Using Knowledge Graphs for Fake News Detection

Siyana Slavova Pavlova

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Aberdeen.

1 4 9 5

Department of Computing Science

2018

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2018

Abstract

Enormous amounts of news are being create and shared over online mediums daily. As a result, it has become increasingly difficult for people to keep track of all the information available online and all the more difficult to outweigh fact from fiction. Therefore, it is imperative that effective automated solutions are found to tackle fake news detection. This project proposes a system which builds an ever growing news knowledge graph and translates it into an embedding model. Parts of this system have been used for a research project to explore the effectiveness of using knowledge graph embedding for fake news detection. Furthermore, this project proposes a comparison between a knowledge graph embedding approach to the topic and a reachability one. The results are positive in favour of the knowledge graph embedding approach when the background knowledge graph has been constructed entirely from real news articles.

Acknowledgements

I would like to express my sincere gratitude to all those who helped and supported me in the making of this project.

First of all I would like to thank my project supervisor Dr Jeff Z Pan for his assistance, support and the helpful advice throughout the duration of this project.

Secondly, I would like to thank Chenxi Li, Ningxi Li and Yangmei Li for sharing their knowledge and experience in the research area of my project.

Finally, I would like to thank my parents, friends, and my boyfriend for their moral support and their patience during the more stressful times of this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the widespread popularization of the Internet, many business domains have embraced technology as a means to survive. The news publishing industry has been no exception. Newspapers have gone from seeing technology as a rival to treating it as a companion, with social media becoming an important element of news distribution [5].

However, due to the open nature of the Internet, the spread of fake news without effective supervision has become easier. Fake news are news articles that are intentionally and verifiably false, and could mislead readers [1]. With characteristics such as low cost, easy access, and rapid dissemination, fake news can easily mislead public opinion, disturb the social order, damage the credibility of social media, infringe the interests of the parties involved and cause a crisis of confidence. We know how it has occurred and exerted an influence in the past 2016 US presidential elections. Therefore, it is important and valuable to develop methods for detecting fake news.

1.1 Motivation

Versions of a popular idea whose origins can be traced back to 1662 have been used by a number of people throughout the 20th century [Investigator]. The most popular version of this idea goes "If you don't read the newspaper, you are uninformed. If you do read the newspaper, you are misinformed."

Figures from the 2017 Reuters Institute Digital News Report [16] suggest that people's opinion on the topic has not changed much. The report shows that only 24% of the respondents think social media does a good job at separating fact from fiction, and 40% think that for the news media. What is more, 29% state that they often or sometimes avoid the news, one of the reasons being that they cannot rely on news to be true.

This, however, should not come as a surprise. The shift from traditional news media to an online news environment, has created a trade-off between fast and convenient access to information and fact-checking and verification. Chen et al.[3] outline two important aspects to explain this: information overload and a high rate of news sharing, especially through social media. Without the need for printing and distributing physical newspapers, and with the help of social media, it has become easier to reach an audience at (almost) no cost and thus produce more sources of information than ever before. However, the information from these sources can be misleading, intentionally or not, or present a highly opinionated viewpoint. Efforts like Snopes[1] contribute immensely to the goal of exposing fake news and providing evidence as to why they are fake.

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

However, due to the vast amount of news articles being produced daily, it is fair to assume that fact checking every single news article manually is far from viable. Therefore, it is imperative that effective automatic fake news detection methods are developed and research in this area continues.

1.2 Requirements and Goal

This project proposes a system which is able to collect online news articles, extract knowledge from them and build an embedding model of the knowledge graph which could be used to determine whether a news article is true or fake.

The goal of this project is to produce a system which builds an embedding model of an ever growing news knowledge graph. Some of the components of the system have been used as a part of a research conducted together with three other students from the university and under the supervision of Dr Jeff Pan. The scope of the research was to explore how effective content-based fake news detection using knowledge graphs, and knowledge graph embedding in particular, is. Therefore, this was also one of the goals of this project. Finally, this project takes a step further to compare the results from the research's experiment with those of another fact-checking system.

It must be noted that initially, the intended scope for this project was to build a system to aid children (aged 9-10 years old and above) spot fake news. Such a system would need to combine fake news detection and text simplification techniques, together with a friendly user interface. Furthermore, this would require a deep research into children psychology. These are all very extensive tasks and after a feasibility study and careful consideration of the components, it was thus decided that only the fake news detection component of this task should be tackled for the purposes of this project. Future work pointers of how the current state of the project can be extended for the purposes of fake news detection for children can be found in Section 6.3.

Thus with the goals described above in mind, the functional and non-functional requirements for the system are defined below.

1.2.1 Functional requirements

1. *FR1: Automatically collect news articles from a list of sources.*

   Provided with a list of sources, the system should be able to collect the articles that appear on the RSS feed of each source and collect important information (title, full body text, date published).

2. *FR2: Automatically extract knowledge from news articles*

   With an existing database which contains news articles (title, full body text, date published, etc. for each article), the system should be able to extract triples from the text and store them.

3. *FR3: Automatically build a Knowledge Graph Embedding model based on the triples extracted*

   With the extracted triples from 2, the system should be able to build a Knowledge Graph Embedding model that can be used for evaluating whether new incoming articles are true or fake.

1.2.2 Non-Functional requirements

1. *NFR1: The system and its storage should be scalable with the amount of articles that need to be processed per cycle.*

2. *NFR2: Existing systems and tools used as parts of any components of the system should be open source or free for non-commercial use.*

1.3 Overview of Related Work

With respect to related work, in recent years, various fake news detection approaches have been proposed. A number of them are based on news content. For these approaches, the most straightforward way is to check the truthfulness of the statements claimed in news content, that is to do fact checking [21]. Then, the computational fact checking approaches focus on path reachability trying to find a path in an existing knowledge graph for a given triple [13, 19, 20]. Besides these content-based (or knowledge-based) approaches, there are other style-based ones which focus on capturing the writing style of news content as features to classify news articles [7, 8, 11, 23]. Finally, some systems exist which concentrate on metadata, such the the source of a news article to determine its truthfulness. A more in-depth research into related work is proposed in section 2.3.

1.4 Objectives

As described in section 1.2, a part of this project was participating in a research on content-based fake news detection. Therefore, the objectives of this project partially overlap with the initial stages of said research, a paper for we have submitted to the 17th International Semantic Web Conference (ISWC 2018)[2]. The objectives of this project are presented here as research questions, with a brief description of the research questions addressed by the paper which are not in the scope of this project.

RQ1: Can we use incomplete and imprecise knowledge graphs for fake news detection? Computational knowledge-based approaches mainly focus on simple common relations between entities, such as "*author*", "*hasChild*", "*dateOfBirth*". In the paper, we argue that the knowledge graphs they use are too incomplete and imprecise to cover the complex relations that appear in fake news articles. For example, the triple *(Anthony Weiner, cooperate with, FBI)* extracted from a news article has the entities "*Anthony Weiner*" and "*FBI*", and the relation "*cooperate with*". While the entities can be easily found in open knowledge graphs, the relation is not. The paper and this project attempt to utilise knowledge graphs built on real news articles that are more complete and precise than open ones in that they cover the relations and entities appear in news articles. FR1 and FR2 address this issue by proposing tools to automatically collect news articles and extract triples from them in order to enrich the knowledge graph.

RQ2: What happens if we don't have a knowledge graph in the first place, but only have articles? For fake news detection, it is likely that at the beginning there is no knowledge graph to rely on for fact-checking. In the paper, we address this problem by proposing building a knowledge graph based on the news articles and using related sub-graphs from open knowledge graphs. This project focuses on the first aspect, building a knowledge graph from news articles and this problem is addressed by FR1 and FR2.

[2]http://iswc2018.semanticweb.org/

RQ3: How can we use Knowledge Graph Embedding for fake news detection to reasonably utilize the complete and precise knowledge graph related to the fake news topic once we have it? In the research paper, we propose using TransE [2] to train a single model on each knowledge graph that has been have constructed to compare their performance. This problem is addressed by FR3 of the system proposed in this project.

RQ4: Is a single embedding model based on one related knowledge graph enough? This research question is addressed in the paper but not in this project. It explores whether using Binary TransE models which combine a positive single model and a negative single model perform better at fake news detection than single TransE models.

RQ5: How does knowledge graph embedding for fake news detection compare to reachability approaches? Some studies show that reachability approaches such as Knowledge Stream perform better than embedding approaches such as TransE on fact checking. These, however have been tested on rich knowledge graphs, such as DBPedia. Do reachability approaches perform better than embedding approaches on scarcer knowledge graphs, which contains a wider variety of relations nonetheless, such as that constructed from real news articles as well?

1.5 Main Contributions towards the Objectives

Experiments on test datasets show that knowledge graphs built on real news articles can play an important role in fake news detection. Further experiments show that TransE performs better than KnowledgeStream on knowledge graphs built from news articles, despite KnowledgeStream's better performance on fact checking using open knowledge graphs. As a part of the joint research project, I have contributed to the following aspects of the research towards its objectives:

- To the best of our knowledge, we are the first to propose the approach of fake news detection based on the article content by constructing complete and precise enough knowledge graph to cover the fake news articles' topic.

- To the best of our knowledge, we are the first to propose the approach of fake news detection using positive and negative knowledge graph embedding models, and it performs well once the complete and precise knowledge graphs have been obtained.

- We find that a binary TransE model which combines positive and negative single models performs better than the individual one.

Additionally, my contributions towards the objectives of this project alone are:

- I propose an empirical comparison of an embedding approach to a reachability approach on fake news detection using knowledge from real news articles. I show that the embedding approach performs better.

- I propose an automated system which is able to collect up to date news articles, extract knowledge from them and build an embedding model from the constructed knowledge graph.

1.6 Marker's Guide

This project starts by analyzing the current opinions about the online news environment, the reasons behind the ease of spread of fake news and the difficulties arising due to that, and explores why it is necessary to work towards automated solutions to this problem. In this chapter the goals of the project are outlined, the functional and non-functional requirements for the proposed system are set and finally the objectives of the project and my contributions towards them are specified.

Chapter 2 presents existing techniques and background knowledge for the tackling of the problem. Then, related work is presented and analyzed. The most relevant existing systems are assessed against the requirements and gaps are identified.

The design and architecture of the project are presented in Chapter 3 with justification for the design choices and decisions for the components. Alternative approaches are discussed for each of the components.

In Chapter 4 the implementation of the project is presented, with an in-depth discussion of how each component of the system was constructed. Challenges and how they were addressed are also outlined.

The approaches to testing the system are outlined in 5. Then, the results of the evaluation are presented and discussed.

Finally, the conclusion and discussion are presented in Chapter 6. Some pointers for future work are provided.

Chapter 2

Background and Related Work

This chapter outlines some of the concepts and techniques that can be used to answer the research questions from Chapter 1.

2.1 Information Extraction

Information Extraction (IE) is an area of Natural Language Processing. It aims at automatically extracting structured knowledge from an unstructured text. This knowledge can be represented by facts that can then be used as database entries or for building an ontology.

Traditional IE systems require extensive human involvement to hand-craft extraction rules and hand-tag training examples [GATE]. IE systems can be split into two categories: Closed Information Extraction and Open Information Extraction (OIE). The first one requires pre-defined templates for which to extract information and is usually for specific domains and situations (e.g. extract all student names, degrees and student ID number from a set of documents with a similar format). OIE, on the other hand, is not used to find relations in a specific domain. OIE systems can extract unseen relational facts without having a pre-specified vocabulary.

2.2 Knowledge Graph Embedding

A Knowledge Graph (KG) is a collection of entities and the relations between them. KGs have seen a rise in popularity thanks to the developments in automated knowledge extraction. While KGs can be very useful in many tasks, they can often have missing facts [4]. Knowledge Graph Embedding (KGE) is a technique used to populate the KGs. KGE models map entities and relations in a knowledge graph to a vector space in order to predict unknown triples by scoring candidate triples. A number of KGE algorithms have been developed. Inspired by word2vec [15], Translating Embeddings (TransE) [2] has been proposed to learn vector embedding **h**, **r** and **t**, for which $\mathbf{r} \approx \mathbf{t} - \mathbf{h}$. TransR/CTransR [24] and TransD [10] have been proposed to address the issue of TransE when modelling 1-to-N, N-to-1 and N-to-N relations. A further overview of KGE algorithms is given in 2.2.

The KGE approach used in this project is Translating Embeddings (TransE). TransE is a method which models relationships in the knowledge graph by interpreting them as translations operating not on the graph structure directly but on a low-dimensional embedding of the knowledge graph entities, which it has learnt [Vandenbussche]. In TransE, if a given fact (h,r,t), where h is the head (or subject) entity, t is the tail (or object) entity and r is a relation type, is true, then the embedding of h plus some vector representing r should be close to the embedding of t. If the fact is not true, then the embedding of $h+r$ should be far from the embedding of t. If we want to predict missing facts using TransE, i.e. if we have an embedding of an entity h, and a vector representing a relation type r, the closest entity to $h+r$, will be the tail of the fact being predicted. Pierre-Yves Vandenbussche explains TransE in more details in a very easy to follow manner in [Vandenbussche].

2.3 Related Work

In this chapter related work in Fake News Detection and Knowledge Graph Embedding is presented.

2.3.1 Fake News Detection

An effective approach is of prime importance for the success of fake news detection that has been a big challenge in recent years. Generally, those approaches can be categorized as knowledge-based and style-based.

A variety of Fake News Detection approaches exist. The first difference is due to the fact there is no strict definition of what fake news is. Some [http://www.fakenewschallenge.org/] regard a fake news article as an article for whose title is misleading, i.e. it promises that a . This is the click-bait type of articles which attract the reader by promising to reveal some "shocking" information when in reality the article itself only vaguely touches on the topic or presents information that is not at all "shocking". The Fake News Challenge [1] was designed to address one of the problems related to this topic: Stance Detection. The goal of this challenge was to work on an algorithm which, given the title and the text of an article, first decides whether the title is related to the topic discussed in the main text and then, if the article is related to the title, decides whether it agrees with it, disagrees with it or discusses it. The existing baseline prior to the competition had a weighted accuracy score of 79.53%, and the winning team's algorithm had an accuracy of 82.02%.

This project, however, takes Allcott et al.'s [1] definition of fake news as news articles that are intentionally and verifiably false, and could mislead readers. There are a number of approaches to tackle this issue. They are split into three generic categories in this report. The first one takes into account the original source of the article, the second one is style-based and the third one is knowledge-based (or content-based).

A heuristic used by many people when verifying the truthfulness of a news articles is to check the source. If it is a well-known source with good reputation, we generally regard the information they present as true. On the other hand, if it is a less well-known source or one that has an untraditional name, we tend to doubt the information they present and do further research on the topic. A similar approach has been adopted by a number of systems and tools for automatic fake news detection. BS Detector [2] is a Chrome and Mozilla browser extension that searches all links on a webpage for references to unreliable sources. The list on said sources is a manually compiled one. Flock [3], a team messenger used in some business companies, has developed a Fake News Detector for their chats which relies on a database of around 600 verified fake news sources [http://www.business-standard.com/article/technology/flock-unveils-fake-news-detector-to-curb-spread-of-misleading-information-117012000268_1.html].

[1] http://www.fakenewschallenge.org/

[2] http://bsdetector.tech/

[3] https://flock.com/uk/

Style-based. Style-based approaches attempt to capture the writing style of news content. [8] find that there are some similarity between fake news and spam email, such as they often have a lot of grammatical mistakes, try to affect reader's opinion on some topics in manipulative way and use similar limited set of words. So they apply a simple approach for fake news detection using naive Bayes classifier due to those similarity. [7] applies term frequency-inverse document frequency (TF-IDF) of bi-grams and probabilistic context free grammar (PCFG) detection and test the dataset on multiple classification algorithms. William Yang Wange [23] investigates automatic fake news detection based on surface-level linguistic patterns and design a novel, hybrid convolutional neural network to integrate speaker related metadata with text. [11] find that some key words tend to appear frequently in the micro-blog rumor. They analyze the text syntactical structure features and presents a simple way of rumor detection based on LanguageTool.

Knowledge-based. The most straightforward way to detect fake news is to check the truthfulness of the statements claimed in news content. Knowledge-based approaches are also known as fact checking. The expert-oriented approaches, such as Snopes[4], mainly rely on human experts working in specific field to help decision making. The crowdsourcing-oriented approaches, such as Fiskkit[5] where normal people can annotate the accuracy of news content, utilize the wisdom of crowd to help check the accuracy of the news articles. The computational-oriented approaches can automatically check whether the given claims have reachable paths or could be inferred in existing knowledge graph. [13] take fact-checking as a problem of finding shortest paths between concepts in a knowledge graph; they propose a metric to assess the truth of a statement by analyzing path lengths between the concepts in question. [20] propose a novel method called "Knowledge Stream(KS)" and a fact-checking algorithm called Relational Knowledge Linker that verifies a claim based on the single shortest, semantically related path in KG. [19] view fake news detection as a link prediction task, and present a discriminative path-based method that incorporates connectivity, type information and predicate interactions.

2.3.2 Knowledge Graph Embedding

[2] propose a method, named TransE, which models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities. TransE is very efficient while achieving state-of-the-art predictive performance, but it does not perform well in interpret such properties as reflexive, one-to-many, many-to-one, and many-to-many. Thus, [24] propose TransH which models a relation as a hyperplane together with a translation operation on it. [12] propose TransR to build entity and relation embeddings in separate entity space and relation spaces. TransR learns embeddings by first projecting entities from entity space to corresponding relation space and then building translations between projected entities. [10] propose a model named TransD, which uses two vectors to represent a named symbol object (entity and relation), and the first one represents the meaning of a(n) entity (relation), the other one is used to construct mapping matrix dynamically.

[4]http://www.snopes.com/
[5]http://fiskkit.com

2.4 Summary

In this chapter, related work on Information Extraction, Fake News Detection and Knowledge Graph Embedding is presented. An overview of similar systems is given and an analysis of the strengths and weaknesses of each one is presented. Furthermore, it is explained how this project differs from each of them along with how it addresses the gaps that each of the other approaches has with respect to the requirements specified in Section 1.2.

Chapter 3

Design and Architecture

This chapter covers the system design and architecture based on the list of requirements. The techniques and tools described in Chapter 2 have been used. The choices and decisions made before the implementation of the system are discussed in detail.

The project consists of two main parts: the system proposed to build knowledge graph embedding models from real news articles which, and an experiment to compare the performance of an embedding approach to a that of a reachability approach to tackling fake news detection. The first is made to serve as a tool to for the research paper discussed in Chapter 1 and the latter is designed to compare the results obtained from the research paper to those of a different system in the same context. In this chapter the architecture of the system and the design of each of its components is discussed first and the details of the experiment after that.

The system architecture consists of four main steps as shown in Figure 3.1. The first component collects news articles from a number of sources in textual format daily and stores them. The second component creates summaries for the articles collected by the first component. The third one extracts triples from the summaries and finally, the fourth component creates a Knowledge Graph Embedding model based on the triples extracted by the third one. A summary of which components satisfy which functional and non-functional requirement is given in Table 3.1.

3.1 News Collection

The first component of the system is designed to scrape content from a number of online news sources. In order to get links to the latest news articles, the project makes use of RSS (Rich Site Summary) feeds.

The main reason to use RSS feeds is that it is easy and straightforward to get links to the latest

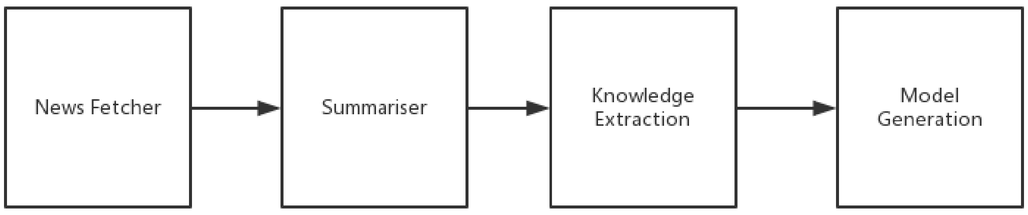


Figure 3.1: System Architecture

Component	Functional Requirement	Non-functional Requirement
News Collection	FR1	NFR1
Summarisation		NFR1
Triple Extraction	FR2	NFR2
Model Generation	FR3	NFR2

Table 3.1: Requirement Satisfaction by Component.

news articles. This component then visits each of the links and scrapes important information about the article on that link: title, full body text, date of publication.

A variety of web scraping libraries are available for Python, the most popular being Selenium, Scrapy, Beautiful Soup and lxml. Selenium is preferred for scraping websites that contain a lot of Javascript and Scrapy is preferred when a complete spider that can crawl through entire websites systematically is needed. Neither of these was the case for this project, therefore the final choice was between Beautiful Soup and lxml. The two have had advantages and disadvantages over each other in the past. However, Beautiful Soup 4 now supports using lxml as an internal parser and lxml includes a soupparser module, so both can be considered equally good for the purposes of this project. Therefore, better familiarity of the author with the former made Beautiful Soup 4 the final choice.

7 news feed sources were considered initially. The choice was mainly based on popularity and trustworthiness. Four UK sources were chosen, namely BBC, Sky News, The Guardian and The Independent, and three US sources, namely ABC, CBS and NBC. Each of the sources was contacted via email to ask for copyright permission for the purposes of this project. One of the British sources pointed out that according to British government law students are allowed to make limited copies of all types of copyright works for non-commercial research or private study []. One of the US sources, however, denied such permission and no similar note was found regarding copyright for non-commercial study in the US. Therefore, only the four UK sources were used for this project.

The two main options for storing the information collected from articles were to use files or to use a database. A database solution was quickly regarded as the best option because it allows for much easier search and querying. MongoDB[1] was chosen for storage due to its ability to scale, and it being a document oriented storage.

3.2 Summarisation

News reporting in English is said to use the Inverted Pyramid Structure, which is believed to have been developed in the 19th century [17]. This means that the most important information from an article is often summarised within the first two sentences. This is the most common structure fro print, broadcast and online news articles in English [18]. Extracting knowledge from the full text of an article. Furthermore, knowledge extraction is a time consuming task. Therefore, using the full article body would cost a lot of time to the whole system without adding much valuable information and thus worsen its scalability. It was therefore decided that for the purposes of this project, the so called default summary of each article will be used, where a default summary

[1]https://www.mongodb.com/

consists of the title plus the first two sentences of a given article. If the full body text is shorter than two sentences (which can sometimes be the case for some breaking news), then the title and the full body text is used.

3.3 Knowledge Extraction

The third component of the system, the knowledge extraction tool, needs to extract triples from the summaries prepared by the second component. A number of tools were considered for knowledge extraction.

The first candidate considered was PIKES[2]. It uses deep NLP techniques to annotate text which can then be semantically interpreted by Semantic Web resources. Furthermore, PIKES matches any entities to existing open knowledge graphs, such as DBPedia and Yago. However, the main disadvantage of PIKES was that it extracts a few tens or even hundreds of triples from a text of a few sentences. A lot of this information was redundant and other tools were considered.

DBPedia Spotlight[3] was considered because of its ability to annotate known entities. However, this was quickly rejected due to its inability to extract relations.

Finally, Stanford's OpenIE[4] was considered. The tool is good at extracting a reasonable amount of triples for the length of the text it is presented with. However, the quality of the triples was not ideal. A combination of OpenIE and a number of tools and techniques used to improve the quality of the triples were the final choice for Knowledge Extraction. The tools and techniques used in addition to OpenIE are:

- Neuralcoref[5] was used to disambiguate pronouns so that a text such as "The man woke up. He took a shower." would be transformed to "The man woke up. The man took a shower". We use Neuralcoref to do this.

- The length of the entities is shortened by finding the head of a phrase and ignoring additional information. For example "western mainstream media like John Kerry" is shortened to "western mainstream media".

3.4 Model Generation

TransE was chosen as the Knowledge Graph Embedding model due to its lower computational cost compared to some other more complex embedding models and its good performance nonetheless. OpenKE's[6] implementation of TransE was used for this component.

3.5 Comparison Experiment

An important aspect of this project as a part of the research mentioned in Chapter 1 was also to evaluate the effectiveness of using KGE for the purpose of fake news detection as a part of co-writing a paper for ISWC 2018. The research questions outlined in Section 1.4 were defined as a result. Therefore, an experiment was designed to test them. The components of the experiment

[2]http://pikes.fbk.eu/
[3]http://www.dbpedia-spotlight.org/
[4]https://nlp.stanford.edu/software/openie.html
[5]https://github.com/huggingface/neuralcoref
[6]http://openke.thunlp.org/index/about

followed a structure similar to that of the main system. However, as this was an one-off experiment, the components were ran one by one instead of as a part of an integrated system. My main contributions towards the experiment were in collecting news articles regarding the 2016 US election from three sources, creating summaries for collected articles along with the articles from a readily available dataset and preparing the embedding models for the collected articles.

However, due to time constraints, we did not manage to compare the performance of the embedding approach proposed to that of any other systems on the same datasets. Therefore, as a part of this project, I compare the embedding approach to another system.

The system chosen for the comparison was KnowledgeStream. As discussed in Section 2.3, KnowledgeStream is a new method proposed by Shiralkar et al. aimed at Fact Checking. As this method's performance has been compared in [20] to TransE's performance with regards to fact checking at a number of datasets extracted from open knowledge graph, and shown to perform better, it is interesting to compare the two method's performance on fake news detection where the underlying knowledge graph is constructed from real news articles.

It must be noted that the KGE approach was evaluated on a three different background knowledge graphs (in order to address RQ1, RQ2 as a part of the paper), then training single TransE models and binary TransE models (to address RQ3 and RQ4 respectively as a part of the paper) to build embeddings in low-dimensional vector space based on the background knowledge graphs. The background knowledge graphs were as follows:

- A knowledge graph based on triples from true news articles
- A knowledge graph based on triples from fake news articles
- A knowledge graph based on a sub-graph from DBPedia which is related to the news topic used for the evaluation.

However, for this project it was decided that only the first knowledge graph based on true articles would be used for comparing KnowledgeStream to TransE. The reason behind not considering the knowledge graph based on fake articles was that Knowledge Stream is designed to be a fact checking system that relies on true facts and as such, using an knowledge base of false facts would not be reasonable. Finally, the reason behind not considering DBPedia was that this projects aims at evaluating the two systems on a knowledge graph that has been constructed from real news articles and using an open knowledge graph was not a priority.

3.6 Summary of Design Decisions and Choices

In this chapter the architecture of the whole system was presented, as well as the design of the experiment used to evaluate the effectiveness of using KGE for Fake News Detection. The design of each component was discussed in detail and the reasons behind choices made were outlined.

Chapter 4

Implementation

This chapter describes how different parts of the system were implemented, based on the requirements, design and architecture described in Chapter 2 and Chapter 3.

4.1 Development Process

Prior to the beginning of the development process, an in-depth research was carried out in order to find the best existing approached and technologies. An agile methodology was followed throughout the development. The system was developed in Python due to its simplicity and rich collection of libraries.

4.2 Data

As this project is made of two main components, there are also two main sets of data for to be considered. One was used for comparing the KGE approach to KnowledgeStream's approach on fake news detection. The other set is the data collected daily by the system proposed by this project.

4.2.1 Comparison Data

Two article bases were used for the experiment: one with fake news and one with news that were regard as true. For the research paper, the fake article base was used for both building a fake news model and using it for testing the approach. For, the comparison experiment, however, the fake article base was used only for testing and not for building an underlying fake knowledge graph for the reasons outlined in 3.5. The fake article base used was from the Kaggle's ' "Getting Real about Fake News" dataset[1]. The dataset contains news articles on the 2016 US Election. 1,400 articles were selected from this dataset as the Fake News Article Base (FAB) for the paper experiment: 1,000 as training data and 400 as test data. The same 400 articles were used as test data for KnowledgeStream. These articles in Kaggle's dataset have been manually labeled as *Bias*, *Conspiracy*, *Fake*, *Bull Shit*, which, for the purposes of this project have been regarded as *fake*.

The true news article base was produced by using the BBC News[2], Sky News[3] and The Independent websites[4] to scrape 1,400 news articles whose topic was âĂIJUS ElectionâĂİ and were published between 1st January and 31st December 2016. These articles have not been manually labeled, however, for the purposes of the experiments, they are regarded as *true*. The statistics of

[1]https://www.kaggle.com/

[2]https://www.bbc.co.uk/

[3]https://news.sky.com/uk

[4]https://www.independent.co.uk/

Article Base	Label	Source	Quantity
FAB	*fake*	Kaggle's "Getting Real about Fake News"	1,400
TAB	*true*	BBC, Sky, Independent news	1,400

Table 4.1: Statistics of fake and true news article bases.

two article bases are shown in Table 4.1. Again, this article base was is divided into two parts, 1,000 for training and 400 are for testing. The same 1,000 articles that were used to train a true model for the KGE approach, were used as the background knowledge for KnowledgeStream and the other 400 were used for testing it.

It it worth mentioning how the third knowledge graph used for training TransE was obtained. This is outside the scope of this project, however, as statistics about it appear in the results too, a brief explanation is given here. This knowledge graph is the so called **D4** knowledge graph. It was from DBPedia using a 4-hop approach to find the subgraph that includes all entities, relations and triples within 4 hops from a topic chosen as the center.

Knowledge Graphs

We produce three knowledge graphs for our experiments: one named FKG from FAB, one named D4 (DBpedia 4-hop) from DBpedia, and one named NKG from TAB.

To produce the two parts of the external KG from an open knowledge graph, as outlined in section 4.1, we use the following steps:

4.2.2 The Main System

When initially built, the system does not require any readily available data. It is the admin's responsibility to collect articles and expand the database. The database can be enriched using the components described in Section 4.3.

4.3 Backend

The components can be run from the *main.py* file in the main project folder. The pseudocode for this has been presented in 1.

4.3.1 News Collection

The news collection script is designed to automatically collect news articles from four sources: BBC News, Sky News, The Indepedent and The Guardian. This component uses FeedParser to find the latest news from the RSS Feeds of the news agencies mentioned above. For each article in each source the following information is collected:

- URL: The unique URL at which the article resides. This is a URL to a webpage on the website of one of the four sources used.
- Title: The title of the article on the source webpage as of the time of collection.
- Body: Full body text of the article on the source webpage as of the time of collection
- Date Published: The date on which the article was published on the source webpage in ISOformat. That is "YYYY-MM-DDTHH:MM:Sec.MsZ".

Each news article is then saved in the database. A few additional fields are added upon the creation of each object in the database. These are:

- ID: A unique ID for the news article. This is the form of a UUID string which is a 36-character string.
- Source: the source of the article. This can be “BBC”, “Sky News”, “The Independent” or “The Guardian”.
- Summary: This field is empty upon the creation of each entry. When the summary for the given article is produces, this field gets updated with the summary.
- Extracted: This is a boolean field which shows whether the triples from an article have already been extracted and saved or not. The default value of this field is False.
- Metadata: Some additional information about each entry is collected here. This is for admin use and does not affect the content of the article. The metadata is a dictionary which currently has four fields:
  - Created Date: This shows when the entry was created is ISOformat.
  - Created By: This shows who created the entry. The default value is “fetcher”. The reason for the field to exist is that it allows for potential scenarios where an entry is added manually or is suggested by users through the web application.
  - Updated Date: This shows when the entry was updated last. The value of this field is the same as Created Date by default and can be changed when edits are made to the entry (e.g. when a summary is added).
  - Updated By: This shows who updated the entry last. Its default value is “fetcher”.

4.3.2 Summarisation

The summarisation component is designed to produce the so called default summaries. When called, this component passes the whole database to check which articles are missing a summary. It then uses a function which takes the title and the full text of an article and returns the default summary as described in Section 3.2. The returned summary is then recorded in the “summary” field for that article. The “updated date” and “updated by” fields are also updated to the current date and “summariser” respectively.

4.3.3 Knowledge Extraction

The knowledge extraction component uses a Stanford CoreNLP Java server to run OpenIE. It further relies on Neuralcoref for pronoun disambiguation, and then NLTK’s WordNetLemmatizer for transforming verbs to their present tense.

The component takes an article summary as input and produces a list of triples extracted from it. The triples are then added to a CSV file, *triples.csv*, where each line has the format *head, tail, relation*, as this is the required format for the next component of the system.

For the evaluation of the KGE approach, two sets of triples were produced: one from true articles and the other one from fake articles.

4.3.4 Model Generation

The model generation component takes a CSV file, *triples.csv*, with the format specified in Section 4.3.3. The component then produces four files: *entity2id.txt, relation2id.txt, train2id.txt* and *type_constrain.txt*.

- *entity2id.txt*: This file contains a list of all the entities in *triples.csv* matched with a unique entity ID.
- *relation2id.txt*: This file contains a list of all the relations in *triples.csv* matched with a unique relation ID
- *train2id.txt*: This file contains a list of all the triples from*triples.csv*. However, in *train2id.txt* each triple is represented by a set of three numbers, which match the ID of the head from *entity2id.txt*, the ID of the tail from *entity2id.txt* and the ID of the relation from *relation2id.txt*.
- *type_constrain.txt* This file contains the type constrain for each relation. This means that for each entity, all head entities and all tail entities are specified.

The model generation component then uses OpenKE which takes the three files above as an input to produce an embedding model based on TransE.

Three single models based on TransE have been produced for testing the embedding approach to fake news detection as a part of the paper: the first model FML using the negative knowledge graph FKG; the second model TML-D4 using the positive knowledge graph D4; the third model TML-NKG using the positive knowledge graph NKG. My contribution to this part of the paper has been in producing TML-NKG.

4.4 Testing KnowledgeStream

The software for KnowledgeStream is readily available online[5]. This is excellent tool if one needs to run experiments using DBPedia as the background knowledge graph. However, for the purposes of this project, it was necessary that a different underlying knowledge graph is used and getting to the stage where this is ready for testing was anything but an easy process. The main reason behind this is that the KnowledgeStream, as most systems, relies on a very specific input format for the input data that is being using and unfortunately, the authors had not provided all the scripts necessary for this to happen. In this section I describe the process through which I went in order to be able to use KnowledgeStream on the same test data as the one used by TransE.

Firstly, KnowledgeStream loads the background knowledge graph from four numpy arrays, saved in four files, namely *undir_data.npy*, *undir_indeg_ve.npy*, *undir_indices.npy* and *undir_indptr.npy*. The authors have provided a script for creating these files. However, the input to this script is the adjacency matrix of the knowledge graph, saved as numpy array too. Since for testing TransE, as a part of the paper, we used regular text files to store our knowledge graphs, I wrote a script to translate them to a numpy adjacency matrix. The script for this is in the main folder of the project under the name *make_adj_matrix.py*.

Once this step was complete, however, running sample tests did not follow immediately. The approach used by KnowledgeStream is based on relational similarity of the knowledge graph. The

[5]https://github.com/shiralkarprashant/knowledgestream

relational similarity is a quantity that show how similar two relations are. The authors of KnowledgeStream propose an original approach to calculating the relational similarity via the line graph of a knowledge graph in [20]. However, no source code was provided for this either. Therefore, an in-depth understanding of their algorithms was needed before I could implement it to suit the needs of my project. The code for my implementation can be found in the *ks_train_plus_test_triples* folder under the name *make_relsim.py*. The code needs a list of the relations and a list of the triples that a knowledge graph contains in order to produce its relational similarity matrix.

Finally, testing KnowledgeStream on the 400 true articles and 400 fake articles required that a file exists for each article which contains all of the triples extracted from the given article and with IDs for each entity and relation accordingly. Once these files existed, they were run in KnowledgeStream to produce scores for each triple in each file. The output for that is under *kstreamnew/output/400pos* and *kstreamnew/output/400neg*. The *computing_scores.py* file was then used to create one joint score for each article, using two different methods. One was to calculate the average of the scores of all triples from a given article and assign that as the article's score. The results from this have been saved in *scores_file_avg.csv*. The other method was to choose the score of the triple with the highest one as the article's score. The results from this have been saved in *scores_file_max.csv*.

The last script related to this part of the project is *evaluate_kstream_data.py*. This script is used to calculate KnowledgeStream's performance, the details of which are outlined in Chapter 5.

Algorithm 1 Main menu

```
 1: function MAINMENU
 2:     print Please choose an option 1-5
 3:     1. Fetch articles
 4:     2. Get summaries of current articles
 5:     3. Add new triples
 6:     4. Generate model
 7:     5. Exit
 8: end function
 9: while true do
10:     MainMenu
11:     choice = User input[1-5]
12:     if choice == 1 then
13:         fetchAll()
14:     else if choice == 2 then
15:         summarise()
16:     else if choice == 3 then
17:         addTriples()
18:     else if choice == 4 then
19:         generateModel()
20:     else if choice == 5 then
21:         raiseSystemExit
22:     else
23:         invalidInput()
24:         choice = User input[1-5]
25:     end if
26: end while
```

```
fnd
├── ks_train_plus_test_triples
│   ├── (...)
│   ├── make_relsim.py
│   └── (...)
├── kstreamnew
│   └── (...)
├── newsData
│   ├── entity2id.txt
│   ├── textTriples.py
│   ├── relation2id.txt
│   └── train2id.txt
├── openKE
│   └── (...)
├── article.py
├── computing_scores.py
├── evaluate_kstream_data.py
├── main.py
├── make_adj_matrix.py
├── modelGeneration.py
├── models.py
├── newsFetcher.py
├── scores_file_avg.csv
├── scores_file_max.csv
├── summariser.py
└── tripleProduction.py
```

4.5 Google Cloud Computing

From the start of the project, some components of the system were ran locally while the rest were ran on an Ubuntu 16.04 Virtual Machine on Google Cloud[6].

The stable environment configuration of the system requires 15GB of memory. In Google Cloud this is a standard virtual machine, which is available for free for up to a year with a trial account.

The system and all its components are currently stored on Google Cloud.

4.6 Summary of Implementation Decisions and Choices

In this chapter the implementation of the project is presented by describing the details of each individual component. The implementation follows the list of requirements specified in Section 1.2 and adheres to the design and architecture decisions presented in Chapter 3.

[6]https://cloud.google.com/compute/

Chapter 5

Testing and Evaluation

In this chapter testing and evaluation methodologies used in the project are described. The Knowledge Graph Embedding approach from the paper that I co-wrote is tested against another approach for fact checking, Knowledge Stream. Furthermore, the final system is evaluated for its usefulness. The results at each evaluation stage are discussed and interpreted and problems are identified.

5.1 Testing

The system follows an agile development approach and thus testing was not only one phase of the project but was performed in the same agile manner. Continuous testing was performed whenever a new component was added to the system or when one component was swapped for another (e.g. Fuseki was used initially for Knowledge Extraction but later on this component was swapped for OpenIE). This allowed for having a system where each component was fully functioning and integrated with the rest of the system. Testing throughout the development process also allowed for the identification of any bugs and their timely resolving.

5.1.1 Unit and Component Testing

As a part of the agile methodology, unit testing was performed concurrently with writing the code for each component. Both Black-Box and White-Box testing were used to ensure not only that the code produces the desired output but that it also works as intended.

5.1.2 Integration Testing

As each component was added in a continuous manner, they were subject to testing as a part of the system at its current state at the moment they were added. This allowed for the identification of any incompatibilities and there timely resolving. Furthermore, considering that the system is highly dependent on a variety of open-source tools, continuous integration was vital for ensuring the system ran smoothly.

5.1.3 Software Testing

The list of the requirements proposed in Section 1.2 is tightly related to each of the components of the system. Moreover, the design choices for each component were taken with the requirements in mind and thus the successful development of each of the components as defined in Chapter 3 ensures that all the proposed requirements are satisfied.

Models	Bias Function	Precision	Recall	F1 score
FML	max bias	0.75	0.78	0.77
	avg bias	**0.80**	0.65	0.72
TML-D4	max bias	0.73	**0.86**	**0.79**
	avg bias	0.77	0.68	0.72
TML-NKG	max bias	0.69	**0.86**	0.77
	avg bias	0.79	0.71	0.75

Table 5.1: Performance of Single TransE Model.

5.2 Evaluation

This section focuses on the results from the comparison between KnowledgeStream and TransE on fake news detection. The results from the evaluation section of the research paper are also included in order to provide a better perspective for the meaning of the comparison results and in order to be able to answer all the research questions outlined in Section 1.4.

5.2.1 Results

The results of the single TransE model with different bias functions are shown in table 5.1. From table 5.1, we can conclude that:

- TML-D4 model performs well for the fake news detection task. This answers a part our RQ1: it means that using incomplete knowledge graph is effective for fake news detection task.

- FML and TML-NKG models also perform well. Therefore, this completes the answer to RQ1 in that using imprecise knowledge graph is also effective for fake news detection. If we do not have a knowledge graph in the first place, but only have articles, contracting a knowledge graph from articles is a effective method too, which answers our RQ2.

- Max Bias significantly outperforms than Avg Bias in terms of F Score. Maybe there are a few true triples in the triple set of one true news, so that the average bias of the triple set becomes smaller. Not all the triples extracted from one fake news is false, so max bias is more useful in fake news detection task.

- What' more, TML-D4 performs a little better than TML-NKG and FML. The results may correlate with the training data of the transE model: There are 1K training news of TML-NKG and FML, but there are 132K entities and 312K triples of the training data set of TML-D4.

The results of B-TransE Model with different bias function are shown in Table 5.2. From Table 5.2, we observe that B-TransE Model is better than Single TransE Model. Therefore we conclude that:

- The approach based on one related knowledge graph is not enough, and combining related knowledge graph with external knowledge graphs is better than one for fake news detection, which answers our RQ4.

Models	Bias Function	Precision	Recall	F1 score
FML + TML-D4	max bias	**0.85**	**0.80**	**0.83**
	avg bias	0.80	0.78	0.79
FML + TML-NKG	max bias	0.75	0.79	0.77
	avg bias	0.81	0.72	0.76

Table 5.2: Performance of Different Models.

Method	Function	Precision	Recall	F1 score
TransE FML	max bias	0.75	0.78	**0.77**
	avg bias	**0.80**	0.65	0.72
KnowledgeStream	max	0.50	0.99	0.66
	avg	0.47	**1.00**	0.64

Table 5.3: Comparison between TransE and KnowledgeStream.

Finally, Table 5.3 shows the results of the comparison of the performance of the TransE FML and that of KnowledgeStream. From the table we observe that while KnowledgeStream has a very high recall value, TransE outperforms it significantly. Therefore, we conclude that:

- TransE is better than KnowledgeStream on the task of fake news detection when the background knowledge graph is constructed from real news articles, despite the latter's better performance at fact-checking when using open knowledge graphs. Thus, we answer our RQ5.

5.2.2 Discussion

It is worth inspecting the possible reasons behind KnowledgeStream's worse performance. One of the most significant differences between the triples extracted from news articles and the triples used by the authors of KnowledgeStream is that in the former there are many more relations present. The undirected graph used by the KnowledgeStream authors contains 24M triples and only 663 relations [20]. On the other hand, from the 1,000 articles used, there were 11,744 triples extracted which covered 3,715 relations. Furthermore, from the 800 test articles, almost the same amount of triples and relations was added. Since these new triples are not a part of the background knowledge graph, if they introduce new relations, these relations will not neighbor any other relations and thus will have a similarity score of 0 with every other relations in the knowledge graph. Since KnowledgeStream relies heavily on relation similarity, it is easy to see why this might be a problem.

Furthermore, from this follows that the score given to many test triples by KnowledgeStream is also going to be 0 (false). My observations of the test data after the triples have been given scores confirm this: most of the triples had a score of 0. This, therefore allows for a lower variety in scores and classifying the results becomes more difficult. However, this also means that using a richer background knowledge graph will likely improve the results.

It is interesting to note KnowledgeStream's high recall on this data though. When classifying the test results, SVM classified almost all of the articles as true (hence the precision measure as well). This is likely due to the fact that the scores of articles were not very varied (about a half or more than half of them had a score of 0). Interestingly though, while there were many false positives, there were almost never any false negatives, thus the high recall.

5.3 Summary

In this chapter the methods used for testing and evaluation are described. The results from the experiments are presented and finally, a discussion of KnowledgeStream's result is provided.

Chapter 6

Conclusion and Outlook

In this project, I propose a start-to-finish automated system which collects news articles online, summarises them, extracts knowledge from the summaries and finally builds an embedding model which can be used for fake news detection. Furthermore, I compare the results of the approach of KGE used in the paper I co-wrote with those of a reachability approach, KnowledgeStream. In this chapter I provide a summary of what has been presented in this project, then move on to discuss the process and finally, I provide some pointers for possible future work.

6.1 Conclusion

The project started by presenting the current situation on the topic of online news and people's mistrust in them. A number of reasons were given as to why the situation might be this way. A system was proposed that works on building a component which can be used for fake news detection. A list of requirements was drawn for such a system and a number of research questions related to using Knowledge Graph Embedding for Fake News Detection were outlined.

Next, background knowledge and related works in the area of Fake News Detection, Fact Checking, Information Extraction and Knowledge Graph Embedding were presented.

Then, the project design and architecture were presented along with an experiment design to compare a KGE approach to fake news detection with a reachability one. The reasons behind the design choices and decisions are provided. The implementation process for the system and the experiment was then described in details.

Finally, the results from the experiment were presented and discussed.

6.2 Discussion

Looking back at the process, this project provided me with many learning opportunities with regards to both knowledge about Computing Science, and Knowledge Graphs in particular, as well as programming tools and skills. I was able to explore current state-of-the-art NLP and Semantic web tools and approaches and combine them to produce a proof of concept system for fake news detection.

Despite it being a demanding project, with a lot of time spent on researching current technologies and building the components of the system, it was well worth the efforts. Having a proof of concept system which can serve as the basis for many future projects on fake news detection tool, made the experience very fulfilling.

There are, of course, some aspects of the project, for which if I could start anew, I would do differently. For example, I would write scripts for automatically collecting fake news articles (for example, from Snopes, provided I had the permission) and would also collect articles on a number of different topics. This would have been useful for a more varied evaluation of the system.

Furthermore, I would spend more time cleaning up and documenting code throughout the duration of the project. This was intended to happen biweekly, however, due to always trying to develop new components, this was done at the end for the whole project.

Finally, there are some aspects of the development process which I wish I had a better idea of before starting the project. An example of this is how demanding and time-consuming it might be to reuse software produced by scholars as a part of their research.

6.3 Future Work

This project has shown that knowledge graphs can play a significant role in content based fake news detection. With the work done so far as a basis, there are a number of ways in which this project can be expanded or improved.

The initial plan for this project was to tackle fake news detection for children. However, in order for this to happen, the sub-task of fake news detection needs to be solved. A potential direction for this project to expand, is to make use of text simplification techniques before triple extraction is performed on the news articles. Related work on simplifying news stories for children can be found in [14]. A complete project on fake news detection for children will also require a friendly and easy to use interface and potential workshops in schools and evaluation from children will greatly help find out what works best. Finally, a number of heuristics can be added to help children reach a conclusion about the truthfulness of a news article by themselves. An example of such a heuristic can be found in []. This considers questions such as what the source of an article was and who wrote it.

Furthermore, the data used in this project covers one main topic, the 2016 US Presidential Election. thus it is concentrated on the political area. Future work can address a wider variety of topics and explore more areas, such as Sports, Entertainment or Business, to name a few. Such a project will explore whether using a number of embedding models (one for each general area or even one for each topic) works better than using one model for the triples from all articles. If more a number of models perform better, the project will further need to address the problem of tagging each article according to its topic(s). This is an important problem in Natural Language Processing and in-depth research for the best performing system will need to be done as a part such a project.

Combining the above with real time news collection, triple extraction and model generation will ultimately produce a fake news detection system for children which can be used as a helping tool at home and in schools. Once such a system exists, it can be used as a browser add-on to help detect possible misleading facts or it can be integrated as a part of a text editor. In order to have a richer dataset, the system will benefit from the use of wider variety of sources and an automated web crawler could add a lot of value to it. The system, as it currently is, collects only articles which are regarded as true. Automatically collecting a variety of articles which are known to be fake (e.g. from Snopes) would be an essential requirement for a system which runs in real time.

6.4 Summary

In this chapter the overall conclusion of this project is discussed. Furthermore, an overview of the achievements and personal experience are presented. Finally, a list of future work ideas is provided along with problems that might be faces and starting points for each one.

Bibliography

[1] Allcott, H. and Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–236.

[2] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

[3] Chen, Y., Niall, J. C., and Rubin, V. L. (2015). News in an online world: The need for an "automatic crap detector". In *Proceedings of the Association for Information Science and Technology*, pages 1–4.

[4] Ebisu, T. and Ichise, R. (2017). Toruse: Knowledge graph embedding on a lie group. *arXiv preprint arXiv:1711.05435*.

[5] Everett, E. (2011). Transformation of Newspapers in the Technology Era. *The elon Journal of Undergraduate Research Of Communication*, 2(2):102–113.

[GATE] GATE. Gate information extraction.

[7] Gilda, S. (2017). Evaluating machine learning algorithms for fake news detection. In *Research and Development (SCOReD), 2017 IEEE 15th Student Conference on*, pages 110–115. IEEE.

[8] Granik, M. and Mesyura, V. (2017). Fake news detection using naive bayes classifier. In *Electrical and Computer Engineering (UKRCON), 2017 IEEE First Ukraine Conference on*, pages 900–903. IEEE.

[Investigator] Investigator, Q. If you don't read the newspaper you are uninformed, if you do read the newspaper you are misinformed.

[10] Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.

[11] Jiang, Y., Liu, Y., and Yang, Y. (2017). Languagetool based university rumor detection on sina weibo. In *Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on*, pages 453–454. IEEE.

[12] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

[13] Luca Ciampaglia, G., Shiralkar, P., Rocha, L., Bollen, J., Menczer, F., and Flammini, A. (2015). Computational fact checking from knowledge networks. 10.

[14] Macdonald, I. and Siddharthan, A. (2016). Summarising news stories for children. In *9th International Natural Language Generation conference*, pages 1–10.

[15] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[16] Newman, N., Fletcher, R., Kalogeropoulos, A., Levy, D. A., and Nielsen, R. K. (2017). Reuters institute digital news report 2017.

[17] Pöttker, H. (2003). News and its communicative quality: The inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

[18] Rich, C. (2015). *Writing and Reporting News: A Coaching Method*. Cengage Learning, eighth edition.

[19] Shi, B. and Weninger, T. (2016). Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102. International World Wide Web Conferences Steering Committee.

[20] Shiralkar, P., Flammini, A., Menczer, F., and Ciampaglia, G. L. (2017). Finding streams in knowledge graphs to support fact checking. *arXiv preprint arXiv:1708.07239*.

[21] Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

[Vandenbussche] Vandenbussche, P.-Y. Translating embeddings *transe*.

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

[24] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119.

Appendix A

User Manual

A.1 Running The System

The system depends on the installation of a number of external tools. These are described in detail in Chapter B.

A.2 Using The Main System

The system offers a text-based interface which allows you to collect current news articles from four online sources, create summaries of the articles collected, extract knowledge from the summaries in order to produce a knowledge graph and build a TransE embedding model based on the knowledge graph.

To use the tool you need to follow a number of steps:

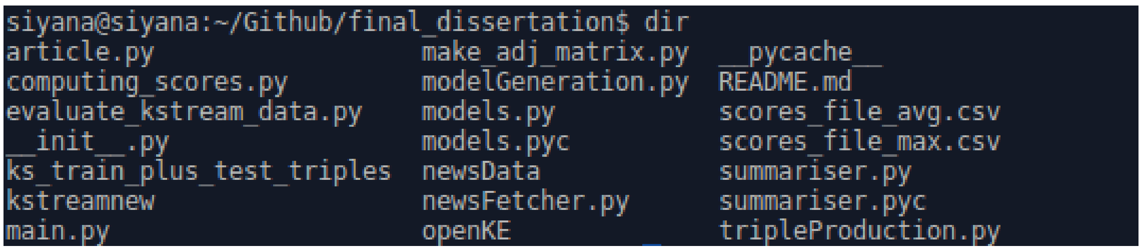
1. Open a terminal on your machine.

2. Navigate to the main project directory. If you list the contents of the main directory, you should be able to see all the components as shown in Figure A.1.

3. Run the main script with the command python3 main.py. You will now be able to see a menu of the tools available for the purpose of collecting articles, summarising them, extracting knowledge and generating an embedding model. You will be prompted to select the command you with to you, as shown is Figure A.2.

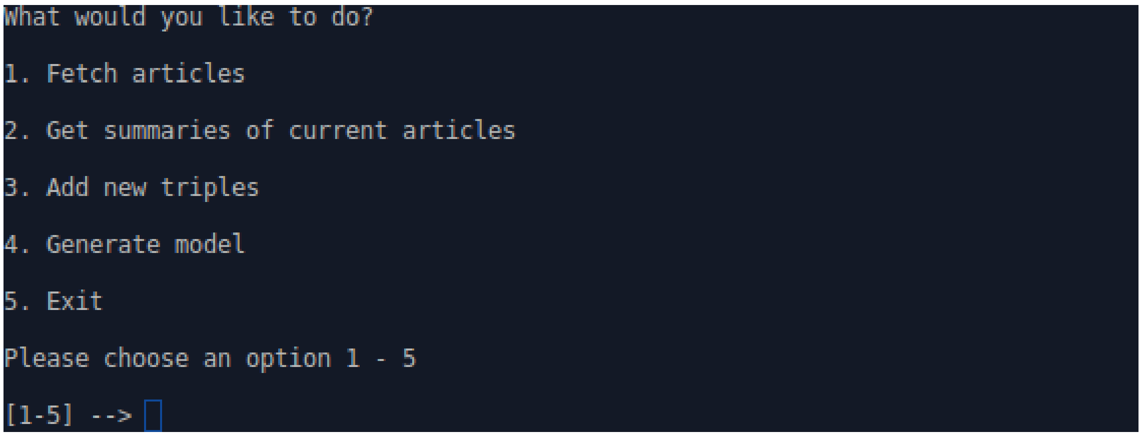
   (a) To let the system automatically collect articles from the four sources, choose 1. The system will start collecting articles and uploading the important information from them to a database

   (b) To create summaries for the articles which do no have the, choose 2. The system will



```
siyana@siyana:~/Github/final_dissertation$ dir
article.py                   make_adj_matrix.py  __pycache__
computing_scores.py          modelGeneration.py  README.md
evaluate_kstream_data.py     models.py           scores_file_avg.csv
__init__.py                  models.pyc          scores_file_max.csv
ks_train_plus_test_triples   newsData            summariser.py
kstreamnew                   newsFetcher.py      summariser.pyc
main.py                      openKE              tripleProduction.py
```

Figure A.1: Project Directory



```
What would you like to do?

1. Fetch articles

2. Get summaries of current articles

3. Add new triples

4. Generate model

5. Exit

Please choose an option 1 - 5

[1-5] -->
```

Figure A.2: Main Menu

find all the article in the database which do not have summaries, will prepare them and will update that in the database.

(c) To extract knowledge from the newest articles, choose 3. The system will find which are the articles from which knowledge has not been extracted yet and will extract them. The new triples extracted will be added to newsData/textTriples.csv.

(d) To generate a model from all the triples that have been extracted so far, choose 4. The system will train an embedding model on the triples contained in newsData/textTriples.csv. The new model will be saved in openKE/res/embedding.vec.json.

A.3 Running Tests with Knowledge Stream

- If you want to run more tests on the Knowledge Stream scores, simply run python3 evaluate_kstream_data.py. This will run a classifier on the data and let you know what the accuracy, precision, recall and F-measure of the testing are.

- If you have your own score files and want to run tests on them, simply replace the scores which are ready for evaluation, simply replace the scores_file_avg.pr and scores_file_max.py files in the main directory with your own files and run python3 evaluate_kstream_data.py

- If you have CSV files for another set of news articles but no score files yet, which you would like to test, replace the files in /kstreamnew/output/400pos with any number of CSV files produced from true articles, the the files in /kstreamnew/output/400neg with any number of CSV files produced from fake articles. Then run python3 computing_scores.py from the main project directory. This will update the scores_file_avg.pr and scores_file_max.py files. Finally, run python3 evaluate_kstream_data.py to test the data.

  Note: The CSV files you put in /kstreamnew/output/400pos and /kstreamnew/output/400neg need contain the scores for each triple contained in each files which means they need to have been run through the Knowledge Stream software[1].

[1]https://github.com/shiralkarprashant/knowledgestream

Appendix B

Maintenance Manual

B.1 System Requirements

The system is designed to run on 64-bit Linux/Ubuntu operating systems. Different components of the system may require up to 15GB RAM, therefore a machine with higher memory is needed.

B.2 Dependencies

The following components need to be installed before the system can run:

- Python 2.7 or higher 2.x version
- Python 3.5 or higher 3.x version
- OpenKE[1] - no need to download this as it comes as a part of the project files
- KnowledgeStream[2] - no need to download this either as it comes as a part of the project files too. However, you will need to download the original KnowledgeStream test data[3] if you intend to use it.
- Stanford CoreNLP[4]
- MongoDB[5]
- Python packages
  - pip and pip3 or easy_install - for installing new packages
  - pandas[6]
  - numpy[7]
  - sklearn[8]
  - requests[9]

[1]https://github.com/thunlp/OpenKE
[2]https://github.com/shiralkarprashant/knowledgestream
[3]http://carl.cs.indiana.edu/data/fact-checking/data.zip
[4]https://stanfordnlp.github.io/CoreNLP/download.html
[5]https://www.mongodb.com/
[6]https://pandas.pydata.org/
[7]http://www.numpy.org/
[8]http://scikit-learn.org/stable/
[9]http://docs.python-requests.org/en/master/

- bs4[10]
- pymongo[11]
- nltk[12]
- pycorenlp[13]
- neuralcoref[14]

B.3 Running the System

In order to run the system, you need to have a number of tools running in the background which the system is dependent on. The steps to start the system are as follows:

1. Start your mongoDB instance with parameters *mongod –nojournal –dbpath /path/to/mongo* replacing */path/to/mongo* with the path where you have installed mongod

2. Navigate to the folder where you have installed CoreNLP. Start the CoreNLP server with the command

   java -mx8g -cp "" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -annotators "tokenize,ssplit,pos,lemma,parse,sentiment" -port 9000 -timeout 30000*

3. Navigate to the main project folder and start the system with *python3 main.py*

[10]https://www.crummy.com/software/BeautifulSoup/
[11]https://api.mongodb.com/python/current/
[12]https://www.nltk.org/
[13]https://github.com/smilli/py-corenlp
[14]https://github.com/huggingface/neuralcoref