Université de Montréal

Content-Based Automatic Fact Checking

Par

Téo Orthlieb

Département d'Informatique et de Recherche Opérationnelle
Faculté des Arts et des Sciences

Mémoire présenté en vue de l'obtention du grade de
Maitre ès sciences (M.Sc.)
En Informatique, option Apprentissage Machine

31 Décembre 2021

Université de Montréal

Faculté des études supérieures et postdoctorales

This Msc. Thesis named

Content-Based Automatic Fact Checking

Presented By

Teo Orthlieb

Was evaluated by a jury composed of the following :

Miklós Csürös
(jury president)

Claude Frasson
(research supervisor)

Philippe Langlais
(jury member)

Abstract

The spreading of fake news on social media has become a concern in recent years. Notably, [Sha+16] found that fact checking generally takes 10 to 20 hours to respond to a fake news, and that there is one order of magnitude more fake news than fact checking. Automatic fact checking could help by accelerating human work and monitoring trends in fake news. In the effort against disinformation, we summarize content-based automatic fact-checking into 3 approaches: models with *no external knowledge*, models with a *Knowledge Graph* and models with a *Knowledge Base*. In order to make Automatic Fact Checking more accessible, we present for each approach an effective architecture with memory footprint in mind and also discuss how they can be applied to make use of their different characteristics. We notably rely on distilled version of the BERT language model [Jia+19], combined with hard parameter sharing [Car93] on two approaches to lower memory usage while preserving the accuracy.

Keywords: Fake news; Automatic Fact-Checking; Transformers

Résumé

La diffusion des *Fake News* sur les réseaux sociaux est devenue un problème central ces dernières années. Notamment, [Sha+16] rapporte que les efforts de fact checking prennent généralement 10 à 20 heures pour répondre à une *fake news*, et qu'il y a un ordre de magnitude en plus de *fake news* que de *fact checking*. Le *fact checking* automatique pourrait aider en accélérant le travail humain et en surveillant les tendances dans les *fake news*. Dans un effort contre la désinformation, nous résumons le domaine de Fact Checking Automatique basé sur le contenu en 3 approches: les modèles avec *aucune connaissances externes*, les modèles avec un *Graphe de Connaissance* et les modèles avec une *Base de Connaissance*. Afin de rendre le *Fact Checking* Automatique plus accessible, nous présentons pour chaque approche une architecture efficace avec le poids en mémoire comme préoccupation, nous discutons aussi de comment chaque approche peut être appliquée pour faire usage au mieux de leur charactéristiques. Nous nous appuyons notamment sur la version distillée du modèle de langue BERT [Jia+19], combiné avec un partage fort des poids [Car93] sur 2 approches pour baisser l'usage mémoire en préservant la précision.

Mots Clés: Fake news; Fact-Checking Automatique; Transformers

Contents

List of Tables

List of Figures

List of Acronyms & Abbreviations

API	Application Programming Interface
BoW	Bag of Words
GloVe	Global Vectors for Word Representation
GPU	Graphics Processing Unit
KB	Knowledge Base
KG	Knowledge Graph
LM	Language Model
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
NLP	Natural Language Processing
NN	Neural Network
PRA	Path Ranking Algorithm
RNN	Recurrent Neural Network

Acknowledgements

The completion of this study could not have been possible without the invaluable guidance of my research supervisor, Prof. Claude Frasson. I would like to express my sincere gratitude to him for all his support, empathy and motivation. His advice carried me through all the stages of my research and writing of this thesis.

In addition, a debt of gratitude is owed to Dr. Hamdi Ben Abdessalem, for his keen interest on me at every stage of my research. I'm truly inspired by his dynamism, kindness and patience in helping me with different aspects of my research.

We acknowledge NSERC-CRD (National Science and Engineering Research Council Cooperative Research Development), Prompt, and BMU (Beam Me Up) for funding this work.

Chapter 1

Introduction

1.1 Motivation

The problem of fake news has received more and more attention in recent years. Notably, the act of purposely misleading public opinion with disinformation across social media and newspapers has become a concern in political elections, undermining the trust in reputable media and harming democracy.

In order to keep up with the growing amount of fake news and hoaxes, automatic fact checking could prove useful, as a tool to speed up the work of journalist, or in a monitoring system to warn regular users.

In this introduction we first discuss the different kinds of fake news, present the human resources for fact checking and identify 4 categories of automatic fact checking tools, discuss their characteristic, and justify our choice to focus on content-based fact checking.

1.2 Type of Fake News

There are many types of fake news, all with different kind of sources, motives and writing style. [RCC15] identifies 3 broad categories:

- Serious Fabrications (Clickbait): fraudulent reporting found in tabloids, they exaggerate and rely on sensationalism to increase revenue, but they are not picked up by mainstream audience or media.
- Large-Scale Hoaxes: attempts to deceive the mainstream audience, typically not made for profit and can cause physical or financial harm. Political propaganda as well as some conspiracy theories fall in that category.
- Humorous Fakes: Satire news, such as The Onion, can sometimes be accidentally repeated by mainstream media, but they do not harbor intention of profit or harm.

Out of these 3 categories, Large-Scale Hoaxes is arguably the most problematic, as well as the hardest to detect automatically. This is because Satire articles are easily

identifiable by their sources, and Clickbait articles often use very specific writing style and vocabulary. On the other hand, hoaxes often require an understanding of the topic discussed to be debunked, involving expert knowledge and cross-referencing.

1.3 Human Fact Checking

The recent increase of fake-news has given birth to many debunking websites, that identify and break down fake news as they come up. These websites are often maintained by professional journalists, requiring a lot of human work, but they provide a very thorough analysis complete with counter arguments to the user. Some notable examples are:

- Snopes: They cover many topics and try to stay as up to date as possible, providing in depth-analysis of viral fake news.
- Politifacts: Focused on US politics, they developed a *truth-o-meter* to sum up their opinion on a news, as well as a *flip-o-meter* to answer accusations of a politician switching sides on a given topic.
- TruthOrFiction: Focused on viral content in general, including memes and mail scam, they try to rely on open sources as much as possible.
- HoaxEye: Work of an individual to identify images that are either photoshopped or out-of-context for deceptive purposes.

Another movement related to fake news is the skeptic and critical thinking communities, generating educative content on social media and video platform such as YouTube. They are able to reach a large audience, and convey useful principles that can help internet users against hoaxes and fake news.
Examples of such content creators include:

- Captain Disillusion: Produces educational and entertaining videos about viral hoaxes or conspiracies based on edited video. The channel aims at explaining common video editing technique and how to spot them.
- Hygiène Mentale [in french]: Produces educational videos teaching concepts of critical thinking, such as Ockham Razor, Confirmation Bias, etc, helping the viewer to reason rationally about news in general.
- Skeptic: Publish podcast and interviews with authors of books around conspiracies, cults and critical thinking in general. The podcasts are very thorough, lasting between 1 and 2 hours.

1.4 Automatic Fact Checking

On the other hand, automated tools do not require human intervention each time a novel fake news come up, but they typically are not as insightful as debunk articles

written by journalists. We find and describe below 4 main categories of automated tools that help with fact checking [1].

1.4.1 Aggregators

Aggregators are tools that allow the user to search for fact checking articles that already exist for a given news. They rely on pre-existing human fact checking work but are still helpful.
Some example are:

- Google Fact Check Explorer: Allows to search fact check article, and present the result as a series of claims, followed by the verdict of fact checking websites. Each result is associated with related key words that help the user learn more about it, as well as the source of the claim and of the debunking.

- ClaimBuster: ClaimBuster offers 2 services, extracting meaningful claims from a discourse, and looking up facts with existing search engines such as google and wolfram. The claim extraction process is detailed in [HLT15], and uses a Random Forest to classify each sentence as either *Non-Factual*, *Unimportant and Factual* or *Check Worthy Factual*.

Since Aggregator tools rely on pre-existing debunking article, they are not able to answer to a newly created fake news that has not been debunked yet. However they are still useful for users and journalists to find existing Fact Checking work.

In practice, Aggregators are specialized search engines, which we detail in greater length in chapter 4.

1.4.2 Source-Based

Source-Based tools use databases of reliable and unreliable sources or authors in order to assess the reliability of a news. Since they do not evaluate the content of the news, they are fast and easier to implement, but have the downside of not being able to evaluate a news with an unknown source.
Instances of this include:

- BS Detector: A Browser extension that automatically searches for unreliable sources in a page content (against a manually curated database) to assess truthfulness.

- The Factual: A chrome extension that grades the credibility of a page based on 4 sub-scores (also displayed): Site quality, Journalist quality, Quality and diversity of sources, Article's tone. The details behind are not public.

Since source-based fact checking tools simply lookup the source of articles against a database, their implementation is often straightforward and do not require specialised expertise.

[1] Note that some tools combine multiple techniques to automated fact checking

1.4.3 Network-Based

Network-Based tools analyze how a given news is spread on social media to assess its reliability. Similarly to source-based methods, content of the news can be disregarded. These methods can also be effective but only work in a social media setting. We found no finished tools using this, but summarize below some relevant work on this topic.

[Jin+16] Presents a model exploiting Tweet answers and their relations to assert truthfulness of a news event using Conflicting viewpoints mining and Credibility Networks.
Steps of Conflicting viewpoints mining:

1. Gather a dataset of news labeled by topic and viewpoint
2. Train a Naive Bayes Classifier on this dataset
3. For each tweet in the credibility network, compute a Jensen–Shannon divergence between viewpoints distributions and use a constrained k-means to put viewpoints into 2 groups.

Steps to establish the Credibility Network:

1. Start from a single Tweet and give it an initial credibility score in [-1; 1] from a tweet level classifier
2. Search every tweet that it is related to and construct a network with links that are either positive if the 2 tweets have the same viewpoint and negative otherwise (using the Conflicting viewpoint mining)
3. Then for each pair of linked tweets, make their credibility score closer if they agree and farther if they disagree. Repeat until convergence, generating the final credibility score for the news.

[Tac+17] present a new algorithm to classify fake news on Facebook using the likes and dislikes of each user, that works with even a small dataset of labeled Facebook posts. In the training phase the algorithm computes the preference towards hoax for the most users it can, using this method:

1. Characterize each post by the users that liked it
2. Characterize each users by the amount of labeled hoax/non hoax they have liked
3. Alternate 5 times between:
   (a) Use each user to label more posts, a user that likes almost exclusively hoaxes will be a bigger red flag than a user who likes both hoaxes and non hoaxes.

(b) Use each newly labeled posts to update the hoax/non hoax counts of more users

The author note that this method 99% on facebook posts despite only 1% of the posts being flagged initially.

1.4.4 Content-Based

Content-Based tools analyze the content of an article to provide insight on its reliability. These are arguably the most challenging tools to implement as they usually require some understanding of the natural language. The upside of these tools is that they are able to work even on novel fake news with unknown source.
Instances of Content-Based tools include:

- Factmata: analyses a page and report any hate speech, political propaganda or clickbait. The process behind is not public.
- Brenda [BSS20] (demo): browser extension that extracts claims from the page using ULMFiT [HR18], queries Google API to get relevant articles, scrape the content with Newspaper3k and passes the results to SADHAN [MS19] to classify the claim.

Content-based techniques are challenging to implement because they require a certain level of language understanding to analyze the content of news article, but can be powerful as they can adapt to any news even with no source.

We categorize Content-Based Fact Checking models into 3 approaches:

- Models *without knowledge*: after training, these models rely on no external knowledge to assess a given news. Not relying on external data usually makes these models faster, but limits their ability to debunk a news.
- Models with a *Knowledge Graph*: A Knowledge Graph (KG) contains structured knowledge in the form of (entity 1, relation, entity 2). Models that use KG will be able to find links between entities in the text, increasing their ability for reasoning. However, the constrained format of information in a KG makes these models less flexible when understanding natural sentences.
- Models with a *Knowledge Base*: A Knowledge Base (KB) is a collection of unstructured information, such as Wikipedia. As we will see, a KB can be a great tool for a fact-checking model, allowing it to cite credible references while still being flexible.

1.5 Objectives

Out of these 4 categories of automatic fact checking, we feel that Content-Based fact checking methods has the most potential and the most room for progress, because of

their challenging implementation and capacity to function even with unknown source.

In the following chapters, we will take a closer a look at each approach to Content-Based Fact Checking, provide an overview of the existing work and suggest a new model with 2 objectives in mind:

- **Low memory cost**: in order to make fact checking accessible and portable on ordinary computers, low memory cost is important. This justifies the need to look for application of techniques such as Distillation or Parameter Sharing to save space while preserving accuracy as much as possible.
- **Explainability**: in order to make fact checking tools the most useful, they need to provide explanations for their results whenever possible, so that any user can explore the subject in more depth if they want to. This will prompt us to favor certain approaches, which may be more constrained but yield more understandable results.

At the end of each chapter we will also discuss how the 3 approaches can be applied to make use of their specific strengths and weaknesses.

1.6 Language Models

This work uses the recent advances in Deep Learning with Language Models extensively, which we briefly introduce in this section.

Formally, a Language Model is a probability distribution over sequences of words, meaning that for each sequence of words w, it yields $p(w_1, w_2, ..., w_n)$ the probability of w.
For instance, a proper Language Model of English would be able to state that *"Alice is eating a movie"* is a much more unlikely sequence of words than *"Alice is watching a movie"*.

In the field of Natural Language Processing, Neural Networks (NNs) have been used to tackle a variety of tasks: sentiment classification, translation, question answering, etc.

Traditionally, the NNs were trained on each tasks using exclusively specific datasets for the task containing labelled examples, but recent works such as [Rad+18a] demonstrated that using the NN to model the language first before fine-tuning it on specific tasks later was a much more effective strategy, because training a language model only requires natural text, which is abundant in books and on internet.

This technique of building Language Models with Deep Learning before specializing them on downstream tasks gave rise to the modern models derived from the Transformer architecture [Vas+17] such as BERT [Dev+19] and its distilled version TinyBERT [Jia+19].

Because these Language Models have demonstrated good results that could be scaled up as needed, we chose to make use of them in this work as Automatic Fact Checking encompasses many tasks centered around natural language.

Chapter 2

Model without knowledge

2.1 Introduction

A model without external knowledge is the simplest approach to content-based fact checking, as it can be framed as a regular text classification task in natural language processing. In this introduction we discuss Text Classification and its application to Fake news.

2.1.1 Text Classification

Text classification is a well studied problem of which many instances exist already, such as emotion classification, language identification, or spam detection. A prerequisite of text classification is the gathering of a dataset of text and labels, specific to the task, that will then be used to train and evaluate models. Given an input text, a text classification model is expected to output a label, depending on the task.

The first step of text classification is the tokenization, which breaks down the input text into pre-defined tokens, which can subsequently be processed by the model. The tokens belong in a pre-defined vocabulary, and Out Of Vocabulary (OOV) tokens are often represented with a special 'UNK' token. Most model function with words as tokens, but exceptions such as Convolutional Neural Networks (originally from computer vision) can work with individual characters as tokens, which has the advantage of handling even unknown words (since all the characters are known). In order to reduce the variety of words, tokenization processes often include simplifications such as setting the text to lower case or stemming, i.e. dividing the word into a stem with prefixes and suffixes.

A classic model for text classification in statistical learning is the Naive Bayes Classifier as a Bag of Words (BoW) model. In short, Bag of Words models represent each text with the distribution of the tokens forming it, disregarding the order between words. Despite its simplicity, these models are surprisingly effective with the advantage of being fast.

Another important step for text classification is the work on Recurrent Neural Networks, notably Bi-LSTM [GFS05] combined with word2vec tables such as GloVe [PSM14]. These models first translate every token into a meaningful, high

dimensional, vector, with similar words being closer together, and then process them sequentially, carrying the information between each step in a hidden state computed by the model. Contrary to BoW models, RNNs have the capacity to take the orders of words into account, granting deeper understanding of the language. However, they are notoriously bad at transmitting information over long text, due to their sequential nature.

To improve the flow of information, recent research has focused on Transformers [Vas+17], a successful family of neural networks which we use extensively in this work. Instead of processing text sequentially, Transformers process every token by directly looking at every other token in the text, using an attention mechanism that grants a higher weight to relevant tokens. This parallel nature allows for faster treatment of text and better flow of information.

A recent procedure which proved a great boost of accuracy with Transformers is pre-training, as seen in BERT [Dev+19]. This typically consists of training the model to predict the next word on a huge corpus of text before training the model on specific tasks downstream. This initializes the model with some understanding of natural text and does not require any label (it is called *self supervised*).

2.1.2 Objectives

In our case the text to classify is a piece of news (length can vary), and the associated label can be fake/true, or a type of news in some datasets, making it either a binary or a multiclass classification task.

However, not relying on external knowledge makes it difficult for a model to provide justification as to why a given news would be fake or not. Where a journalist might find a scientific article or a credible source to refute a fake news, such a model would not be able to do that, and instead only predict that a news is fake based on "how it looks", using clues such as writing style or topic. On the other hand, not going through the process of querying an external source of knowledge make these kinds of model fast, which is interesting for monitoring applications, processing a large amount of news and attempting to flag the suspicious ones.

In this chapter we first provide an overview of fact checking without external knowledge, present a new simple and lightweight model that can both give a reliability score and identify the type of fake news, and discuss the applications best suited for fact checking models without external knowledge.

2.2 Related Work

Content-based fact checking is a very recent area in general, and relatively little work has been put into approaching it using model with no external knowledge. Still, we describe below a few papers that we find relevant to this approach.

[HLT15] notes that not all sentences in an article or a discourse are worthy of fact checking, they distinguish 3 categories:

- Non-Factual Sentence (NFS): Subjective opinions, feelings and most question falls into that category (eg: *I prefer to get up early; You remember when you said that ?*)
- Unimportant Factual Sentence (UFS): Factual but checking it doesn't have any interest (eg: *I ate lunch at a restaurant 2 days ago*)
- Check-worthy Factual Sentence (CFS): Factual and interesting to the public (eg: *He voted against the 1st Gulf War*)

Indeed, this is an important distinction to keep in mind when building a fake news monitoring tool for example, as we don't want to be flooded with reports of non factual/unimportant sentences. In this work the authors also built a small dataset of 1.5k sentences labeled with these categories from presidential debates, on which they achieved 85% accuracy with a random forest model on hand picked features extracted with AlchemyAPI (from IBM) and NLTK.

[BTB16] focuses on identifying clickbait articles, first gathering a dataset of 4k articles labeled as either clickbait or non-clickbait, they use gradient boosted decision trees with hand picked features to reach 77% accuracy. The authors identify important features for finding clickbait articles, such as informality measures (cf [PT16]) and forward references (eg: *"This Is the Most Horrifying Cheating Story"*). While it is interesting to note such features, new deep learning models typically do not need hand picked features to function, as they are capable of finding their own.

[Zel+19] tackles the slightly different problem of identifying AI generated Fake-News. Indeed, not all fake news are written by humans, and while text generation models such as GPT-2 [Rad+18b] are far from perfect, they are still capable of generating some convincing samples that could fool humans. The authors present GROVER, a GPT-2 style Transformer, able to generate convincing fake news, and subsequently show that GROVER is also the most accurate model to detect its own fake news (with 92% accuracy), underlying the importance of disclosing model architecture publicly.

In the section below, we present an efficient model to identify multiple type of fake news and give an overall score of how "fake" the news is between *[0, 1]*, using recent advances in distillation in combination with parameters sharing.

2.3 Datasets

In order to train and evaluate our model, we identify 2 fake news datasets that we use:

- FakeNewsCorpus: This corpus includes a total of 9+ millions labeled news article. 1 label is attributed to reliable news, and 9 other labels help differentiate between the type of fake news, such as *"Conspiracy"*, *"Satire"*, *"Clickbait"*, etc... The dataset has been scrapped automatically using a database of both reliable and fake news website, each with a label. This dataset has a lot of data

and the multiple categories of fake news allow us to build a more fine-grained classifier.

- FakeNews Kaggle: A dataset of about 20k fake news, labeled with either true or false. Since the dataset is present on Kaggle (a machine learning competition site), we observe a leaderboard with top scores around 98% accuracy, an easier dataset comparatively.

- Liar [Wan17]: A dataset of about 10k statements from US politics, with information on the speaker, the context and a justification. The dataset was built with Politifacts API and uses the same 6 categories of statements: *pants on fire, false, barely-true, half-true, mostly true and true*. Due to the similarity between the labels this is a very difficult dataset as we will see.

While these datasets include meta information such as authors, newspaper, or tags that would definitely help classifying articles, we decide to ignore this information and only use the content of the news article itself. This decision is motivated by the objective of building *content-based* fact checking models, so using additional information not present in the content of the article would be unfair.

2.4 Model

2.4.1 Architecture

We note that when predicting the type of fake news with FakeNewsCorpus or predicting if a news is fake or not with FakeNews Kaggle the input is the same (a piece of news) and only the output differs (multiple classes vs a probability). We thus apply hard parameter sharing and use 1 model for both task, only changing the final layer depending on the task.
We use TinyBERT_4 [Jia+19] as our shared model because of its efficiency, getting 96.8% of BERT [Dev+19] performance while being 7.5x smaller.
The detailed procedure is as follow:

- Prepend a special '[CLS]' token to the input news that will hold the representation of the whole news

- Pass the tokenized news to TinyBERT_4

- Retrieve the last hidden representation of the '[CLS]' token, and pass it through the corresponding forward layer for the task, yielding either a type of fake news or a probability of the news being true

This architecture is very simple but obtains good performance by relying on modern deep learning developments.

2.4.2 Catastrophic Forgetting

A central problem when training a single model on multiple tasks is Catastrophic Forgetting. In short, training a model on a task A and then on a task B carries the risk of the model "forgetting" task A, and getting worse at it.

Various strategies have been researched to counter this effect, which we briefly discuss:

- Orthogonality: Orthogonality techniques consist of pushing parts that handle different tasks to be orthogonal (linearly independent), thus reducing interference between different tasks and the risk of catastrophic forgetting. A recent example of orthogonality is [Che+19] which evaluate a method of using task specific keys to rotate the weights so that they are roughly orthogonal.

- Pre-training: Pre-training is mostly cited as a technique to boost model performance by pre-training the model on a general task of the same domain using unlabeled data, available in greater quantities than labeled data. However pre-training has also been shown to be an effective technique to prevent catastrophic forgetting [MH93]. Similarly to how humans use prior knowledge even on completely new tasks, it seems that the knowledge gained in pre-training constrains how the knowledge gained on new tasks is incorporated.

- Rehearsal: rehearsal techniques tackle the problem of catastrophic forgetting by retraining the model on previously learned tasks. This can be done simply by storing old task examples but has the downside of accumulating a lot of data over time. To overcome this, *pseudo-rehearsal* techniques such as [Rob95] instead generate estimations of inputs for old tasks and rehearse on these, removing the need to access old examples.

In this work, we implement a kind of Rehearsal method, that consists of simply mixing batches of both tasks so that they are evenly distributed in a unified training dataset (code in Appendix section .1). This has the advantage of not requiring extra work from the model, and has no extra computational or memory cost. The efficiency of this method is evaluated in the Results section 2.5.

2.5 Results

2.5.1 Evaluation

To evaluate the model's performance, we display the test accuracy for the following in Table 2.1:

- *Majority Baseline*: predicts the most common labels for each dataset, this is the highest score that is obtainable without looking at the data.

- *Best Known*: Regroups the best known accuracy scores for each dataset at the time of writing. For the Kaggle dataset the leaderboard present on the site

was used, and for the Liar dataset, the best score obtained with only text data was used [Wan17]. Unfortunately, we could not find another model tested on the Corpus dataset, but we display our performance to serve as a reference for future work.

- *Individual Models*: displays the performance of a TinyBERT$_4$ trained on each dataset independently, with no parameter sharing.
- *Parameter Sharing*: the performance of TinyBERT$_4$ with hard parameter sharing applied on Kaggle + Corpus and Kaggle + Liar, using the batch mixing scheme described in subsection 2.4.2.

	Kaggle	Corpus	Liar
Majority Baseline	55.0%	28.6%	20.4%
Best Known	98.6%	-	26.0%
Individual Models	99.0 %	79.7%	26.1%
Parameter Sharing	**99.5%**	**81.1%**	**26.7%**

Table 2.1: Performance comparison of the model with and without parameter sharing, against majority baseline and best known accuracy.

We note that parameter sharing was effective, not only did it group what would have been 2 models into 1, saving almost half of the memory, it also offered a slight performance boost. We also tried separating the final layer of TinyBERT$_4$ for each task but it did not improve the results, suggesting that the boost of data was more beneficial to the training that the cost of doing 2 tasks at once, which was to be expected since the 2 pairs of datasets were so close in nature.

Additionally, we also evaluated our Batch Mixing scheme by training the same model on the Kaggle dataset first and then the Corpus dataset to observe the impact of Catastrophic Forgetting on the first dataset. Indeed, despite the Kaggle Dataset being easy in comparison, training this way resulted in a test accuracy of 97.6% on Kaggle vs the 99.5% obtained with Batch Mixing. This is a very positive result as Batch Mixing does not require any additional computation as opposed to most Catastrophic Forgetting mitigation techniques.

As previously mentioned, we note that the Liar dataset is very difficult in practice, most likely because of how close the different classes are. Overall we observe that this light and simple model obtains competitive results on automatic fact checking with no external knowledge, and is capable of providing a reliability score as well as a detailed fake news type, when trained on Kaggle and Corpus with parameter sharing and batch mixing.

2.5.2 Type of Fake News in Corpus Dataset

We look into the confusion matrix obtained on the Corpus dataset after training with parameter sharing and batch mixing. The confusion matrix displays the Ground Truth on the X axis vs the Model Prediction on the Y axis.

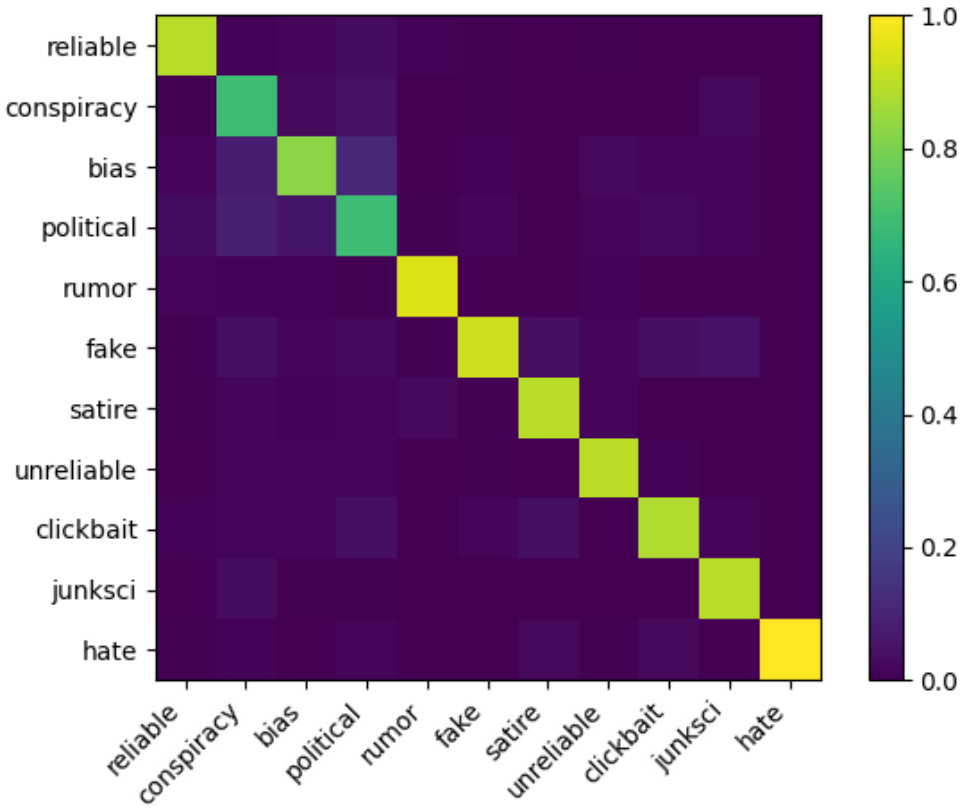

Figure 2.1: Confusion Matrix for the type of fake news

At first glance, we observe that the hardest class of fake news to identify is *Political*. This is not surprising due to the broad nature of "political news", and we note that it is often confused with *Bias*, which is explained by the proximity of the 2 classes. It's interesting to note that the model tends to mislabel *Political* news and *Junk Science* news as *Conspiracy* equally often but *Political* and *Junk Science* are not confused together, suggesting that the model might see conspiracy news as something between political news and junk science, which is understandable.
It seems that the performance of the model would increase greatly by either merging *Bias* and *Political* news together or attributing more resources to separating them, but otherwise the classes are all decently well separated.

2.5.3 Explainability

As stated in the section 1.5, being able to give a justification associated with the predictions of the model is important for the end user in the context of fact checking. Unfortunately, fact checking models with no external knowledge are particularly ill suited to provide a justification since they don't rely on any explicit form of knowledge.
Despite that, we can still obtain some information about the prediction by probing the model. Since our model is a kind of Transformer Encoder, information about the sentence is carried in a special vector associated with the '[CLS]' token. Within the model, this vector is repeatedly updated with the other vector representations of the tokens in the sentence, using an attention score to attribute variable importance to each tokens. Thus, by reading the attention score between the '[CLS]' token and the rest at the last layer of the model, we can get a sense of how important the words in the sentence were to classify a piece of news.
To display this information to the end user, words could be colored by how high was their score, as illustrated below on sentence that would be labelled as conspiracy:

Covid vaccine is a micro-chip delivery tool

As we can see, 'vaccine' and 'micro-chip' were impactful words for the classification of this sentence as conspiracy.
With some refinement, this display could be particularly useful, when classifying long texts, to highlight the parts that were most important for the model to label the text as reliable or fake-news of some kind.

2.6 Discussion

In this chapter we summarized Automatic Content-Based Fact Checking with no external Knowledge, re-framed as a Text Classification task. We suggested a lightweight model that could both identify the type of a fake news and give a reliability score by combining TinyBERT_4, Parameter Sharing and Batch Mixing, a training scheme to mitigate Catastrophic Forgetting with no additional costs.

2.6.1 Application

By not using external Knowledge the model can be fast, and because it is easily parallelizable, it is orders of magnitude faster than the models we will present in the following chapters on GPU. Also, because it relies on superficial cues instead of knowledge, it should be more robust to detect fake-news on which no explicit knowledge is available to the other models. However, by not relying on external Knowledge for fact checking, this model has little ability to provide a justification to the user. Where a journalist might refer to scientific publications or expert testimony to debunk a fake news, this model cannot. We detailed how some insight could be given to the end user by inspecting the attention scores in the last encoding layer

to understand which words had the most impact in the assessment given to a news. While this could not be called a "justification" it can still inform the end user as to which words or formulation are indicative of the type of news according to the model.

Taking in consideration its speed, robustness to novel fake-news and the limited capacity of highlighting important words or passages, we suggest that this category of model is best suited for monitoring purposes, were it can do the preliminary work of churning to a large amount of post on social medias and news platform, reporting suspicious ones with highlighted passages.

2.6.2 Future Work

In this experiment, simply mixing the batches of every task evenly when training was enough to overcome Catastrophic Forgetting, but it would be interesting to test the limit of this method. Is Batch Mixing still effective as the number of tasks grows ? Is it still effective if there is much more data on one task than the other ? These are some questions that could be investigated by further research.

In succeeding works on Automatic Fact Checking, the Corpus Dataset would also need to be more tested upon, as it contains a large amount of examples compared to most fake news dataset, and its fine grained classification of types of fake news defines an important task in fact checking.
However, some caution needs to be taken about results obtained on datasets where the label of the news was assigned based on the newspaper it was published in, as other biases from specific writing style could seep in and change the nature of the task. The Liar dataset has the advantage of being 100% human labelled so it should be more reliable on this aspect, but on the other hand it also suffers from labels that are too close to each other, such as *barely-true* and *half-true*. An interesting direction for future fake news dataset would be to build a human labelled dataset that distinguishes the various type of fake news like the Corpus dataset does.

Chapter 3

Model with Knowledge Graph

3.1 Introduction

A fact checking model with Knowledge Graphs (KG) uses the structured information presents in KG to perform deductions on a given news. In this introduction we introduce Knowledge Graph in detail, and discuss their application to fact checking.

3.1.1 Knowledge Graphs

Formally, a KG is a directed graph with labeled edges, where each node represents an entity, and each edge a relation. The aim of a KG is to store information about the world (or sometime a specific domain) in this structured manner, with every fact being a triplet of *(head entity, relation, tail entity)* or *(h, r, t)* for short. These triples can be alternatively written as *(subject, predicate, object)* and called a RDF/Semantic Triple.
For example, the date of birth of Donald Knuth could be stored in a KG as *(Donald Knuth, birthday, 1938-01-10)*, with *Donald Knuth* as the *head entity*, *birthday* as the *relation*, *1938-01-10* as the *tail entity*.
Of course, storing information in this constrained way presents several challenges described below, each an active area of research.

Triplet Extraction

The task of extracting semantic triples from a natural sentence, an essential step to build or to interact with a Knowledge Graph. For instance, from the sentence *"Barack Obama is an American politician and attorney"*, we could extract semantic triples such as: *(Barack Obama, nationality, USA)*, *(Barack Obama, occupation, Politician)*, or *(Barack Obama, occupation, attorney)*.
The first approach to triplet extraction, is to use a classic tree parser, able to break down a sentence into grammatical components, and then apply a rule based algorithm on the parse tree to extract fact triplets [Rus+07]. More recently, deep learning

approaches to triplet extraction has been suggested, such as [WLS20b], in which a Transformer is used to extract semantic triples from natural sentences directly.

Named Entity Extraction

Tightly related to Triplet Extraction, *Named Entity Extraction* is the task of identifying and linking entities in natural sentences to known entities in the Knowledge Graph, often broken down into Recognition (NER), Disambiguation (NED) and Linking (NEL). Similarly to triplet extraction, the classical approach is a mix of rule based and statistical learning systems, and deep learning is also starting to be applied here, as described in details in [Al-+20].

Link Completion/Prediction

Link Completion/Prediction is the task of predicting whether a new triple is true or not given a Knowledge Graph, or alternatively predicting t given *(h, r, ?)*. This is a common task which is very relevant to fact checking; indeed, if we store knowledge of the world in a Knowledge Graph, we can re-frame fact checking as a link completion task by representing facts as semantic triples and predicting whether they are true or not given the KG. Link Completion is a very active area of work on Knowledge Graphs, and will be described in more details in the next section.

3.1.2 Objectives

If Knowledge Graphs are still popular despite the challenges described above, it is because storing information in a KG makes it subsequently easy to process with computers, even without machine learning at all. A fact checking model making use of KGs can thus benefit from this, provided that it is able to fulfill the 3 challenges.

In this Chapter we will first provide an overview of the 2 main approaches to Link Completion/Prediction, justify which one best suits the fact checking problem, and describe a complete pipeline for fact checking with Knowledge Graphs, including a novel model for Link Prediction as well as a new dataset. We will conclude by discussing the applications of fact checking models with KGs.

3.2 Related Work

The main challenge we focus on in this section is Link Completion/Prediction, for which we distinguish 2 main approaches.

3.2.1 Knowledge Graph Embedding

Arguably the most popular approach to link completion, *Knowledge Graph Embedding* methods assign a position to each entities (nodes) and relations (edges) of the KG, often in a high dimensional vector space, and then applies a scoring function on

the positions of entities and relations to assert if a given triplet *(h, r, t)* is true. A prime example of this approach is TransE [Bor+13], in which embeddings are computed for entities and relations such that $h + r \approx t$ [1]. The benefit of KG embeddings is that once the embeddings are computed, querying the model is usually very fast, since it is just a matter of computing the score function on a few vectors. However they are also quite opaque, in that it is difficult to provide any justification for their results. When a triplet *(h, r, t)* is ruled as false by a KG embedding model because it had a bad score given its embedding it doesn't mean much for a human, which is a problem in our application where we want to give the user useful information when possible.

3.2.2 Path-Based Methods

Path-Based methods on the other hand exploit paths between entities in the Knowledge Graph to produce an answer. A foundational work in the application of Path-Based methods to Knowledge Graphs is the Path Ranking Algorithm adapted to KGs, PRA for short [LMC11], which roughly works like so:

- Training: For each relation r, and for each known triple *(h, r, t)* with r, PRA looks for alternative paths that connect h, and t. Some paths will come up more often than other, because they are correlated with the relation r.
  For instance, if we are looking at the relation *nationality*, we may observe that a lot of time a person is born in the same country they are citizen of, so the path *birthplace, country* may come up very often. PRA will thus score each path by how strongly they predict the relation r.

- Prediction: For a unseen triple *(h, r, t)*, PRA will try to run paths that predicted well the relation r starting from the head entity h. If these paths consistently reach the tail entity t, then we know *(h, r, t)* is probably true. Of course this require for paths to exists in the KG between h and t.

Contrary to Knowledge Graph Embeddings, Path-Based method like PRA can thus produce some form of interpretable reasoning when assessing a semantic triple, because they rely on other relations in the KG to give an answer, which we find preferable for fact checking applications. Another advantage of Path Based methods is that as opposed to Graph Embedding methods there is no need to recompute an embedding for the KG when adding new entities and relations to the graph, which is a particularly common need in fact checking applications, since knowledge about the world keeps evolving. On the other hand, Path-Based methods are slower than Graph Embedding methods when assessing a fact triplet because of the need to find paths between the head entity and the tail entity, and since they rely on paths they are more effective on denser KGs with many connections between entities. We find this trade-off acceptable and thus choose to focus on Path-Based methods, of which

[1] On its own this would collapse every embedding to 0 (because $0 + 0 = 0$) so incorrect triplets *(h, r, t')* have to be generated and another cost function is used to maximize the distance between $h + r$ and t'.

we present relevant works below.

[Cia+15] presents a simple approach to fact checking based on "infoboxes". In Wikipedia, infoboxes are boxes that summarize information about an entity in a format compatible with Knowledge Graphs. The authors observe that a given fact triplet *(h, r, t)* is more likely to be true if there is a short distance between h and t in the KG generated with infoboxes. While this heuristic completely disregards the relation r between the entities h and t, the author found that it can work surprisingly well. For a given triplet *(h, r, t)* they find the shortest path between h and t in the Knowledge Graph using a special distance function, and if the distance exceeds a threshold t then the triplet is deemed to be false. The authors found that the distance function that works best is to assign more distance when going through an entity that has a lot of connection. The reasoning is that such an entity is very general and a path that goes through it is unlikely to be meaningful.

[XHW18] instead frames the task of path ranking as a Reinforcement Learning task, with the model DeepPath. For a given (h, r, t) the agent is tasked to find the most informative paths from h to t. The state of the agent is described as a position in an abstract embedding representing the entities (what we could get from TransE), and the action it can take is jumping on a neighbor entity, a path can thus be defined as a sequence of actions. To ensure good generalisation, the authors use 3 reward functions, the global accuracy (+1 if the agent reaches the target, -1 otherwise), path efficiency (favoring shorter paths) and path diversity (encouraging the agent to try new paths). The policy network mapping the current state to a probability distribution over all possible actions is a simple feedforward network with ReLu activations. DeepPath has the benefit of considering entities on top of paths (which PRA do not), as well as having better control on the path explored, and result in a slight improvement over PRA.

[Liu+19a] takes another approach to path ranking by incorporating type hierarchies. The reasoning is that for any entity, different level of abstractions can be relevant. For instance, we can think of a Fork as Cutlery, Tableware or simply Ware. Given a head and tail entity h and t a set of all paths (up to depth) connecting h and t are found, then the model returns the probability of any relation r connecting h and t (which is similar to predicting t given h and r). Every path is then encoded with two LSTM, the first encoding the associated meta-path, with the output being the first hidden state of the second which encodes the entities in the path using attention to select the appropriate entity type. The outputs of both LSTM are merged to yield the encoding of the path, and a representation of r is built using the encoded paths as context in an attention module. This representation is then fed through a feedforward network to predict the likelihood of r connecting h and t. This model has the advantage of reasoning over all the path collectively but obtains roughly the same performance as DeepPath. This model also has the disadvantage of requiring information on type hierarchy which isn't available on every dataset.

In our approach, we try to take advantage of the information contained in the labels of entities and relations, with natural language understanding, to build a

simple and efficient model.

3.3 Datasets

Of the 3 tasks of Fact Checking with Knowledge Graphs, *Triplet Extraction*, *Named Entity Extraction* and *Link Completion/Prediction*, we only use datasets on *Link Prediction*, in order to train a model.
As mentioned in the introduction, the task of Link Prediction is to take as input a fact triplet *(h, r, t)* and to output between *[0, 1]* the probability of the triplet being true. Since we chose the Path-Based framework the model is also provided with every alternative paths between the head and tail entity up to a certain size.
We use the following datasets for this task:

- FB15k-237 [Det+18]: FB15k-237 offers ~250k training examples in the form of *(head entity, relation, tail entity)*, extracted from FreeBase. The dataset include "multi-hop relations", meaning head and tail can be separated by more than 1 relation. FB15k-237 is a well known dataset in Link Completion but does not include negative examples, so we generate our own as described in the next part.

- WikiData: WikiData is a general Knowledge Graph with ~100 millions entries. We trim it down to 10 millions training examples and clean it to have more meaningful relations. A controlled trimming allows for a denser Knowledge Graph, with more paths between entities (link to the code).

In addition to the FB15k dataset, the WN18RR dataset [Det+18] is also very common as a Knowledge Graph Link Completion benchmark. However, as illustrated by Figure 3.1 and Figure 3.2, the entities in test examples of WN18RR are much farther apart, making this dataset impractical for Path Based Link Completion approaches.

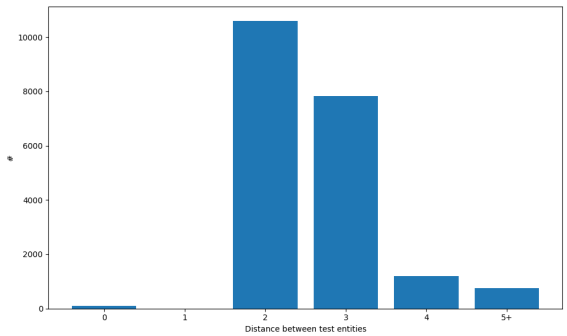

Figure 3.1: Distances in FB15k-237

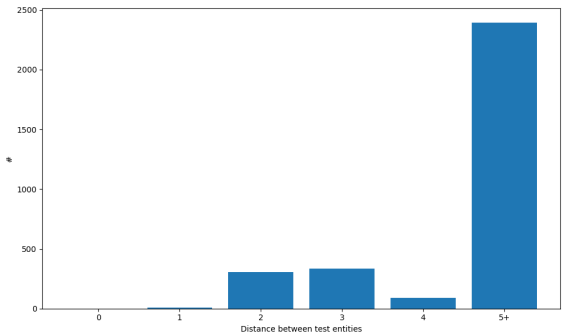

Figure 3.2: Distances in WN18RR

These figures show that a Path Based approach to Link Completion with a maximum distance of 4 (which we chose to limit the computations) will not be able to establish a link between 2 entities in WN18RR most of the time, whereas the same

approach will be able to do so in FB15k-237.
Additionally, WN18RR was created from WordNet, a lexical database for the English language, and as such it does not describe facts about the world, which we deem too far from our application.

3.3.1 Negative Example Generation

Most datasets for Link Prediction do not include negative examples (fake facts) so we need to generate our own. A simple way to generate negative examples is to randomly select a tail entity *t'* given a real head *h* and relation *r*, yielding the negative example *(h, r, t')*. However, this method tend to generate incoherent negative examples, such as *(Bill Gates, born in, Star Wars II: Attack of the Clones)*, because the tail entity is randomly chosen. Additionally, there is a high chance that no short path connect the head and the fake tail entity, which is a problem with Path-Based approaches to Link Prediction.
To overcome these issues we select fake tail entities that are compatible with the real relation *r* and not too far from *h*. This method gives more coherent fake fact such as *(Bill Gates, born in, London)* (link to the code). We use it to generate 5 fake examples for each positive one, for both FB15k-237 and WikiData.

3.4 Model

3.3.1 Triplet Extraction

To extract fact triplets *(h, r, t)* from natural sentences we rely on the library spaCy to construct a simple algorithm to extract triplets.
This algorithm is based on the identification of the parse trees produced by spaCy, which attributes a type to every token and establish their relations as illustrated in Figure 3.3.

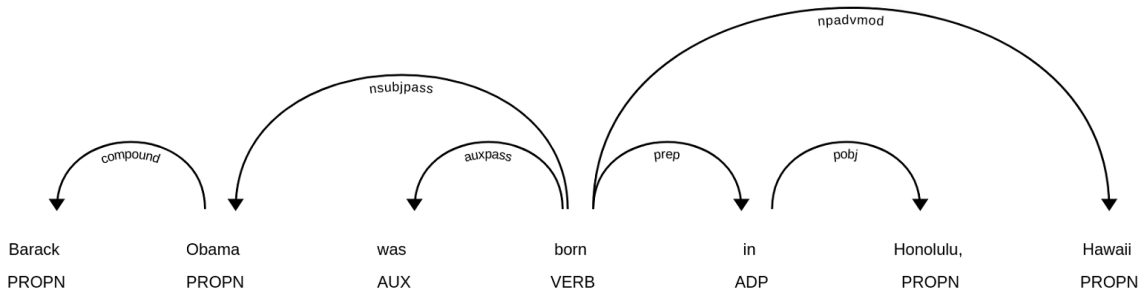

Figure 3.3: A parse tree produced by spaCy

Using this information, the algorithm to extract triplets follows these steps (code in Appendix section .2):

1. Iterate over the tokens to list every subjects by checking if the token is at the end of a relation in *{nsubj, nsubjpass, csubj, csubjpass}* in the parse tree. In the example the only subject would be *"Obama"*.

2. For every subject:

   (a) Reconstitute the whole subject by following the relations in *{amod, compound, cc, conj, prep, pobj}* of the parse tree. In the example this would yield *"Barack Obama"*.

   (b) The root of the relation, *"born"* is the head of our subject (*"Obama"*). From the root of the relation, get every relations by following the links with *{prep, agent}* ending with *{attr, pobj, dobj}* where the target of the relation is. In our case this would give the relation *"born in"* and the target *"Honolulu"*.

   (c) Reconstitute the whole target with the same method as with the subject, and register the *(subject, relation, target)* as a fact triplet.

In the example illustrated with Figure 3.3 this algorithm would yield 1 fact triplet: *(Barack Obama, born in, Honolulu)*.
Despite being quite simple, this algorithm is effective enough and does not require any training.

3.4.2 Named Entity Recognition

In principle Named Entity Recognition should be able to link words to registered entity in the Knowledge Graph. For example "apple" could be linked to the company or the fruit depending on the context in which it is used. Since there is little to no ambiguities between the named entities in the datasets we use, we do not require elaborate Named Entity Recognition in this work. Therefore, we implement a matching algorithm that identifies the entities h and t in the triplets *(h, r, t)* to the entities in the Knowledge Graph that have the best similarity score using the Fast Passage Retrieval method of subsection 4.4.1, disregarding contextual information.

3.4.3 Link Prediction

In most Knowledge Graphs, entities and relations are represented by a unique ID (often an integer) for efficient storage on computer, as well as an English label (not necessarily unique) to allow for data visualization that is understandable by humans. Traditionally, link prediction models ignore the English labels entirely, and build a representation of entities and relations purely based on the structure of the Knowledge Graph.
While this has the advantage of being independent of language, it also ignores the meaningful information about the entities and relations that is contained in their names. This label information is also particularly useful as a fallback when an entity or relation is rare or not present in the KG.

With this model, which is an improvement of our previous version in [OAF21] (Appendix section .4), we incorporate information contained in labels first to build a representation of the entities and relations as we describe below.
The architecture of our model for Link Prediction is parametrized by n the embedding size, and d the maximum depth of the paths. The default values are $n = 128$ and $d = 4$.

1. Given an input triplet *(h, r, t)* get most paths up to depth d connecting h and t in the Knowledge Graph. The KG is built before training the model and the search is done with random walks. Random Walks is an heuristic that cannot guarantee that all paths will be found but finishes in a reasonable time.

2. Embed h, r, t and all paths separately using a 2 layered BERT (an English Language Model) with hidden size n. This is done by tokenizing their labels, adding the special '[CLS]' token and retrieving the last hidden representation of '[CLS]'. This step turns h, r, t and each path into vectors of size n, allowing them to be processed by the next module. With this technique paths and entities alike are treated as sequences of words, and a dot is added to separate the relations in each path.

3. Using the vector representations, feed *h, r, t, paths* as input of a 4 layered Transformer Encoder, retrieving the final hidden representation of h, r and t. This step is where the model incorporate information from the paths into the processing of the fact triplet *(h, r, t)*. Reading the attention values of this module tells us which path of the Knowledge Graph was important in the final decision, allowing the user to gain insight in the automatic fact checking process.

4. Finally, pass the final representation of h, r and t into a feed-forward layer with a single output, representing the probability of the fact triplet being true.

This model takes advantage of the information contained in the labels in the KG, as well as the capacity of Transformers to use all of the information in the input at every step.

Shortening Paths

Because our model works with labels, the names of the entities and relations in the Knowledge Graph are important. Notably, relations in the FB15k-237 dataset tend to be quite long and repetitive, for example:
/award/award_nominee/award_nominations./award/award_nomination/award_nominee is a common relation in the dataset. Since a path is a sequence of relations, the label of some paths in FB15k-237 end up too big for the model, so we shorten them using .3, which would shorten the example to *'award/nominations.nominee'*.

3.5 Results

3.5.1 Link Prediction Evaluation

MAP%, which stands for *Mean Average Precision*, is a standard metric for Link Prediction benchmarks. The definition we use here is derived from Information Retrieval, which defines it as:

$$MAP = \frac{\|\{relevant\_documents\} \cup \{retrieved\_documents\}\|}{\|\{retrieved\_documents\}\|}$$

In our case, this is computed as follow:

1. ranking all the test triples *(h, r, t)* by the probability outputted by the Link Prediction model, thus assigning a rank i to each triples. A perfect model will rank every true triples higher than every other fake triples.

2. Iterate over every true triples counting them in a variable n, and add n/i to the final score for each one. We then divide the final score by the total amount of true triples. This way, a perfect model will have a *MAP%* score of 100%.

To evaluate the model's performance, we display the *MAP%* of the following in Table 3.1:

- Random Ranking: Assigns a uniformly random probability of any fact triplet *(h, r, t)* of being true. The resulting MAP% score is equal to the amount of positive examples divided by the amount of negative examples.

- Best Known: The previously best known *MAP%* on FB15k-237 is taken from [Liu+19a]. There is no previous best known *MAP%* for WikiData as it is an original dataset in this work, we display the results for future reference.

- Ours: The *MAP%* of our model, which make use of the information contained in the label of entities and relations.

	FB15k-237	WikiData
Random Ranking	20%	20%
Best Known	57.4%	-
Ours	**65.4%**	**85.1%**

Table 3.1: Performance comparison of our model on FB15k-237 and WikiData.

As we can see, the information contained in the English labels of entities and relations proved useful for the model, since it resulted in a significant increase of MAP% compared to previous methods that were not using this information. The price for this increase in performance however is that our model is language dependent, but we feel that it is an interesting trade-off to make for our use.

3.5.2 Explainability

In accordance to the objectives of this work, we detail how this model can be probed to obtain explanations associated to the predictions of the model, in order to be used by the end user.
As mentioned in section 3.2 (Related Work), using a Path Based approach instead of the KG Embedding for Link Completion allows us to ground the prediction of the model in a framework that is easier to understand for end users. By looking at the attention values of the model (3rd step of the architecture) over the paths we can extract the paths that were most relied upon by the model and visualize them, as we illustrate with some examples below, extracted on the dataset FB15k-237.
Figure 3.4 shows the paths that had the most importance (according to the model) when predicting that the triplet *(Wayne Knight, people/nationality, United States)* is true.

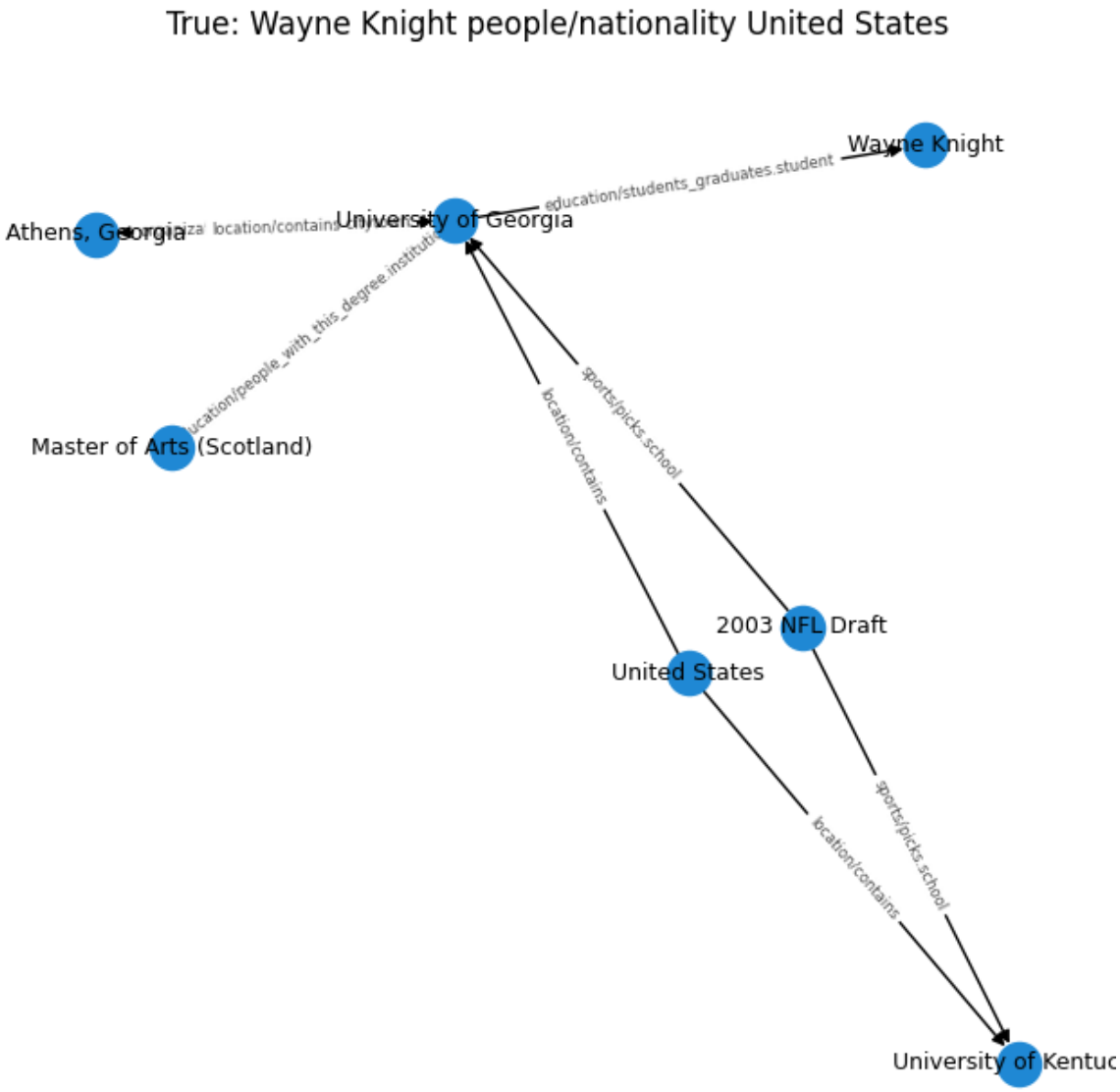

Figure 3.4: "Reasoning" on a True fact triplet

As we can see, the fact that *Wayne Knight* attended to a US university as well as the fact that he was part of a National Football League in the US was a strong predictor that *Wayne Knight* is indeed a US citizen.
This type of automatic "reasoning" unique to Path Based methods resembles a human deduction, and as such it is easily explained to the user.
Another example is Figure 3.5, showing the paths that had the most importance when predicting that the triplet *(Island Records, music/artist, Nat King Cole)* is false.

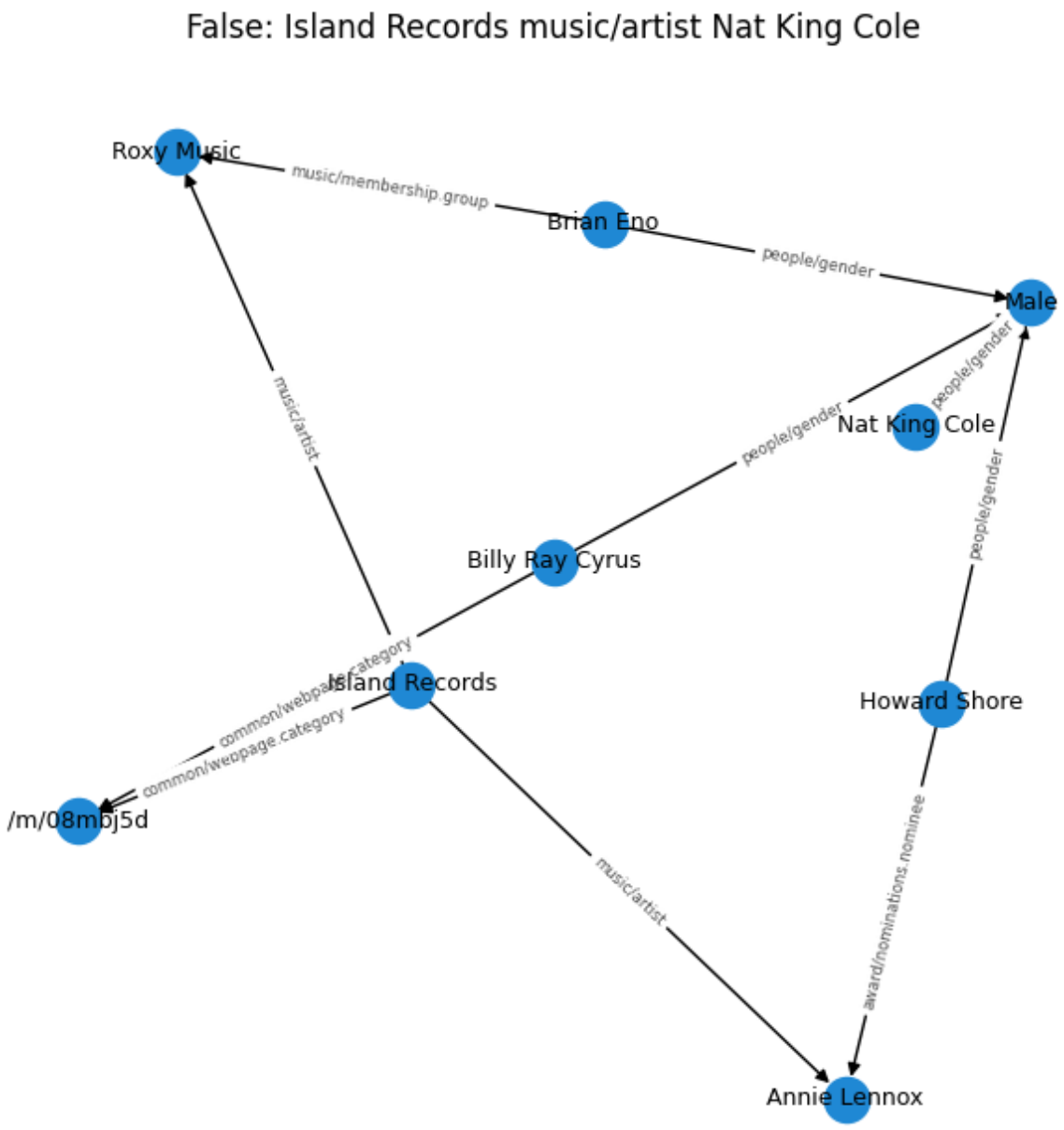

Figure 3.5: "Reasoning" on a False fact triplet

When presented with a false fact triplet, even the best connections between the 2 entities are too loose to assert that the triplet is true, thus leading the model to predict that it is false. And indeed, in this example the model predicts that *Nat King Cole* is not an artist of *Island Records* because the best connection it could find was that *Nat King Cole* had the same gender as other artists employed by *Island Records*, which a human would agree is not a sufficient connection to assert that *Nat King Cole* is also employed by *Island Records*.

3.6 Discussion

In this chapter we reviewed Automatic Content-Based Fact Checking with Knowledge Graph, re-framed as a Link Prediction task with a KG holding structured information about the world. We discussed how Path Based approaches to Link Prediction were able to yield more explainable results than the KG Embedding approach, with the constraint that the KG had to be denser. We also presented a new model that takes into consideration the labels of entities and relations in the Knowledge Graph, granting it a boost of performance at the cost of being language dependent by using a Transformer specialized in language modeling.

3.6.1 Application

As we saw in this chapter, this Path Based Approach to Link Completion in Knowledge Graphs can be used to produce explainable result, which is most appropriate for fact checking, however because of the necessity of searching for paths between entities in a huge KG, it is also a slower method comparatively. As opposed to the no-knowledge approach presented in chapter 2, the Path Based method with Knowledge Graph provides greater insight at the cost of speed, we thus suggest that this approach is best used as an analysis tool for fact checkers to query individual piece of news only.

3.6.2 Future Work

In this work, we used Knowledge Graphs that were already prepared and formatted, but a great deal of information about the world isn't written in this specific format. To apply KGs to fact checking it would be interesting to further the work of automatically building Knowledge Graphs from unstructured text such as news articles or social media posts. Notably, [WLS20a] provides a promising starting point by reading the attention scores of Transformers (technique mentioned in subsection 2.5.3) to infer fact triplets *(h, r, t)* that can be stored in a KG.

Furthermore, a common assumption used in this work is that fact triplets stored in the Knowledge Graph are always true, but this assumption can easily be violated by simple mistakes in the data or in the process of creating the KG. For this reason, work such as [KXO16] on probabilistic Knowledge Graphs is relevant and its application to fact checking could be explored.

Additionally, the recent work of scaling language models to enormous size [Ope20] combined with pre-training on as much data as possible has demonstrated excellent results. Since our architecture heavily relies on language models, scaling it up seems like a promising way of improvement.

Chapter 4

Model with Knowledge Base

4.1 Introduction

A fact checking model with a Knowledge Base (KB) uses the unstructured information present in KBs to support or contradict a given piece of news. In this introduction we discuss Knowledge Bases and their associated tasks, followed by their application to fact checking.

4.1.1 Knowledge Bases

Contrary to Knowledge Graphs, a Knowledge Base (KB) is a collection of unstructured information, most often in the form of natural text. A well known example of a Knowledge Base is Wikipedia, which hosts written articles about almost everything, with the aim of being factual, unbiased and easily understandable. As opposed to KGs which are better suited for computers to process, Knowledge Bases are designed to be read and understood by humans, making their utilization challenging in automated fact checking. However, because they do not have the format constraint of KGs, Knowledge Bases are more flexible in their ability to express facts, and the recent advents in Natural Language Processing makes them a very interesting tool for automated fact checking. Another advantage of using Knowledge Base for fact checking is that they naturally provide interpretable results to the user, as they rely on a corpus of knowledge that is adapted for humans.

Using Knowledge Bases in an automated system still poses a challenge known as Information Retrieval (IR), which is the task of retrieving a relevant passage/document from the KB given a query. Historically, a common approach to Information Retrieval (still in use today) is Bag Of Word (BoW) methods, the most famous of which is Okapi BM25. In short, BoW methods consist of representing each document of the Knowledge Base by the distribution of the words inside, disregarding structure, and then use the distribution of the words in the query to match it to documents. It is often complemented by a word weighting scheme such as tf-idf, allowing more specific words to have more weight when retrieving documents. With the recent advents of Deep Learning, new approaches to Information Retrieval have been developed, notably Passage Re-ranking. In Passage Re-ranking, a bunch of rel-

evant passages are first retrieved using a Bag of Words method, and are subsequently sorted by relevance with a deep learning model trained on a relevance dataset such as MS-Marco. This technique combine the speed of BoW methods and the greater accuracy of recent Deep Learning models, and has be shown to be effective in practice. Additionally, Passage Re-ranking can be complemented with Query Expansion [Nog+19a], in which a model tries to predict new relevant words to extend the query so that more relevant passages are retrieved.

4.1.2 Objectives

Fact checking using Knowledge Bases also implies some form of Textual Entailment task, because given a relevant passage from the KB, the model must be able to infer if the passage presents supporting or contradicting evidence to the fact. Textual Entailment is arguably more difficult than Information Retrieval, because deciding if a passage is relevant or not is a sub-task of deciding if it contradicts or supports the query.

In this chapter, we will give a broad overview of fact checking with Knowledge Bases, present a complete pipeline from Evidence Extraction to Textual Entailment, including a new model and a faster training procedure and discuss the application of this kind of model.

4.2 Related Work

[MW10] presents a fact checking model based on an external search engine, that works like so:

- Fact selection: Similarly from Triplet Selection in Knowledge Graphs, the author extract fact triplets from sentences using a Tree Parser. Note that this is not required when using Knowledge Bases, but because it makes Textual Entailment simpler the author chose this approach.
- Individual Fact Assessment: The fact triple is searched using Bing, the top results are then crawled and facts are extracted the same way as in step 1. The fact triple is then compared to the extracted triples and scored by whether or not they match.

The authors tested their model on an LCD catalog (newspaper aggregation) for true facts and general knowledge quizzes for false facts (such as *"The Dalai Lama lives in China"*). By relying on an external search engine, the authors avoid the problem of Information Extraction entirely, and get around Textual Entailment with semantic triples similar to Knowledge Graphs. This has the upside of being easier to make but limited in its capacity to express facts.

[Pop+18] presents DeClarE (*Debunking Claims from Interpretable Evidence*), a Neural Network model that takes as input a claim and its source, as well as a news

article, and outputs a credibility score for the claim. In this work, the articles are also retrieved with the external search engine Bing, and are compared against the claim one by one to yield a final credibility score. In short, the model works in 3 stages:

- Build a representation: Using Bidirectional LSTM [GFS05] and word embeddings from GloVe [PSM14], a vector representation of the claim and the article are made.
- Matching: Using an attention mechanism, the claim is matched to the relevant parts of the article, which are then combined to produce an embedding for both the claim and the article.
- Regression: To the previously obtained embedding is added an embedding for the source of the claim and the source of the article, yielding a final vector that is then passed through 2 feed-forwards layer to produce a credibility score as an output.

The authors use debunking website to test their models, Politifacts and Snopes, and reach 75% and 88% accuracy respectfully, and note that the accuracy does not decrease much when source of articles is not included. While the accuracy leaves room for improvement, they note that the model has the advantage of yielding interpretable results thanks to the attention mechanism that can highlight relevant parts of the article.

Another relevant work in fact checking with Knowledge Bases is FEVER [Tho+18], a dataset of 185k verifiable claims to evaluate a fact checking models on both passage selection and textual entailment. Additionally, the authors proposed a baseline model on their dataset, working with these steps:

- Document Retrieval: from [Che+17], returns the k nearest document from the query using cosine similarity on uni/bi-grams encoded with TF-IDF. A classical Bag of Word methods for Information Retrieval as discussed in the introduction.
- Sentence Selection: A simple TF-IDF similarity on uni-grams between sentences and the query, tuned on validation set.
- Recognizing Textual Entailment: uses decomposable attention [Par+16] to match fragments between the query and the passage from the Knowledge Base, and then compare those fragments with feed forward layers to conclude if they are in support or opposition, averaging to yield an answer for the whole sentence.

With this model, the authors reach 32% accuracy on evidence selection (selecting the best passage to support or refute the claim), and 51% accuracy to label the claim as either *verified*, *rejected* or *not enough info*. While this baseline model shows there is room for improvement, the steps of the model are still an inspiration to build fact checking models with Knowledge Bases.

4.3 Datasets

Before introducing the model, here is a breakdown of the tasks and datasets involved in this section.

Evidence Extraction

As stated in the introduction, a fact-checking model that uses Knowledge Base needs to automatically extract relevant passages. The evidence extraction part of our model is framed in the Passage Re-ranking framework, more formally:
Given a query and n candidate passages as input (extracted with a BoW method), the model must output a relevance score in $[0, 1]$ for each passage.
In this setting, the training data consists of *(query, passage, relevance)* examples, *relevance* being 0 if the passage is not relevant to the query and 1 if it is relevant.
We use the following datasets for this task:

- MS-MARCO: MS-Marco Passage re-ranking consists of ~500k training examples in the format *(query, passage, relevance)*. The queries are a filtered collection of real queries submitted to the search engine Bing with manually checked relevant passages. It is not focused on fake news but still useful due to the large amount of data.

- FEVER: FEVER (Fact Extraction and VERification) is made of ~150k training examples in the format *(fact, label, evidence)*, label being either *Supported*, *Refuted* or *NotEnoughInfo*. This dataset is geared towards fact checking but merges both *Evidence Extraction* and *Textual Entailment* in a single task, which can be cumbersome when training both task separately.

Textual Entailment

In a fact checking model, Textual Entailment is needed after extracting evidence from the Knowledge Base to indicate if the extracted passages support or contradict the fact to evaluate.
Given an input fact and an input passage from the KB, the model must classify the passage as being either in *Support* of the fact, in *Contradiction* with the fact, or *Neutral/Lacking info*.
We use the following datasets for this task:

- SNLI + MLNI: The *Stanford* and *Multi-genre Natural Language Inference* datasets are a collection of ~500k and ~400k training examples respectively, in the format of (query, label, evidence). They differ in that the MLNI dataset covers a wider range of genres of text than SLNI, but both are often used together as training data for Textual Entailment. Both datasets are general with no focus on fact checking but still useful for their large amount of data.

- The FEVER dataset is also used since it covers both Evidence Extraction and Textual Entailment, as described above.

Training on FEVER

Since FEVER mixes both Evidence Extraction and Textual Entailment together, it's not straightforward to extract from it a separate dataset for both task. FEVER examples are in the format *(fact, label, evidence)*, but when a fact is labeled with "Not enough Info" it means that there is no passage in FEVER's Knowledge Base that can either support or contradict the fact, so the *evidence* field is empty. This is a problem for both Textual Entailment and Passage Extraction, since the model must be able to tell if a passage is not relevant to the input fact.
Here is our procedure to circumvent this issue and extract complete training examples for both tasks from FEVER:

- If the training example *(fact, label, evidence)* has the label *Supports / Contradicts*, use it as is for Textual Entailment, and replace the label with a relevance score of 1 for Evidence Extraction.
- If the training example *(fact, label, evidence)* has the label *Not enough Info*, run the Passage Retrieval method in subsection 4.4.1 with the fact to extract a related passage, and put it in the *evidence* field. Use that as a training example for Textual Entailment, and replace the label with a relevance score of 0 for Evidence Extraction.

This procedure yields examples of non-relevant passages that are not too trivial to discriminate for the model so that it can learn properly.

4.4 Model

4.4.1 Passage Retrieval

In the passage re-ranking framework that we use, passage retrieval refers to a fast method to retrieve a bunch of candidate passage given a query from the Knowledge Base that will be re-ranked by a slower but more powerful method.
A classical way of doing passage retrieval is to use the Okapi BM25 algorithm, but it is too slow for our purposes and takes too much memory, so we introduce here a modified version, slightly less accurate but much faster and lighter.

Like all passage retrieving models, the BM25 variants [TPB14] operate by first **Indexing** the documents, and then **Querying** the index upon receiving a query. We will briefly detail these 2 steps in BM25 and explain the variation we apply to have a lighter Index and a faster Querying. We use this repository as reference for a common BM25 implementation.

BM25 Indexing

As a *Bag Of Words* technique, BM25 represents each document as a distribution of the tokens used in the document, disregarding order between words.
In the Indexing phase BM25 techniques tokenize each documents, and store:

- *f(t, D)*: the frequency of each term in each document, which indicate which documents are associated to each terms.
- *IDF(t)*: the inverse document frequency of each term, if the term t is present in few documents, *IDF(t)* will be high. Computed as $IDF(t) = ln\left(\frac{N-n(t)+0.5}{n(t)+0.5} + 1\right)$ with N the number of documents and *n(t)* the number of documents containing t; this indicates how *specific* each term is, poorly informative terms such as *the, and, a, an* will have a very low IDF score.
- $|D|$: the document length, i.e. the amount of tokens in each document, useful for regularization.
- *avgdl*: the average document length across the corpus.

Depending on the exact BM25 variant, other parameters will be stored but they typically do not use a significant portion of memory.

This indexing step usually takes a few hours but is done only once per corpus so it does not slow down the training of subsequent models.

BM25 Querying

Upon receiving a query Q, BM25 compute a relevance score for each document in the corpus to retrieve the most relevant.
The relevance score of a document D for the query Q is traditionally computed this way:

$$score(D, Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i, D)(k_1 + 1)}{f(q_i, D) + k_1 \left(1 - b + b\frac{|D|}{avgdl}\right)}$$

with q_i the i-th token of the query and k_1, b free parameters, usually set to $k_1 \in [1.2, 2.0]$ and $b = 0.75$. As is the case with indexing, the order of the tokens in the query is also disregarded.

Evaluating the score of 1 document is fast, but if there is a lot of documents in the corpus, computing the score of all documents to retrieve the highest can take a few seconds.

Our Implementation

When indexing, we drop every terms t for which $IDF(t) < \alpha$, α being an arbitrary cutoff value with default value $\alpha = 3$. The reasoning behind is that terms with a low IDF score are poorly informative and take a lot of space and computation time. α thus gives a trade-off between accuracy and performance; with $\alpha = -\infty$, the implementation is equivalent to BM25. The rest of the indexing is unchanged.

In the Querying phase, instead of computing the score for every document, we only compute the score for documents associated with terms of the query, because documents that are not associated with any query terms get a score of 0 anyway. More formally we evaluate the score of the following subset of documents:

$\{D \in Corpus\ ;\ \exists\, q_i \in Q, s.t., f(q_i, D) > 0\ \&\ IDF(q_i) > \alpha\}$ Combined with the term dropping in the Indexing phase, this means that only the documents associated to terms of the query that are *specific* enough will be evaluated, allowing us to only evaluate a small fraction of the corpus.
This is practical because generic terms such as *the, and, an, a* are associated with a big portion of the corpus, cost a lot of computation time and give very little information.

With a default value of $\alpha = 3$, this optimization drops the querying time from a few seconds down to less than 0.1 second on FEVER Knowledge Base (on our computer), and only cuts off the 105 less informative words. Our complete implementation is available here.

4.4.2 Passage Re-ranking & Textual Entailment

As is the case with the *no external knowledge model*, we also note here that the input of *Evidence Extraction* and *Textual Entailment* are almost the same: a query from a user is compared against a passage from the Knowledge Base, only the output is different, justifying hard parameter sharing again.
Since we adopted the Passage Re-ranking framework for evidence extraction the model has 2 components, Okapi BM25 for passage retrieval, and a single TinyBERT_4 for both passage re-ranking and textual entailment, with a different forward layer at the end depending on the task. Only TinyBERT_4 needs to be trained, using the procedure as follows:

- Tokenize the query and the input passage from the KB in the format: [CLS] Query [SEP] Passage [SEP], which is the standard format to give pairs of sequences as input to BERT models.
- Pass the tokenized input to TinyBERT_4
- Retrieve the last hidden representation of the '[CLS]' token, and pass it through the corresponding forward layer for the task, yielding either a relevance score for Passage Re-ranking or a label *Support*, *Contradiction* or *Neutral/Lacking info* for Textual Entailment.

Since the model learns both task at the same time, the same procedure as in subsection 2.4.2 is used to avoid Catastrophic Forgetting.

4.5 Results

Here each dataset has slightly different evaluation policies which we detail below:

- SLNI + MLNI: For each example consisting of *(premise, hypothesis)* the model is tasked to predict *Support*, *Contradiction* or *Neutral*, and we evaluate its accuracy.

- MS-MARCO: Given a single query the model must evaluate the relevance of 100 answers and is scored using the MAP% detailed in subsection 3.5.1, which reduces to a Mean Reciprocal Rank here given that there is only 1 positive example.
- FEVER: For a query the model is tasked to retrieve the correct passage from the Knowledge Base and asserts if the passage is in Support or in Contradiction with the query. If there is no relevant passage the model must output *Lacking Info*. The accuracy is evaluated but if the model does not retrieve the correct passage or does not output the correct label it counts as a failure.

We display the results of the following in Table 4.1:

- Majority/Random Baseline: On SLNI and FEVER, it is the accuracy obtained by predicting the most common label. On MS-MARCO, it is the MAP% score obtained by outputting a random relevance score.
- Best Known:
  - SLNI + MLNI: [Wan+21] reformulates various NLP tasks tackled by large language models into a textual entailment task, and achieves 93.1% on SLNI alone with the RoBERTa-large language model (355M parameters) [Liu+19b].
  - MS-MARCO: [Nog+19b] reaches 36.8% MAP using a combination of BM25, Document Expansion (similar to Query Expansion) and the BERT language model (110M parameters).
  - FEVER: [KYW21] obtain 74.1% FEVER score notably by applying attention across multiple passages in the Knowledge Base with the RoBERTa-large language model (370M parameters).
- Individual Models: The performance of the model we described on each task with no parameter sharing. Since FEVER is a joint task in nature however we did not evaluate it with no parameter sharing.
- Parameter Sharing: The performance of our model with hard parameter sharing between SLNI and MS-MARCO, and on FEVER after pre-training on SLNI and MS-MARCO first.

4.5.1 Evaluation

	SLNI + MLNI	MS-MARCO	FEVER
Majority/Random Baseline	50.0%	1%	33.0%
Best Known	93.1%	36.8%	76.0%
Individual Models	82.5%	34.5%	-
Parameter Sharing	83.3%	35.6%	68.6%

Table 4.1: Performance comparison of the model with and without parameter sharing, against the majority baseline and previous best accuracy.

While we couldn't achieve the best results with our model, it is important to note that it has only about 15M parameters, whereas the best performing models have between 100 and 300M parameters, making the comparison difficult.

Despite that, the results are still competitive and this experiment demonstrated again that hard parameter sharing with the batch mixing scheme explained in section .1 was able to perform better on the 2 tasks of Textual Entailment and Passage Re-ranking simultaneously than 2 copies of the same models would have done alone.

4.5.2 Explainability

As stated in the objectives, we consider being able to give explainable results to the end user to be a key feature of fact checking models.
Explainability happens to be a strong suit of models that work with Knowledge Base, because KBs are explicitly written to be read by humans (as opposed to Knowledge Graph) and the model relies on passages extracted from the KB to make a prediction.

In order to illustrate that, we broke down English Wikipedia articles that had more than 1000 daily views into passages, and indexed them with our BM25 implementation. We then queried the model trained on SLNI and MS-MARCO with fake and real facts, breaking down passages into sentences after retrieving them with BM25, and have the model select the 2 most relevant sentences, performing Textual Entailment on them to get a score for the fact.

We present below some examples of output of the model.

Input: Vaccines cause autism.

Retrieved passages:
Vaccine_hesitancy#Autism
"The idea of a link between vaccines and autism has been extensively investigated and conclusively shown to be false." -> Contradicts
Vaccine#Preservatives
"Furthermore, a 10–11-year study of 657,461 children found that the MMR vaccine does not cause autism and actually reduced the risk of autism by seven percent." -> Contradicts

Here a famous conspiracy hoax was used as input, and the model was able to correctly select relevant passages from Wikipedia and find a contradiction, even though it was not trained directly on this Knowledge Base. However we can see with the 2nd passage that the model can be prone to logical errors, because the fact that MMR vaccine were not found to cause autism does not imply in itself that all vaccine are safe.

Input: Donald Trump was a republican US president.

Retrieved passages:
Donald_Trump
"He entered the 2016 presidential race as a Republican and was elected in an upset victory over Democratic nominee Hillary Clinton while losing the popular vote, becoming the first U.S. president without prior military or government service." -> Supports
Donald_Trump#Republican_primaries
"After a landslide win in Indiana in May, Trump was declared the presumptive Republican nominee." -> Supports

In this other example a true statement was inputed, and the model was again able to find relevant passage to support it. We can also note the weakness in the reasoning of the model here, because the 2nd passage is not enough to affirm that Donald Trump was a US president.

Generally we observe that while the model is often able to retrieve relevant passages from the Knowledge Base, it seem to have a harder time making correct reasoning on these passages often leading to hasty "Supports" or "Contradicts" conclusion, were the evidence presented would not be enough to deduce one or the other. Fortunately, retrieving relevant passages from the KB is already helpful in itself to the end user, even if the model does not reason correctly with them.

4.6 Discussion

In this chapter we reviewed content-based fact checking with a Knowledge Base, framed as a combination of Information Retrieval and Textual Entailment tasks. We presented a simple pipeline with a faster version of BM25 for passage retrieval, and used hard parameter sharing with batch mixing on a language model to perform both passage re-ranking and textual entailment, with competitive results for its size. We also provided examples of how this model can be used to help fact checkers by using the retrieved passage from the KB.

4.6.1 Application

Out of the 3 approaches we discussed in each chapter, models with Knowledge Base are arguably the most amenable to explainable results, given that they can directly cite passages of natural text from a Knowledge Base when presented with a news to either support or contradict a piece of news.

However, similarly to models with Knowledge Graph, the need to look in the KB for evidence renders them slower than the no-knowledge model, making them better suited for single analysis than mass monitoring.

Not being restricted to the fact triples format *(h, r, t)* of KGs makes these models more flexible in how they represent the world, but less able to make profound connections between entities as easily, making their analysis complementary. Thus, we suggest that models with Knowledge Base can be used in conjunction with models with Knowledge Graphs as an analysis tool used on specific piece of news.

4.6.2 Future Work

Since Knowledge Graphs lack flexibility and Knowledge Base make long connections between entities hard to identify, exploring a combination of both seems like a natural follow-up and is definitely an interesting endeavor for fact checking applications. The work on semi-structured knowledge [HNP12] explores this combination and its application to Question Answering.

Tightly linked to this subject is the work of creating models with a multi-step reasoning ability such as [Zha+21], [Gan+19], which seems very relevant to the task of automatic fact checking and could potentially unite models with Knowledge Base and models with Knowledge Graphs into one model that is both flexible and able to establish deep connections.

Another matter that can be explored in more depth is the optimization of the BM25 algorithm which is widely used for fast passage retrieval from a Knowledge Base. The version presented in this work introduced a parameter alpha to drop terms that contained the least information in order to gain speed, but we unfortunately couldn't study in detail how the variation of this alpha parameters impacted the passages retrieved by the algorithm. It would be helpful to study this in the future and try to improve it further.

Finally the language model we use in this work is small because of technical limitations, but a scaled up version that also uses parameter sharing between passage re-ranking and textual entailment would be interesting to study.

Conclusion

In chapter 1 we briefly identified 4 approaches to automatic Fact Checking:

- *Aggregators*, indexing previous fact checking work to make them easily searchable.
- *Source-Based*, establishing reliable and unreliable sources of information to issue warnings of caution to users.
- *Network-Based*, inspecting the propagation of news in social networks to identify fake news.
- *Content-Based*, analyzing the content of the news itself to assess its reliability.

Because of its ability to function even when the source of a news is unknown and without social media context, we chose to focus on Content-Based Automatic Fact checking, which we divide into 3 approaches, one chapter each:

- models with *no external knowledge*, re-framing the problem of fact checking as simple Text Classification.
- models with a *Knowledge Graph*, re-framing the problem as Link Prediction of a fact triplet *(h, r, t)* in a KG.
- models with a *Knowledge Base*, re-framing the problem as a combination of Information Retrieval and Textual Entailment.

For each approach, we reviewed the prior work, presented an efficient model based on the recent advances in Language Modeling, and most importantly detailed how to get explainable results for the end user.

Additionally we tested hard parameter sharing by training a model on 2 tasks at once in chapter 2 and chapter 4, and evaluated a simple batch mixing technique to boost performance. We also derived a new dataset from WikiData for link-completion in chapter 3 and presented a modified BM25 algorithm in chapter 4 that allowed it to gain speed at the expanse of a small loss in information in a controlled manner.

We concluded that the model with no external knowledge was best suited for monitoring purposes, because of its speed and weak ability to give insight, whereas the models with Knowledge Graph and Knowledge Base were best suited for analysis on specific piece of news, being somewhat complementary with the Knowledge Graph trading flexibility for a deeply connected knowledge.

Future Work

In recent years, deep learning language models like we use here have demonstrated increasingly better results when scaled up and trained on a wide variety of tasks and datasets. In a future work on Content-Based Fact Checking, it would be interesting to try and combine the 3 models we presented into one bigger model, pre-trained on a large corpus of news and fake news.

Another avenue of improvement could come from using Multi-hop reasoning to replace models with Knowledge Graph and Knowledge Base with a model that uses semi-structured knowledge, which would hopefully have the flexibility of the KB model and the ability to establish deep connections like the KG model.

The automatic construction of Knowledge is also a topic of interest to improve Automatic Fact Checking, the construction of semi-structured Knowledge with Language Models specifically seems promising with prior works such as [WLS20a], which constructed a KG by inspecting the attention in a Transformer type Language Model.

Building a clean Fake News datasets with as much data as possible is one of the keys to improve Automatic Fact Checking, as the current datasets are either relatively small or contain a lot of errors. This could be done with a scrapping algorithm tailored for each format of news, similar to the FakeNewsCorpus dataset used in chapter 2 but with a better filtering of text and bad results (such as "Not Found" pages).

The very new architectures aimed at improving Transformers such as the Perceiver [21] should also be tested on the various Fact Checking tasks that we presented in this work, as they could bring significant improvement on performance in the near future, making automatic fact checking more accessible.

Appendix

.1 Distributing batches evenly

```
from torch.utils.data import Dataset


class MultiDataset(Dataset):
    def __init__(self, split_a, split_b, batch_size):
        self.batch_size = batch_size
        self.split_a = split_a
        self.split_b = split_b
        self.r = (len(split_a) + len(split_b)) / (len(split_b))

    def __len__(self):
        return len(self.split_a) + len(self.split_b)

    def __getitem__(self, item):
        """
        picks an element from either split_a or split_b s.t.:
        - a and b are well mixed (evenly spread)
        - we go through all a and b examples at least once
        (we inevitably have to go over some element
        twice due to batching)
        returns the chosen example and 0 if it's from split_a,
        1 if it's from split_b
        """
        i, rem = divmod(item, self.batch_size)
        b_prev = int(i/self.r)
        is_b = int((i+1)/self.r)-b_prev == 1
        if is_b:
            idx = b_prev*self.batch_size+rem
            row = self.split_b[idx % len(self.split_b)]
            return *row, 1
        idx = (i-b_prev)*self.batch_size+rem
        row = self.split_a[idx % len(self.split_a)]
        return *row, 0
```

.2 Triplet Extraction

```
import spacy
from spacy.tokens import Token
_subjs = {"nsubj", "nsubjpass", "csubj", "csubjpass"}
_ent_group = {"amod", "compound", "cc", "conj", "prep", "pobj"}
_rel_group = {"prep", "agent"}
_rel_transi = {"attr", "pobj", "dobj"}
nlp = spacy.load("en_core_web_sm")

def get_entity(ent: Token):
  tokens = []
  for t in reversed(list(ent.lefts)):
    if t.dep_ in _ent_group:
      tokens.append(get_entity(t))
  tokens.append(str(ent))
  for t in ent.rights:
    if t.dep_ in _ent_group:
      tokens.append(get_entity(t))
  return " ".join(tokens)

def get_relations(rel: Token):
  rels = []
  for t in rel.lefts:
    if t.dep_ in _rel_transi:
      rels.append((str(rel), t))
    elif t.dep_ in _rel_group:
      rels += [(f"{_rel} {rel}", _target) for _rel, _target in
    get_relations(t)]
  for t in rel.rights:
    if t.dep_ in _rel_transi:
      rels.append((str(rel), t))
    elif t.dep_ in _rel_group:
      rels += [(f"{rel} {_rel}", _target) for _rel, _target in
    get_relations(t)]
  return rels

def extract(text: str):
  doc = nlp(text)
  # get a list of subjects (usually only 1 but not always)
  subjs = [token for token in doc if token.dep_ in _subjs]
  # for each subject:
  triples = []
  for subj in subjs:
    # reconstitute head entity
    head = get_entity(subj)
    # happen each head, relation, target as a separate triple
    for rel, target in get_relations(subj.head):
      triples.append((head, rel, get_entity(target)))
  return triples
```

.3 Shortening Paths in FB15k-237

```
def shorten_r(rel):
    rs = rel.split(".")
    res = []
    # get a context
    parts = list(filter(None, rs[0].split("/")))
    ctx = None
    for ctx in parts[:-1]:
        if ctx != parts[-1]:
            break
    for r in rs:
        last_part = r.rsplit("/", 1)[-1]
        tokens = [t for t in last_part.split("_") if t != ctx]
        if tokens:
            res.append("_".join(tokens))
        else:
            res.append(last_part)
    ctx = ctx + "/" if ctx is not None else ""
    return ctx + ".".join(res)
```

.4 ITS Article

During this work, we could publish one article about Fact Checking with Knowledge Graphs, in which we explained in detail how Knowledge Graphs could be used to get explainable fact checking results, and published a first version of the model that we later improved. The paper was published in The 17th Intelligent Tutoring Systems Conference, 2021, Springer Verlag Lecture Notes in Computer Science. The contribution of the writers is as follows:

- Téo Orthlieb: Building the model, doing the experiments and writing the paper
- Hamdi Ben Abdessalem: contributed with his advice in this project and revised the paper
- Claude Frasson: revised the paper and financed the project

This complete version of the publication is as follows:

Checking Method for Fake News to Avoid the Twitter Effect

Téo Orthlieb, Hamdi Ben Abdessalem and Claude Frasson

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal, Montréal, Canada H3C 3J7
{teo.orthlieb, hamdi.ben.abdessalem}@umontreal.ca,
frasson @iro.umontreal.ca

Abstract. The recent blocking of President Trump's twitter account has raised awareness of the danger of the impact of fake news and the importance of detecting it. Indeed, if one can doubt information, ignoring what is true or false it can lead to a loss of confidence in the decisions and other dangers. The objective of this paper is to propose an automatic method for fact checking using Knowledge Graphs, such as Wikipedia. Knowledge Graphs (KGs) have applications in many tasks such as Question Answering, Search Engines and Fact Checking, but they suffer from being incomplete. Recent work has focused on answering this problem with an abstract embedding of the KG and a scoring function, yielding results that are not easily interpretable. On the other hand, Path Ranking methods answer this problem with deductions represented by alternative paths in the KG, easily understood by a human. Favoring the Path Ranking approach for its interpretability, we propose an attention-based Path Ranking model that uses label information in the KG, making the model easily transferable between datasets, allowing us to leverage pretraining and demonstrate competitive results on popular datasets.

Keywords: Fact Checking, Knowledge Graphs, Deep Learning.

1 Introduction

Large Knowledge Graphs (KG) such as Wikidata, FreeBase or WordNet aim at storing facts of a domain in a structured manner, which is typically done with a multi-relational directed graph, where each node is an entity and each edge is labelled with a relation. We thus denote a fact by a triplet ***(h, r, t)***, with ***h*** and ***t*** the head and tail entities, and ***r*** the relation, for example (Sean Connery, Profession, Actor).

In this context, the task of fact checking can be reformulated as a task of Link Prediction in the Knowledge Graph.

A popular approach to link prediction is Knowledge Graph Embeddings, which operate on nodes and edges as vectors in a latent space and use a scoring function to assert if a certain triplet is likely to be true. For instance, TransE [2] embeds both relations and entities in the same space, and ensures that $h + r \approx t$.

However, one inconvenient shared by Knowledge Graph Embeddings is that predictions are hard to explain, because the model operates only in a latent space that doesn't necessarily make sense for a human.

On the other hand, path ranking methods base their decisions on alternative paths in the KG.

For example, if the nationality of Sean Connery was not documented, a path ranking model could infer that because Sean Connery was born in Edinburgh, and because Edinburgh is the capital of Scotland, then Sean Connery was Scottish. These kinds of reasoning are understandable from a human perspective, and even if the model might be wrong, they still provide a useful insight to the user.

This article focuses on building a path ranking kind of model that is transferable between different Knowledge Graphs, and also proposes a new dataset that is more adapted for fact checking.

2 Related Work

A notable step for path ranking methods in Knowledge Graphs was [5] in which the Path Ranking Algorithm (PRA) from an earlier work [6] was adapted for KGs.

Simply put, the training procedure is as follows, for each (h, r, t) a number of depth-limited random walks are executed, and the associated meta-paths are then scored by how often they can reach the tail entity t. At prediction time given h and r a number of depth-limited random walks are performed again, and the end node of every paths weighed by the score of their associated meta-paths are combined to yield a distribution of end nodes. Predictions for the tail entity t are then ordered by their likelihood in this distribution.

The Path Ranking Algorithm is thus a purely statistical learning model, and provides a strong baseline that works well even with few data.

More recently, Path Ranking techniques incorporating deep learning have been introduced:

- DeepPath [7] instead frames the task of path ranking as a Reinforcement Learning task. For a given ***(h, r, t)*** the agent is tasked to find the most informative paths from ***h*** to ***t***.
- Path Ranking with Attention to Type Hierarchies [10] takes another approach to path ranking by incorporating type hierarchies. The reasoning is that for any entity, different level of abstractions can be relevant. For instance, we can think of a Fork as Cutlery, Tableware or simply Ware.

However, they only result in a slight improvement over PRA on FB15k-237 [8].

All the path ranking methods above share one characteristic that we must detail. For any KG dataset, every nodes and relations are typically associated with a unique ID that differs between datasets. These IDs have the advantage of being independent of language and resilient to homonyms, but they have the disadvantage of not being transferable between datasets and they are not as informative as a label.

We explore representing nodes and relations with their labels directly instead of an ID, allowing the model make use of pretraining, which has been shown to be very effective in language tasks over the last decade [11].

3 Architecture

At training time, the model is given a fact triplet ***(h, r, t)*** as well as random depth limited meta-paths linking ***h*** to ***t***. The model is tasked to output the probability that the fact triplet is valid. In order to do so the model uses various modules:

- **Label embedding**: the model embeds all the nodes and relations in the input using the word level embedding on their labels. This transforms labels into vectors which the model can then use subsequently.
- **Meta-path encoding**: every meta-path is encoded with a GRU, in order to transform each meta-path into a single vector.
- **Relation encoding**: the model uses all the meta-path as a context in a transformer-type model [14] to get a new representation of ***r***. In other words, this is the module in which the model thinks about all the relations he knows between the head and tail entity.
- **Classification Layer**: ***h***, ***r***, and ***t*** are fed through a final feedforward layer to yield the estimate probability that the fact represented by ***(h, r, t)*** is true.

Through experiments, we found the following hyperparameters to work well:

- Depth of paths = 4
- Embedding size = 50
- Learning Rate Cycling between [0.005; 0.1]

4 Datasets

Unfortunately, we couldn't find a dataset complete with negative examples used in *DeepPath* or *Path Ranking with Attention* to Type Hierarchies, and were not able to run the provided method to generate negative examples. So, we made our own method to generate negative examples as well as a new bigger dataset from WikiData, and published everything here: https://github.com/Inspirateur/KG_Datasets

Since the previous papers don't use the same method to generate negative examples, comparing against them would be unfair, hopefully this will not be a problem for future papers since the complete datasets we use are public.

A random baseline yields 16% Maximum Average Precision (MAP) on FB15k-237 and WikiData, and our model (TAP-KG) reaches 78% MAP on FB15k-237 and 51% MAP on WikiData, which seems to be a harder dataset.

5 Visualization of Results

The ability to understand predictions from our model is a very important point, and we're able to do it by inspecting the **Relation encoding** part of the model to see which paths it relied the most on to make the prediction, illustrated in Fig. 1.

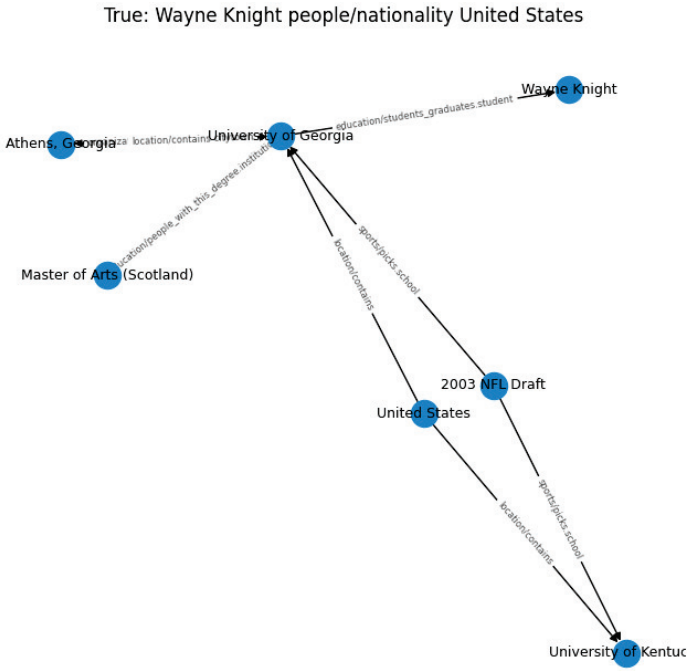

Fig. 1. The model deduces that Wayne Knight is a USA citizen because he graduated University of Georgia which is located in the USA

6 Conclusion

In order to help fighting against the rapid spread of fake news, we proposed an automatic fact checking model with interpretable results. Indeed, even though deep learning models have demonstrated great performance in recent years, they are often thought of as "black box" because it is challenging to understand what's happening inside. However, the datasets used to train and test the model are not perfect. Since they are usually made automatically, some tasks like predicting that England contains Birmingham when given "England", "contains" are nonsensical, because there are a lot of cities in England and the model could give many correct answers before giving "Birmingham".

An improvement for the future would thus be to construct better tasks to train and test models on link prediction in knowledge graphs.

Acknowledgment. We acknowledge NSERC-CRD (National Science and Engineering Research Council Cooperative Research Development), Prompt, and BMU (Beam Me Up) for funding this work.

References

1. C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer, "Hoaxy: A Platform for Tracking Online Misinformation," in Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion, 745–750, Montréal, Québec, Canada, (2016).
2. A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, 2787–2795, Red Hook, NY, USA, (2013)
3. Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space," arXiv:1902.10197 [cs, stat], (2019).
4. R. Wang, B. Li, S. Hu, W. Du, and M. Zhang, "Knowledge Graph Embedding via Graph Attenuated Attention Networks," IEEE Access, vol. 8. 5212–5224, (2020).
5. N. Lao, T. Mitchell, and W. W. Cohen, "Random Walk Inference and Learning in A Large Scale Knowledge Base," in Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, 529–539, Edinburgh, Scotland, UK., (2011).
6. N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," Mach Learn, vol. 81, no. 1, 53–67 (2010)
7. W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 564–573, Copenhagen, Denmark, (2017).
8. T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D Knowledge Graph Embeddings," CoRR, vol. abs/1707.01476, (2017).
9. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, "Toward an Architecture for Never-Ending Language Learning," (2010).
10. W. Liu, A. Daruna, Z. Kira, and S. Chernova, "Path Ranking with Attention to Type Hierarchies," AAAI, vol. 34, no. 03, 2893–2900, Apr. (2020).
11. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," CoRR, vol. abs/1810.04805, (2018).
12. J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543, Doha, Qatar, (2014).
13. K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," CoRR, vol. abs/1409.1259, (2014).
14. A. Vaswani et al., "Attention Is All You Need," CoRR, vol. abs/1706.03762, (2017)

Bibliography

[Car93] Richard A. Caruana. "Multitask learning: A knowledge-based source of inductive bias". In: (1993). URL: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=085ABB965F3D72B5C43829EBD799B295?doi=10.1.1.57.3196&rep=rep1&type=pdf.

[MH93] Ken McRae and Phil A. Hetherington. "Catastrophic Interference is Eliminated in Pretrained Networks". In: (1993). URL: https://www.researchgate.net/publication/2418146_Catastrophic_Interference_is_Eliminated_in_Pretrained_Networks.

[Rob95] Anthony Robins. "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: (1995). URL: https://www.tandfonline.com/doi/abs/10.1080/09540099550039318.

[GFS05] Alex Graves, Santiago Fernandez, and Jurgen Schmidhuber. "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition". In: (2005). URL: https://www.cs.toronto.edu/~graves/icann_2005.pdf.

[Rus+07] Delia Rusu et al. "Triplet extraction from sentences". In: (2007). URL: https://www.researchgate.net/publication/228905420_Triplet_extraction_from_sentences.

[MW10] Amr Magdy and Nayer Wanas. "Web-based statistical fact checking of textual documents". In: (2010). URL: https://dl.acm.org/doi/pdf/10.1145/1871985.1872002.

[LMC11] Ni Lao, Tom Mitchell, and William W. Cohen. "Random Walk Inference and Learning in A Large Scale Knowledge Base". In: (2011). URL: https://www.cs.cmu.edu/~tom/pubs/lao-emnlp11.pdf.

[HNP12] Eduard Hovy, Roberto Navigli, and Simone Paolo Ponzetto. "Collaboratively built semi-structured content and Artificial Intelligence: The story so far". In: (2012). URL: https://www.sciencedirect.com/science/article/pii/S0004370212001245.

[Bor+13] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: (2013). URL: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[TPB14] Andrew Trotman, Antti Puurula, and Blake Burgess. "Improvements to BM25 and Language Models Examined". In: (2014). URL: https://dl.acm.org/doi/abs/10.1145/2682862.2682863.

[Cia+15] Giovanni Luca Ciampaglia et al. "Computational fact checking from knowledge network". In: (2015). URL: https://arxiv.org/pdf/1501.03471.pdf.

[HLT15] Naeemul Hassan, Chengkai Li, and Mark Tremayne. "Detecting Check-worthy Factual Claims in Presidential Debates". In: (2015). URL: https://dl.acm.org/doi/pdf/10.1145/2806416.2806652.

[RCC15] Victoria L. Rubin, Yimin Chen, and Nadia K. Conroy. "Deception Detection for News: Three Types of Fakes". In: (2015). URL: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/pra2.2015.145052010083.

[BTB16] Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. ""8 Amazing Secrets for Getting More Clicks": Detecting Clickbaits in News Streams Using Article Informality". In: (2016). URL: https://ojs.aaai.org/index.php/AAAI/article/view/9966.

[Jin+16] Zhiwei Jin et al. "News Verification by Exploiting Conflicting Social Viewpoints in Microblogs". In: (2016). URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewFile/12128/12049.

[KXO16] Dongwoo Kim, Lexing Xie, and Cheng Soon Ong. "Probabilistic Knowledge Graph Construction: Compositional and Incremental Approaches". In: (2016). URL: https://arxiv.org/pdf/1608.05921.pdf.

[Par+16] Ankur P. Parikh et al. "A Decomposable Attention Model for Natural Language Inference". In: (2016). URL: https://arxiv.org/pdf/1606.01933.pdf.

[PT16] Ellie Pavlick and Joel Tetreault. "An Empirical Analysis of Formality in Online Communication". In: (2016). URL: https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00083/43356/An-Empirical-Analysis-of-Formality-in-Online.

[Sha+16] ChengCheng Shao et al. "Hoaxy: A Platform for Tracking Online Misinformation". In: (2016). URL: https://arxiv.org/pdf/1603.01511v1.pdf.

[Che+17] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: (2017). URL: https://aclanthology.org/P17-1171.pdf.

[Tac+17] Eugenio Tacchini et al. “Some Like it Hoax: Automated Fake News Detection in Social Networks”. In: (2017). URL: https://arxiv.org/pdf/1704.07506.pdf.

[Vas+17] Ashish Vaswani et al. “Attention is all you need”. In: (2017). URL: https://arxiv.org/pdf/1706.03762.pdf.

[Wan17] William Yang Wang. ““Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection"”. In: (2017). URL: https://arxiv.org/pdf/1705.00648v1.pdf.

[Det+18] Tim Dettmers et al. “Convolutional 2D Knowledge Graph Embeddings”. In: (2018). URL: https://arxiv.org/pdf/1707.01476.pdf.

[HR18] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: (2018). URL: https://arxiv.org/pdf/1801.06146.pdf.

[Pop+18] Kashyap Popat et al. “DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning”. In: (2018). URL: https://arxiv.org/pdf/1809.06416.pdf.

[Rad+18a] Alec Radford et al. “Improving Language Understanding by Generative Pre-Training”. In: (2018). URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[Rad+18b] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2018). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[Tho+18] James Thorne et al. “FEVER: a large-scale dataset for Fact Extraction and VERification”. In: (2018). URL: https://arxiv.org/pdf/1803.05355.pdf.

[XHW18] Wenhan Xiong, Thien Hoang, and William Yang Wang. “DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning”. In: (2018). URL: https://arxiv.org/pdf/1707.06690.pdf.

[Che+19] Brian Cheung et al. “Superposition of many models into one”. In: (2019). URL: https://arxiv.org/pdf/1902.05522.pdf.

[Dev+19] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (2019). URL: https://arxiv.org/pdf/1810.04805.pdf.

[Gan+19] Zhe Gan et al. “Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog”. In: (2019). URL: https://arxiv.org/pdf/1902.00579.pdf.

[Jia+19] Xiaoqi Jiao et al. “TinyBERT: Distilling BERT for Natural Language Understanding”. In: (2019). URL: https://arxiv.org/pdf/1909.10351.pdf.

[Liu+19a] Weiyu Liu et al. "Path Ranking with Attention to Type Hierarchies". In: (2019). URL: https://arxiv.org/pdf/1905.10799.pdf.

[Liu+19b] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (2019). URL: https://arxiv.org/pdf/1907.11692.pdf.

[MS19] Rahul Mishra and Vinay Setty. "SADHAN: Hierarchical Attention Networks to Learn Latent Aspect Embeddings for Fake News Detection". In: (2019). URL: https://dl.acm.org/doi/abs/10.1145/3341981.3344229.

[Nog+19a] Rodrigo Nogueira et al. "Document Expansion by Query Prediction". In: (2019). URL: https://arxiv.org/pdf/1904.08375v2.pdf.

[Nog+19b] Rodrigo Nogueira et al. "Document Expansion by Query Prediction". In: (2019). URL: https://arxiv.org/pdf/1904.08375v2.pdf.

[Zel+19] Rowan Zellers et al. "Defending Against Neural Fake News". In: (2019). URL: https://arxiv.org/pdf/1905.12616.pdf.

[BSS20] Bjarte Botnevik, Eirik Sakariassen, and Vinay Setty. "BRENDA: Browser Extension for Fake News Detection". In: (2020). URL: https://arxiv.org/pdf/2005.13270.pdf.

[Al-+20] Tareq Al-Moslmi et al. "Named Entity Extraction for Knowledge Graphs: A Literature Overview". In: (2020). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8999622.

[Ope20] OpenAI. "Language Models are Few-Shot Learners". In: (2020). URL: https://arxiv.org/pdf/2005.14165.pdf.

[WLS20a] Chenguang Wang, Xiao Liu, and Dawn Song. "Language Models Are Open Knowledge Graphs". In: (2020). URL: https://arxiv.org/pdf/2010.11967.pdf.

[WLS20b] Chenguang Wang, Xiao Liu, and Dawn Song. "Language Models are Open Knowledge Graphs". In: (2020). URL: https://arxiv.org/pdf/2010.11967.pdf.

[21] In: (2021). URL: https://arxiv.org/pdf/2103.03206.pdf.

[KYW21] Canasai Kruengkrai, Junichi Yamagishi, and Xin Wang. "A Multi-Level Attention Model for Evidence-Based Fact Checking". In: (2021). URL: https://aclanthology.org/2021.findings-acl.217.pdf.

[OAF21] Teo Orthlieb, Hamdi Ben Abdessalem, and Claude Frasson. "Checking Method for Fake News to Avoid the Twitter Effect". In: *ITS 2021, The 17th Intelligent Tutoring Systems Conference, Athens, Greece, June 7-11, 2021, Springer Verlag Lecture Notes in Computer Science.* (2021).

[Wan+21] Sinong Wang et al. "Entailment as Few-Shot Learner". In: (2021). URL: https://arxiv.org/pdf/2104.14690v1.pdf.

[Zha+21] Chen Zhao et al. “Multi-Step Reasoning Over Unstructured Text with Beam Dense Retrieval”. In: (2021). URL: http://users.umiacs.umd.edu/~hal/docs/daume21beamdr.pdf.