol. 22, pp. 1276–1284, 2009.

[104] F. Gao, K. Musial, C. Cooper, and S. Tsoka, “Link prediction methods and their accuracy for different social networks and network metrics,” *Scientific programming*, vol. 2015, 2015.

[105] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Neural Information Processing Systems (NIPS)*, 2013, pp. 1–9.

[106] R. Jenatton, N. Le Roux, A. Bordes, and G. Obozinski, “A latent factor model for highly multi-relational data,” in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012, pp. 3176–3184.

[107] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[108] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[109] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018.

[110] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” *arXiv preprint arXiv:1704.01212*, 2017.

[111] K. Cho, A. Courville, and Y. Bengio, “Describing multimedia content using attention-based encoder-decoder networks,” *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

[112] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[113] M. Hardalov, I. Koychev, and P. Nakov, “In search of credible news,” in *International conference on Artificial intelligence: methodology, systems, and applications*, Springer, 2016, pp. 172–180.

[114] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[115] W. C. Mann and S. A. Thompson, “Rhetorical structure theory: Toward a functional theory of text organization,” *Text*, vol. 8, no. 3, pp. 243–281, 1988.

[116] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, “The development and psychometric properties of liwc2015,” Tech. Rep., 2015.

[117] Y. Zhang and B. Wallace, “A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1510.03820*, 2015.

[118] K. Shu, L. Cui, S. Wang, D. Lee, and H. Liu, “Defend: Explainable fake news detection,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 395–405.

[119] Z. Jin, J. Cao, Y. Zhang, and J. Luo, “News verification by exploiting conflicting social viewpoints in microblogs,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, Phoenix, Arizona: AAAI Press, 2016, pp. 2972–2978.

[120] S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu, “Unsupervised fake news detection on social media: A generative approach,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5644–5651, Jul. 2019. DOI: `10.1609/aaai.v33i01.33015644`. [Online]. Available: `https://ojs.aaai.org/index.php/AAAI/article/view/4508`.

[121] X. Zhou and R. Zafarani, “Network-based fake news detection: A pattern-driven approach,” *CoRR*, vol. abs/1906.04210, 2019. arXiv: `1906.04210`. [Online]. Available: `http://arxiv.org/abs/1906.04210`.

[122] A. Magdy and N. Wanas, “Web-based statistical fact checking of textual documents,” in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, 2010, pp. 103–110.

[123] B. Shi and T. Weninger, “Fact checking in heterogeneous information networks,” in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 101–102.

[124] T. Ebisu and R. Ichise, “Toruse: Knowledge graph embedding on a lie group,” *arXiv preprint arXiv:1711.05435*, 2017.

[125] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen, “Interaction embeddings for prediction and explanation in knowledge graphs,” in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 96–104.

[126] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, “Embedding entities and relations for learning and inference in knowledge bases,” *arXiv preprint arXiv:1412.6575*, 2014.

[127] R. Kadlec, O. Bajgar, and J. Kleindienst, “Knowledge base completion: Baselines strike back,” *arXiv preprint arXiv:1705.10744*, 2017.

[128] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, “Complex embeddings for simple link prediction,” International Conference on Machine Learning (ICML), 2016.

[129] I. Balažević, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," *arXiv preprint arXiv:1901.09590*, 2019.

[130] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," *arXiv preprint arXiv:1712.02121*, 2017.

[131] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," *arXiv preprint arXiv:1707.01476*, 2017.

[132] G. Hinton. (2014). "Ama geoffrey hinton," [Online]. Available: `https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/` (visited on 12/02/2010).

[133] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, 2017, pp. 3856–3866.

[134] T. Vu, T. D. Nguyen, D. Q. Nguyen, D. Phung, *et al.*, "A capsule network-based embedding model for knowledge graph completion and search personalization," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 2180–2189.

[135] L. Guo, Z. Sun, and W. Hu, "Learning to exploit long-term relational dependencies in knowledge graphs," *arXiv preprint arXiv:1905.04914*, 2019.

[136] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, Springer, 2018, pp. 593–607.

[137] L. Haonan, S. H. Huang, T. Ye, and G. Xiuyan, "Graph star net for generalized multi-task learning," *arXiv preprint arXiv:1906.12330*, 2019.

[138] X. Wang, R. Girshick, A. Gupta, and K. He, *Non-local neural networks*, 2018. arXiv: `1711.07971 [cs.CV]`.

[139] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang, *Star-transformer*, 2019. arXiv: `1902.09113 [cs.CL]`.

[140] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 5165–5175, 2018.

[141] Z. Zhang, F. Zhuang, M. Qu, F. Lin, and Q. He, "Knowledge graph embedding with hierarchical relation structure," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3198–3207.

[142] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.

[143] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, 2020.

[144] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

O.G. Aspaas & O.C. Vik

Leveraging Graph Attention Networks and Knowledge Graphs for Fake News Detection