Neural Relation Extraction on Wikipedia Tables for Augmenting Knowledge Graphs

by

Erin Macdonald

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Knowledge graphs are an important source of information used in a number of applications including web search, online shopping, social networking, and chatbots. They are an effective way of storing real-world data in a machine-readable format. As a result, the construction of comprehensive, trustworthy knowledge graphs has been a well-researched problem. We present a method for adding new facts to an existing knowledge graph using Wikipedia tables as a source of information. Previous work has primarily focused on extracting facts from text, ignoring the information available in tables.

We use an existing knowledge graph to annotate a set of Wikipedia tables using distant supervision with relations between pairs of columns. Then, we run a classifier on these tables to remove as many tables brought in by error as possible. We also create queries based on table formats identified as indicative of certain relations to increase the number of tables collected. In total, we annotate over 200,000 relational tables with these methods.

We then train a long short-term memory (LSTM) network using these tables to predict a relation given a table and pair of columns. We perform an ablation study to identify what features are weighted most heavily and provide the most information to the LSTM. We also explore how two different state-of-the-art word embedding sets fare. Our experiments show that our system is able to correctly predict which relation a pair of columns represents with over 87% accuracy. We compare our results with two other relation prediction systems which use different datasets of tables and show that our

method achieves higher accuracy, though a more direct comparison can not be performed.

Preface

This thesis is an original work by Erin Macdonald. No part of this thesis has been previously published.

Acknowledgements

There are many people who supported and guided me through the process of writing this thesis and without whom I would not be where I am today. First and foremost, I would like to thank my supervisor, Dr. Denilson Barbosa, for introducing me to the fields of NLP and information extraction and for helping me accomplish what seemed to be an impossible task two years ago. I can not imagine completing this work without the supervision Dr. Barbosa has provided the past two and a half years.

I am also grateful for the comments and revisions provided by my committee members, Drs. Marek Reformat and Osmar Zaïane, who have spent their valuable time helping me present my best work. Additionally, I am indebted to Natalie Hervieux and Daniel Cones for the countless hours they spent helping me annotate hundreds of tables. Without their help, I would not have been able to complete this work. Finally, this thesis would not have been possible without the generous support of the Queen Elizabeth II Scholarship, NSERC grant, and Mitacs Accelerate Fellowship I recieved. The aid provided by these three funding bodies allowed me to focus all my energy on my degree.

On a personal level, I have to thank my parents, Loch and Sheilah, for always encouraging and loving me and for instilling in me the value of education. They were able to raise myself and my siblings in a household that appreciated science and critical thinking which provided me with a 25 year head start on this degree. I am also grateful to my older siblings, Iain and Robyn, for setting such amazing examples and always pushing me to keep up with them. They also inspired me to study Computing Science and to pursue a graduate degree and were always there for me when I needed an outlet. My younger sister, Sunshine, was always there for constant cuddling and belly rubs the past two

years and provided the greatest puppy therapy. I also would like to thank my grandparents for always supporting my education and research even if they might not understand it. Finally, the friends in my life have always been there for me when I needed someone to listen. Thanks to Hayley Machat, Victoria Hessdorfer, Daniel Caminhas and Andrea Whittaker for the countless hours of conversation and for sharing their lives with me all these years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Knowledge graphs are a way of representing information in a machine-readable format. The term "knowledge graph" has been used by researchers for years but was popularizes in 2012 when Google used it to describe how they use structured, real-world data gathered from numerous sources to improve the results of and speed up their search engine[1]. Although researchers had been presenting similar information in an analogous format like WordNet and Freebase, they soon adapted the new term[2]. Since then, knowledge graphs have become a popular way of representing relational data found on the web. Knowledge graphs are key to the operation of search engines like Google [46], online shopping retailers like Amazon[3], and social networks like Facebook [33].

Because they are so ubiquitous, a fair amount of research has been focused on building comprehensive and accurate knowledge graphs [34]. Even the best modern knowledge graphs suffer from either inaccuracies or incompleteness. Knowledge graphs with high coverage are often built automatically and contain errors, while accurate knowledge graphs are usually created by humans who are unable to codify all available information. As a result, even the best knowledge graphs are incomplete. This thesis describes one method for finding

[1]https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

[2]https://wordnet.princeton.edu/, https://developers.google.com/freebase

[3]https://aws.amazon.com/blogs/apn/exploring-knowledge-graphs-on-amazon-neptune-using-metaphactory/

new data to add to an existing knowledge graph, DBpedia. We hypothesize that training a neural network with a large annotated dataset to predict relations in tables will be accurate and will provide a large number of new facts to add to Freebase.

The main components of a knowledge graph are entities and relations. Entities are real world "things" which are generally referred to in text by proper nouns. Examples of entities can include (at a high level) individuals, geographical locations, organizations, and products. Entities are the nodes of the graph and are connected to one another through edges representing relations. Examples of relations include the birthplace of a person, capital of a country, director of a movie or political party of a politician.

1.1.1 Building Knowledge Graphs

There are multiple reliable public knowledge graphs. One example is Freebase which was acquired by Google and initially built using mostly human annotation before turning to more automated methods to improve coverage [45] In Freebase, each article on Wikipedia becomes an entity in Freebase. To find relations, the contributers to Freebase used structured dumps from various information-based sites (such as IMDB) and extracted information relatively reliably. By modern standards, Freebase is a moderately clean and comprehensive knowledge graph, though it still misses a large amount of information and has a number of errors.

DBpedia is another public knowledge graph built using only the information available on Wikipedia. Like Freebase, every article on Wikipedia becomes an entity in DBpedia. DBpedia uses the infobox to build relations for a given article. This creates another relatively clean and thorough knowledge graph, though a large number of relations are still extracted erroneously.

In this thesis, we use the tables in Wikipedia articles to add new edges to Freebase as most previous work on relation extraction (particularly with tables) has used Freebase. Tables have never before been used to extract triples for Freebase or any other knowledge graph but provide a valuable source of semi-structured and abundant information.

```
PREFIX dbr:  <http://dbpedia.org/resource/>
PREFIX dbo:  <http://dbpedia.org/ontology/>

dbr:Louise_of_Hesse-Kassel dbo:birthPlace dbr:Kassel.
dbr:Louise_of_Hesse-Kassel dbo:spouse dbr:Christian_IX_of_Denmark.
```

Figure 1.1: An example of some RDF triples describing a small knowledge graph.

1.1.2 Resource Document Format

One way to represent information in knowledge graphs is using Resource Document Format or RDF. This is a triple format where two entities, a subject and object, are assigned a predicate to define how the entities are related. Therefore, each triple defines one edge of the graph. The subjects, predicates, and objects of triples are defined by URIs or uniform resource identifiers (an example of a URI is `http://dbpedia.org/page/Louise_of_Hesse-Kassel`). To remove repetition from a set of triples, we often define prefixes for common URIs and use shortened forms in our triples.

One example of a set of triples is given in Figure 1.1 which uses prefixes `dbr:` and `dbo:` for cleanliness. This small knowledge graph defines 3 entities (Louise of Hesse-Kassel, Christian IX of Denmark, and Kassel) and 2 relationships (birthplace and spouse).

1.2 Problem Definition

Information Extraction (IE) is the term used to describe methods for populating knowledge graphs with information. IE on text encompasses a number of subtasks with the goal of enabling computers to extract the key pieces of information encoded in written text and encode them in RDF to insert into a knowledge graph [19]. These subtasks include performing named entity recognition (NER), coreference resolution, entity linking and relation extraction (RE). NER and coreference resolution attempt to identify and disambiguate mentions of entities or proper nouns in the provided text.

An example of NER and coreference resolution on text is given in Fig-

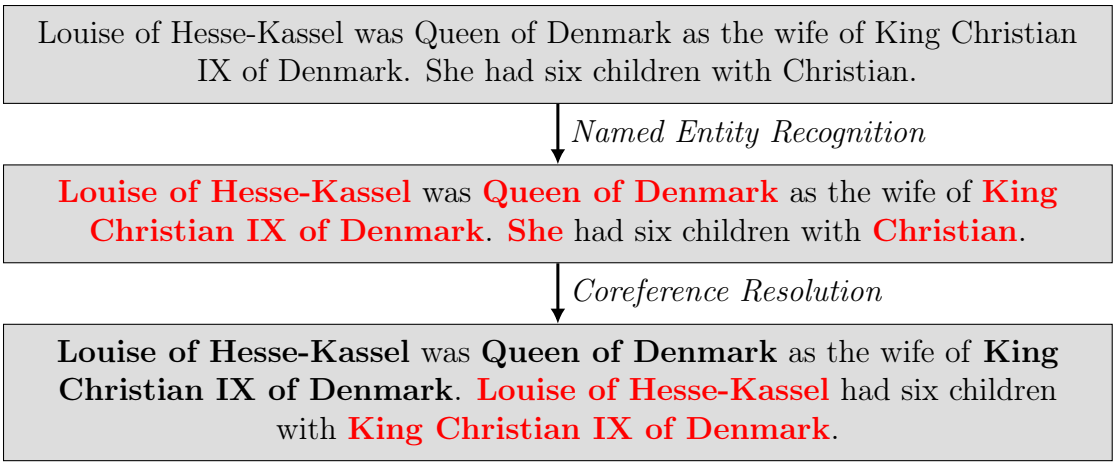

Figure 1.2: An example of named entity recognition and coreference resolution on text taken from the Wikipedia article for "Louise of Hesse-Kassel".

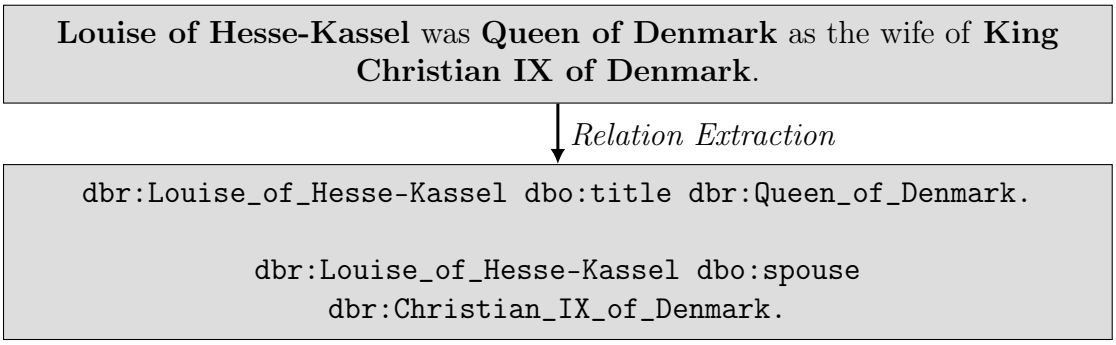

Figure 1.3: An example of relation extraction on a sentence from the Wikipedia article for "Louise of Hesse-Kassel".

ure 1.2. Entity linking is the task of matching those mentions to entities in a knowledge graph. For example, the two mentions of "King Christian IX of Denmark" in Figure 1.2 can be linked to the DBpedia entry for the entity, represented in Figure 1.3 as `dbr:Christian_IX_of_Denmark.`.

An important step in IE is determining how the entities in the text are related to one another. This is referred to as relation extraction and is a key task in the work to build accurate and informative knowledge graphs [19]. Figure 1.3 shows the RDF triples that can be extracted from the annotated sentence.

Table understanding is the term used for ascribing meaning to the information in a table. The goal of table understanding is to determine the types of columns in a table (and by extension the types of entities in the corresponding

rows) or to detect relations between pairs of columns mentioning entities [24]. After predicting a relation between a pair of columns, we can gather triples by assigning this relationship to the entities in the numerous rows of those columns [32]. For this thesis we used Wikipedia articles containing tables as our dataset so the cells mentioning entities were already hyperlinked to the relevant entities. This meant the task of entity detection had already been completed. This left us to solve the task of identifying how a pair of columns were related in order to produce triples.

We chose relations to predict from Freebase (mostly from related work) and used Wikipedia tables as our corpus of tables. This is because not only are the tables generally well-maintained, factual, and hyperlinked, but Wikipedia also contains a wealth of information about a variety of topics. This provided us with a large, clean corpus of non-domain-specific data with which to test our method. Additionally, research has shown that other corpuses of tables contain a small number of relational tables to work with [4].

In general, most tables have one column that acts as a subject and which may be related to all other columns in the table, a concept introduced by Venetis et al. and now accepted among table understanding researchers [48]. Venetis et al. also showed that the leftmost column of a table is the subject column 75% of the time [48]. While their dataset was different, we can assume that in general, tables on Wikipedia and tables elsewhere on the web are structured similarly. This is especially true considering Wikipedia's editing guidelines when compared to the rest of the web.

We predicted relations between any pair of columns, building on earlier research which attempted to identify this subject column first and then predicted relations using that column [48]. For example, in Figure 1.4 the subject column of the table is the first, with the header "Name". We can predict the relationship *person-spouse* between the first and fourth columns and *person-children* between the first and fifth columns, producing 2 triples for *person-spouse* and 14 for *person-children* (relating Frederick VIII of Denmark to his 8 children and Princess Alexandra of Denmark to her 6 children). By also predicting relations for columns other than the subject column, we can produce another

Children [edit]

Louise had the following six children with Christian. Eventually, they had thirty-nine grandchildren.

Name	Birth	Death	Spouse	Children
Frederick VIII of Denmark	3 June 1843	14 May 1912	Princess Louise of Sweden	Christian X of Denmark Haakon VII of Norway Louise, Princess Frederick of Schaumburg-Lippe Prince Harald of Denmark Princess Ingeborg, Duchess of Västergötland Princess Thyra of Denmark Prince Gustav of Denmark Princess Dagmar, Mrs. Castenskiold
Princess Alexandra of Denmark	1 December 1844	20 November 1925	Edward VII of the United Kingdom	Prince Albert Victor, Duke of Clarence and Avondale George V of the United Kingdom Louise, Princess Royal and Duchess of Fife Princess Victoria of the United Kingdom Maud, Queen of Norway Prince Alexander John of Wales

Figure 1.4: The first two rows of the table from the Wikipedia article for "Louise of Hesse-Kassel".

14 triples for *person-children* by recognizing the relationship between columns four and five (this time relating Princess Louise of Sweden to her 8 children and Edward VII of the United Kingdom to his 6 children).

Finally, because tables on Wikipedia are always contained in an article about one subject, we can also predict if there is a relationship between the article subject and a column in the table. The table in Figure 1.4 is taken from the article for "Louise of Hesse-Kassel" so we can also predict the relationship *person-children* between "Louise of Hesse-Kassel" and the first column, producing a final 2 triples (one for each row shown in the figure). Figure 1.5 shows a visualization of the knowledge graph extracted from the table in Figure 1.4. Due to space limitations, not all grandchildren of Louise of Hesse-Kassel are respresented.

In summary, our method takes a Wikipedia article containing a table as input and identifies relations between the various columns as well as between the article subject and any table columns. The output is a set of triples we create using the predictions and entities in the rows on the table which can then be added to an existing knowledge graph if they are not already present.

1.3 Contributions

In our attempt to answer the following research question:

> Can recent neural network technology designed for processing running text be used to predict relationships represented in tables?

our work produced two contributions to the problem of table understanding and the development of knowledge graphs. The first is a large corpus of over 200,000 tables annotated with high accuracy with column pairs or article subject - column pairs each representative of one of 28 relations or an abstention (explained in Section 3.2). This is the first public dataset of its kind that can be used to develop and compare relation extraction methods or other subtasks using tables [26].

The second contribution is a system which is able to predict with ∼87% accuracy a relation out of our 28 (plus a "no relation" prediction used for

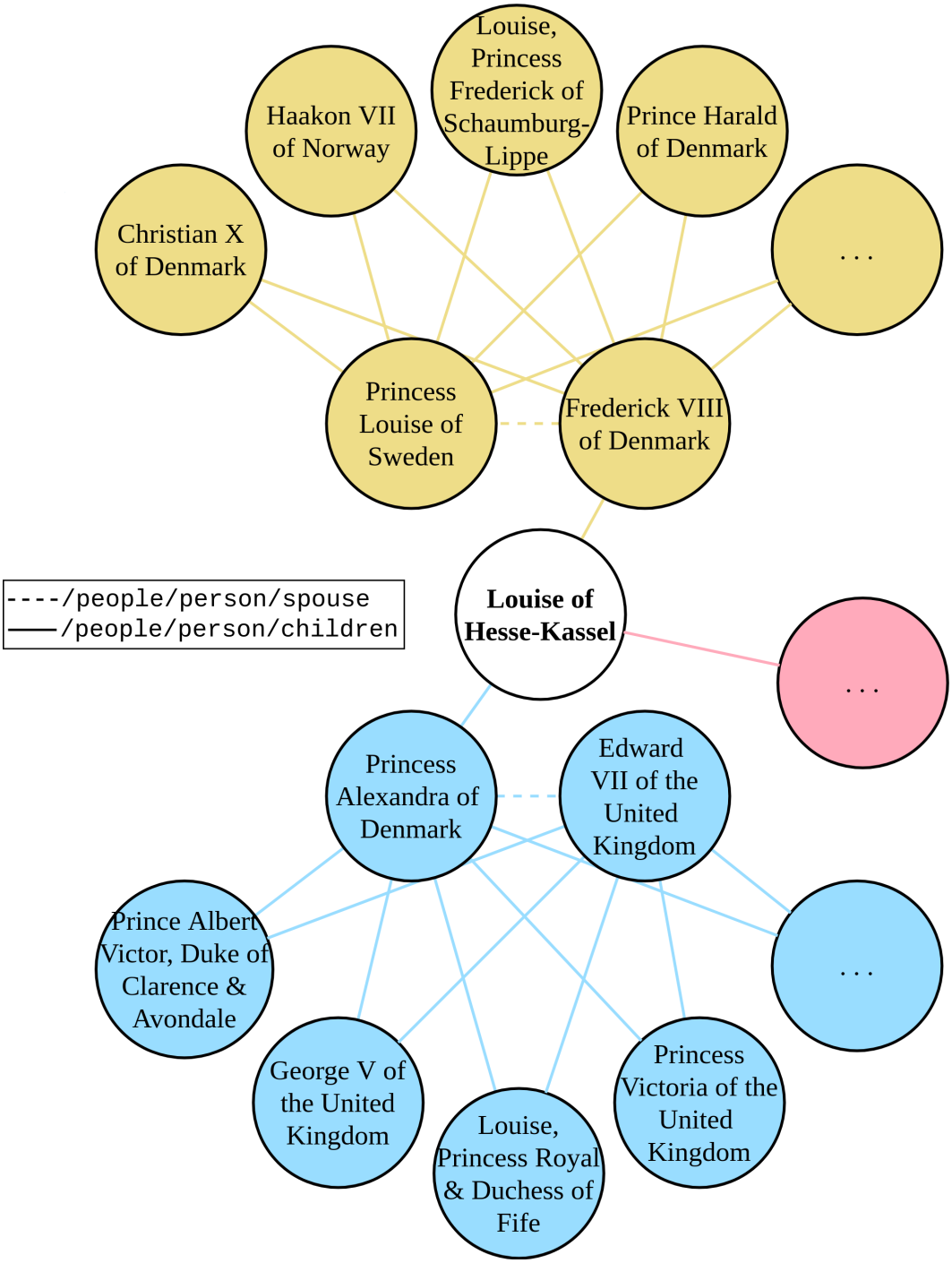

Figure 1.5: A portion of the knowledge graph extracted from the table in the Wikipedia article for "Louise of Hesse-Kassel".

abstentions) when given a new table (explained in Section 3.3). This is an improvement on earlier work in a number of respects. First, our accuracy is competitive with or better than related work (reported in Section 4.3), though it is difficult to compare due to the dissimilar test conditions. Our system is also able to predict a large number of relations compared to related work and predicts relations between columns and between columns and article subjects (discussed in Section 2.3). Finally, our system is able to add an estimated 582,691 triples to DBpedia from 46,077 tables.

1.4 Outline

This thesis begins with a discussion of related work including an introduction to neural networks and word embeddings followed by a discussion of previous research on relation extraction on text and table understanding. We then present the method in Chapter 3 in two parts. Section 3.2 explains how we gathered our corpus of tables using two methods to balance accuracy and coverage. Section 3.3 describes the structure of the network and how we created the input for our network given an article with a table as well as the format of the output. We present our experimental findings in Chapter 4. These experiments include a thorough evaluation of parameters as well as a comprehensive ablation study to determine which features contribute the most to accuracy and how BERT embeddings perform compared to an older set of pretrained word embeddings. Finally, we discuss conclusions, limitations, and directions for future work in Chapter 5.

Chapter 2

Related Work

This chapter outlines the work most closely related to this thesis. Because our method uses a neural network, specifically a long short-term memory or LSTM, we begin the chapter with a brief introduction to neural networks and the enhancements that led to LSTMs [17]. The method explained in this thesis also relies heavily on word embeddings so we then provide a description of word embeddings. Then, we discuss the datasets and previous methods employed for relation extraction, primarily on text. Next, we examine previous work in processing Wikipedia tables, some of which involves relation extraction, before finally explaining how this work blends the two and builds on recent research.

2.1 Neural Networks

Neural networks are not a recent development in computing [12]. In fact, a publication from 1943 could be considered the pioneering work in the field [27]. Interest in networks grew throughout the 1940's and 50's as computational power and researchers fascination increased following the second world war, but suffered a downturn in subsequent decades [2], [20], [40]. There has been a massive resurgence in research in neural networks in the last two decades and a considerable amount of pioneering work in computing science today employs some form of network [13], [16], [44].

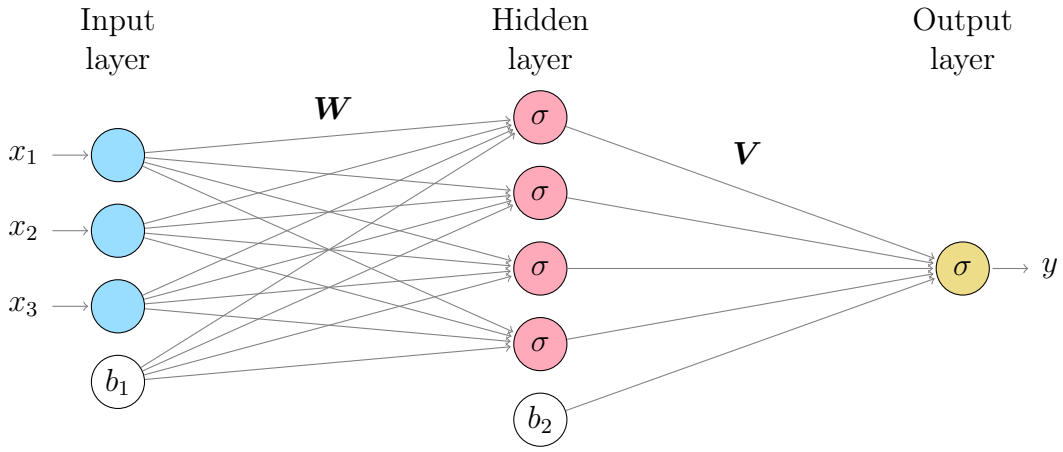

Figure 2.1: An MLP with one hidden layer consisting of four neurons showing the prediction of a single output value, y, given a vector, $[x_1, x_2, x_3]$, as input. These values are first multiplied by a weight matrix, $\boldsymbol{W}$, added to the bias b_1 then pass through the first activation function σ in the hidden later. Then, the result from the hidden layer is multiplied by another weight matrix, $\boldsymbol{V}$, and added to a second bias b_2. Finally, this value is passed through a second activation function σ in the output layer to produce the output value y.

Multilayer Perceptron

The simplest neural network is the multilayer perceptron (MLP) which consists of an input layer, at least one fully-connected hidden layer and an output layer [12]. A hidden layer in an MLP takes the input values (from either a previous hidden layer or the input layer) in the form of a vector $\boldsymbol{x}$ and multiplies it by a weight matrix, $\boldsymbol{W}$ [12]. Then, it adds a bias value and applies a nonlinear activation function to the result to produce an output value y [12]. Finally, this vector is fed forward to the next layer (either the next hidden layer or the output layer) [12]. The activation function, σ, transforms values to the range {0,1} or {-1,1} depending on the function used [12]. The values in the weight matrices are tuned by predicting on a set of labeled examples, computing the error with respect to the gold labels and backpropagating this information to update the weights using gradient descent or a similar optimization function [12].

Mathematically, each layer multiplies a weight matrix, $\boldsymbol{W}$, by the input vector, $\boldsymbol{x}$, adds a bias vector, $\boldsymbol{b}$ and transforms the resulting value y using some activation function, σ: $y = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ [12]. Note that biases can be

incorporated in a variety of ways as either a scalar, vector for the layer, or a vector for each node in the layer. In Figure 2.1, the bias is a scalar unique to each layer. MLPs and other similarly-structured networks are known as feed-forward neural networks because every layer receieves input only from the previous layer [12].

An example of an MLP is shown in Figure 2.1. MLPs, while fairly simple, are a powerful tool and have been proven to be able to approximate any continuous function arbitrarily well using just one hidden layer [18]. Weaknesses of MLPs include the high computational requirements and the propensity to overfit to the data, both of which are due to the high number of connections between layers. The most successful enhancements to MLPs specifically for natural language processing (NLP) are convolutional neural networks (CNN), recurrent neural networks (RNN) and modifications of RNNs such as long short-term memory networks (LSTM) which can be uni-directional or bi-directional all of which were used in top-performing relation extraction methods [22], [49]–[51]. Our work uses an LSTM to leverage the benefits of RNNs for NLP applications.

Recurrent Neural Network

Recurrent connections are a key development to neural networks and were introduced by Rumelhart et al. in 1986 [41]. Initially designed to manage data organized by time steps, sentences are another form of ordered data that is an appropriate input [12]. Recurrent connections are able to share parameters over multiple timesteps, using the hidden state of a previous timestep, $h^{(t-1)}$ as input to the next hidden state calculation [41]. Because RNNs use previous hidden states as input to later hidden states, they excel at dealing with problems where the output depends on the order of the input. The ability to use hidden states generated from earlier input is what makes RNNs so appropriate for NLP problems as the meaning of a word relies heavily on those that preceded it.

Some benefits of using recurrent connections rather than the full connections shown in Figure 2.1 include a reduction in the number of connec-

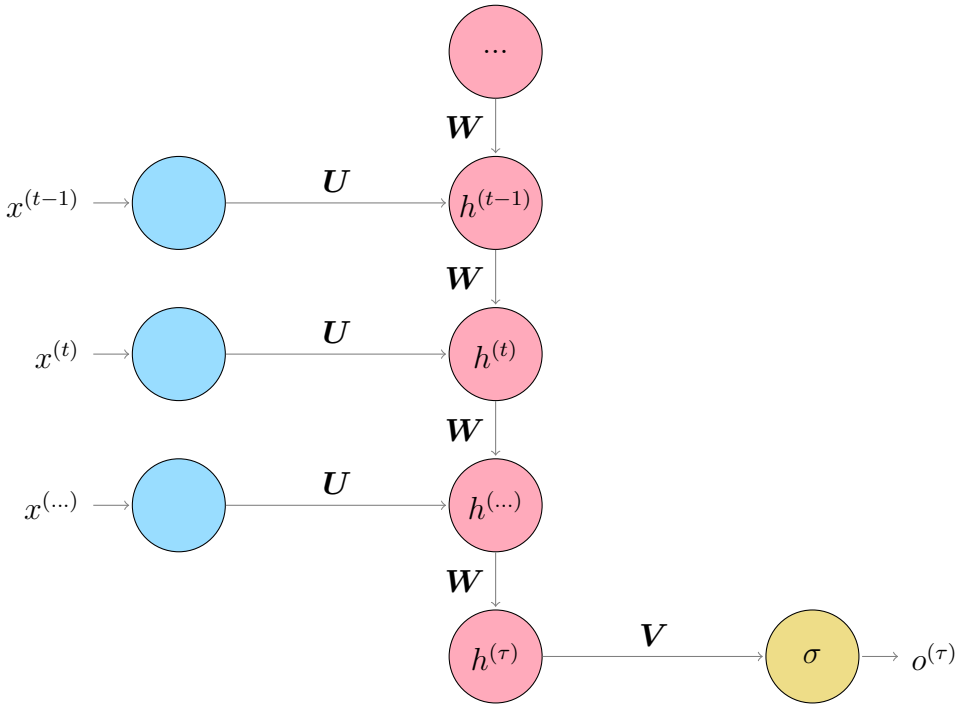

Figure 2.2: An illustration of a simple recurrent neural network. Input values $x^{(t)}$ are superscripted by their timestamp and multiplied by a weight matrix, $\boldsymbol{U}$. Note that this matrix is the same for each time index. This result is then combined with the previous hidden state, $h^{(t-1)}$ (which is multiplied by another a weight matrix $\boldsymbol{W}$). The result of these operations is sent as the input to the next hidden state $h^{(t+1)}$. These operations are repeated until we reach the last input, $x^{(\tau-1)}$. The result of $x^{(\tau-1)}Uh^{(\tau-1)}WW$ produces the final hidden state $h(\tau)$. $h(\tau)$ is combined with the final matrix $\boldsymbol{V}$ and sent through an activation function (like the sigmoid function shown here) to produce the output of the RNN $o^{(\tau)}$.

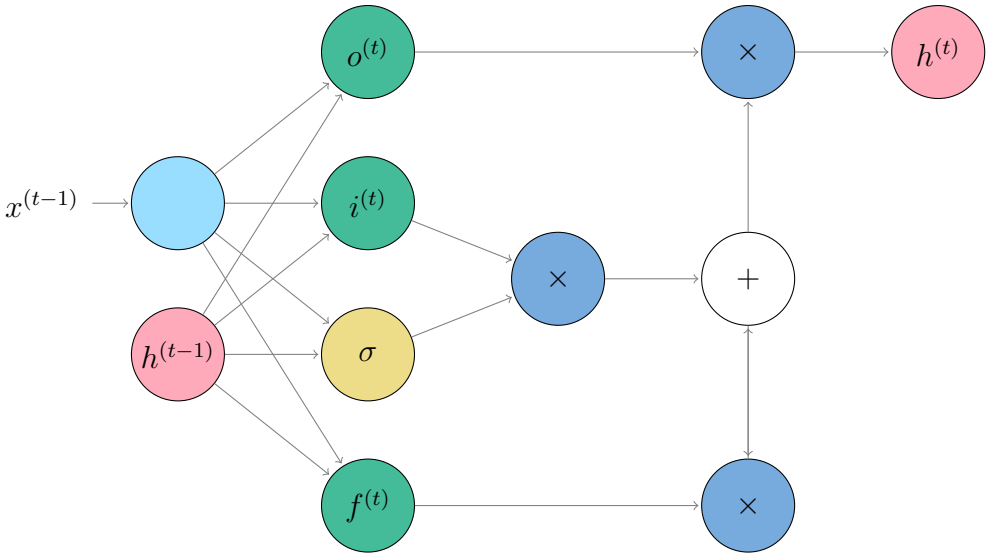

Figure 2.3: A simplified diagram of an LSTM. Bias values and weight matrices have been omitted from this figure for simplicity. Hidden states ($h^{(t)}$ and $h^{(t-1)}$) are coloured in pink with the previous timesteps hidden state used as input along with some parameter(s) $x^{(t-1)}$. Input, forget and output gates are coloured in green and are all sent to matrix multiplication operations (coloured in blue). The input and forget gates' outputs are used in the state (coloured in white) which then passes its output back to the matrix multiplication with the forget gate. Finally the output produces the next hidden state which will be used with the next input $x^{(t)}$.

tions and parameters and a heightened ability to process temporal data [12]. RNNs can also learn from fewer training examples because the weight matrices are shared across time steps [12]. Figure 2.2 shows an RNN with inputs $x^{(1)}, ..., x^{(t-1)}, x^{(t)}, ..., x^{(\tau-1)}$ and one output variable, $o^{(\tau)}$, generated at the end.

Long Short-Term Memory

The adaptation of the RNN to develop an LSTM layer was published in 1997 by Hochreiter and Schmidhuber who introduced self-loops so that the gradient could flow through paths for long times without the risk of increasing to infinity [17]. The addition of the forget gate to this architecture by Gers at al. in 2000 produced the LSTM we use today [10]. This forget gate conditions the weight of the self-loop on the context [12]. Figure 2.3 shows a somewhat simplified LSTM connection.

The main pitfall for neural methods is that they are supervised, mean-

ing they require large amounts of training data in order to accurately learn the weights of the network. This is a common problem for many supervised learning methods which require labeled samples to learn the best features or weights. Other issues with neural methods are their potential to overfit to the training data, that they are often computationally expensive and are effectively black boxes. This means that the weights that are learned provide no insight into the data and we cannot easily determine from them what information was the most useful for prediction. Recently, however, some researchers have been attemping to uncover the meaning of these matrices [21]. Despite this, neural methods are still the most successful and widely researched systems for NLP.

2.1.1 Embeddings

A major development in NLP in the last few years has been the introduction of word embeddings. In the last decade or so, many researchers have focused on this area [1], [8], [29], [35]. Word embeddings are an extension of much earlier work which attempted to represent words as vectors [1]. The goal of word embeddings is to create vectors for each word in a vocabulary by leveraging co-occurence statistics and other contextual information so that words with similar meaning and usage are close to one another in the vector space [1].

Most public systems for creating embeddings offer users the opportunity to either train their own embeddings or download a set of embeddings (referred to as pre-trained embeddings) which have already been computed using some corpus. Until recently, the state-of-the-art set of pre-trained word embeddings was GloVe [35] until work at Google developed an improvement, referred to as BERT [8]. BERT and other embedding models are used as a step in many NLP systems to convert text to meaningful vectors. We use BERT embeddings in our research to convert the text in Wikipedia articles to vectors as input for our neural network.

2.2 Relation Extraction on Text

Relation extraction is a well researched task in information extraction, which encompasses named entity recognition, coreference resolution, entity linking, and relation extraction defined in Section 1.2 [19]. In this section we explain the datasets used to compare different methods for relation extraction. We also discuss the methods applied to this task including statistcal and, more recently, a combination of statistical and neural.

2.2.1 Benchmarks and Metrics

Researchers have developed a number of datasets and benchmarks for relation extraction [15], [28], [38]. Traditionally, these contain a set of sentences each with a corresponding relation label [38]. Each example might also have additional information like annotated or linked entities [28], [38]. In addition to the New York Times [38] and SemEval-2010 Task 8 [15] datasets others include the TAC (text analysis conference) relation extraction [59], ACE [9], and KnowledgeNet [28] datasets, all of which contain thousands of instances of sentences with labeled relations.

In order to compare the results of different methods on the same dataset as accurately as possible, the same metrics for evaluation must be used. For relation extraction this frequently means either precision and recall, F1, or a combination of both[1]. The calculations for precision and recall (P and R respectively) are given in Equation (2.1) where "true positives" is the number of test cases where the system correctly predicted a relation in the benchmark, "false positives" is the number of cases where the system incorrectly predicted a relation and "false negatives" is the number of cases where the system did not predict a relation [19].

$$\begin{aligned} P &= \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \\ R &= \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \end{aligned} \tag{2.1}$$

[1] http://nlpprogress.com/english/relationship_extraction.html

Precision and recall can be calculated for systems where one relation is returned for an example by counting up the "true positives", "false positives", and "false negatives" and inserting them to the formula in Equation (2.1). In systems where relations are assigned a confidence value for each example, we can order all test cases in the dataset by the highest confidence value in each examples ranking and return only the first n of those relations [53]. This simulates predicting a relation on the first n cases and no relation on all others [53]. The number of those n relations that are correctly predicted according to the annotations in the benchmark is referred to as precision at n [53]. Some literature presents a precision-recall curve, created by adjusting the value of n to collect precision values at different recall levels [53].

$$F1 = \frac{2 * P * R}{P + R} \tag{2.2}$$

To compute F1, shown in Equation (2.2), different recall values from the precision-recall curve can be tested and the highest value for F1 from these is reported [19], [53]. This is also referred to as the harmonic mean as it attempts to find a balance between high precision (often sacrificing recall) as well as high recall (sacrificing precision) [19]. Accuracy is another metric that is used more often in neural methods and calculates the total number of correct predictions over all test cases.

2.2.2 Statistical Methods

Statistical methods for relation extraction (RE) generally involve using statistics gathered from labeled data to build a model for each relation and then finding the model that most closely resembles an input sentence [55]. To build the models for these relations, many researchers use some of the data from the benchmarks discussed above for training [9], [15], [38], [59]. Often, the accuracy of these methods is strongly correlated with the volume and quality of the training data [55]. Unfortunately, for training data to be as close to perfect as possible it has to be created by humans inspecting and labeling sentences, making it expensive to create and, therefore, relatively scarce [14]. Even with

well studied tasks like relation extraction, the need for more training data for supervised methods is critical [14].

Distant Supervision

To avoid this issue, many supervised methods use a form of bootstrapping to gather more labeled data for training [11], [30], [36], [47], [54]. The most common form of bootstrapping in relation extraction nowadays is distant supervision, first introduced by Mintz et al. in 2009 [31]. Distant supervision uses an existing knowledge graph and corpus of text to label sentences with relations [31]. In the work by Mintz et al., models are created for 102 relations on Freebase by gathering all pairs of entities related through a relation then annotating the Freebase Wikipedia Extraction (a dump of Wikipedia) with entity mentions and finding all sentences mentioning any of the pairs. These sentences are then assumed to be representative of that relation and are treated like the gold standard in the benchmarks explained above.

To create a model for each of these relations, Mintz et al. build a vector using features extracted from the sentences collected. Negative training data is also included for each relation by randomly selecting entity pairs not appearing in *any* Freebase relation. With a feature vector for each relation, a multi-class logistic classifier is built. At prediction, the classifier takes an entity pair and similar feature vector constructed from a sentence mentioning the new pair and returns a relation name and confidence score representing which relation the input sentence is most likely to represent.

The distant supervision algorithm explained by Mintz et al. manages to evade the need for large amounts of hand-labeled training data to build a useful classifier [31]. Although using only the algorithm presented by Mintz et al. achieves just under 40% precision at 10% recall on the New York Times dataset [25], the method is incredibly valuable for gathering additional labeled data for tasks with insufficient datasets [31]. This has been shown by numerous papers published which use distant supervision as part of a larger system to provide more labeled data [11], [30], [36], [47], [54]. Therefore, we apply a distant supervised approach to Wikipedia tables with features including

information about the article and table themselves.

2.2.3 Language Modeling

Previous work by Cannaviccio et al. [6] also used distant supervision to build language models for 12 Freebase relations using Wikipedia text annotated with entity mentions. Language models are built by first identifying sentences in Wikipedia that mention entity pairs related in Freebase by any of the relations being modeled. The phrases between each entity pair are first checked to ensure they are relational and cleaned by replacing specific terms and values (countries, monetary values, dates) with placeholders. Finally, the normalized phrases between the entities are saved to a list of phrases for their relation. The result of this method is a set of phrases for each relation assumed to be descriptive of that relation.

At prediction time, a sentence is assigned a score for each relation using Equation (2.3) where $c(p, r)$ is the count of phrase p in the language model for relation r.

$$\begin{aligned} score(p, r_i) &= \log c(p, r_i) \cdot P(r_i|p) \text{ where} \\ P(r_i|p) &= \frac{c(p, r_i)}{\sum_{j \in R} c(p, r_j)} \end{aligned} \tag{2.3}$$

The authors then apply this calculation to sentences on Wikipedia mentioning entities in DBpedia in order to identify new instances of the 12 relations not already in the knowledge graph [6].

The main pitfall for modeling methods is their inability to generalize to unseen data [1]. For example, the method by Cannaviccio et al. will not be able to predict a relation for a sentence containing a phrase never seen in the data as the two values for c would be 0 [6]. Our work applies a similar technique to Wikipedia tables but uses a neural network that converts the strings in features to vectors using word embeddings to preserve the meaning of the text. This method maps texts with similar subjects to vectors close to one another, overcoming the problem of generalization faced by language models [35].

2.3 Table Understanding

In recent years, some researchers have attempted to make the tables on the web more useful for search systems [4], [24], [39], [48]. Currently, most search engines only index the text in a web page and ignore information provided by tables [4]. This means users cannot search for the huge amount of data that is described in tables that would be tedious and repetitive to discuss explicitly in plain text [4]. Table understanding is a task parallel to information extraction where the input is a table, rather than text, explained in Section 1.2.

2.3.1 Early Work

NLP work on tables first began with the WebTables project which automated the conversion of HTML tables into relational databases [4]. WebTables also presented a few tools for automated schema matching between tables as well as automated joining of tables [4]. These two tools help consolidate similar information in different tables to provide the user with as much data as possible [4]. This work spawned a handful of other methods also aimed at annotating HTML tables on the web [24], [39], [48].

One hurdle that WebTables and similar methods had to overcome was the variation across HTML tables [52]. While the web is full of unregulated data, Wikipedia offers NLP researchers an informative, relatively noise-free corpus on which to develop methods [32]. The standards for the language and structure of articles as well as crowd-sourced editing on Wikipedia mean there are fewer errors and little ambiguity in articles[2]. Tables on Wikipedia are similarly monitored for structure and quality. This makes Wikipedia tables a more suitable canvas to develop methods for processing tables on a deeper level [3], [32]. As such, the most recent research in table understanding has been performed on Wikipedia tables [5], [7], [32].

WikiTables, the first such work in this field, presents a user application to simplify the exploration of tables on Wikipedia [3]. Bhagavatula et al. implement a table joining method similar to WebTables in addition to keyword

[2]https://en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines

search and a correlation mining task. WikiTables was able to take advantage of the cleanliness of Wikipedia tables and, while working with fewer tables than methods mining HTML tables on the web, presented a competitive system. This work helped to highlight the viability of Wikipedia as a source for tabular data.

More recently, research from the University of Stavanger has focused on returning tables as results of queries and auto-completing tables as they are created [56]–[58]. This improves existing search systems which only examine the text of documents, ignoring potential answers provided in tables [56]. The authors of these three papers take advantage of existing knowledge graphs and a corpus of tables to train a ranking system for the different types of suggestions (tables or cell values).

2.3.2 Relation Extraction on Tables

Some research in table understanding focuses on relation extraction on Wikipedia tables, the topic of this thesis [5], [7], [32]. The research done by Muñoz et al. was the first to study relation extraction from tables [32].

The system created by Muñoz et al. [32] suggests relationships between table columns and between columns and the article subject when pairs of the same row are already related in an existing knowledge graph. Suggestions are then filtered using a classifier by analyzing features of the article, table, columns, entities, cells and resulting triples. The authors evaluate their work on a dump of Wikipedia's tables by manually annotating 750 of the around 37 million suggested triples using three judges. Using only triples for which there was a unanimous agreement amongst judges, five classifiers were trained, evaluated and compared, with the best achieving close to 80% F1 (81% precision and 77% recall) and producing almost eight million new triples for over 30,000 relations.

This work shows that a simple, intuitive method can provide a vast amount of (mostly) accurate information to a knowledge graph. Our work uses a similar dataset (Wikipedia tables) to create the same output (RDF triples) for the same purpose (knowledge graph augmentation). Therefore, although

we used a different dump of Wikipedia tables, we loosely compare our final results with Muñoz et al. showing that our system out-performs their best classifier, which scored 78% accuracy, by over 10%.

Following Muñoz et al., a group of researchers at Roma Tre University in Italy and the University of Alberta in Canada have published three methods for relation extraction on tables [5], [7], [43]. All of these methods use language models, an existing knowledge graph and an additional text corpus (Clueweb) to determine the relationships in tables. Each entity pair in a table is scored against a model for every relation and the highest-scoring relation is selected. The highest F1 value reported in one paper is 71% [5] while the most recent work reports on a different metric (mean reciprocal rank) [7]. We compare with this work though we use a different dataset and our metrics are not the same and report higher precision and recall.

All the methods described above that perform relation extraction on Wikipedia tables use language modeling and statistical information to predict relationships [5], [7], [32], [43]. To our knowledge, our method is the first to use a neural network.

Chapter 3

Method

3.1 Datasets

Because our method is supervised, we required a large corpus of annotated Wikipedia articles containing tables. Since no such dataset exists publicly, we built our own using distant supervision with an existing knowledge graph. We used DBpedia[1] and Freebase[2] as the knowledge graphs to collect entity pairs known to be related to one another so we could search for tables mentioning these entities in the same rows and infer a relation. In total, we collected entity pairs for 28 Freebase relations using Freebase as well as equivalent DBpedia relations.

For gathering our corpus of tables, we used a dump of Wikipedia from March 2019. From this dump, we identified all articles that contained a table object and parsed the table into JSON (referred to as our Wikipedia parser). Information extracted from the table included table headers, captions and cell values. We also saved information in the article including the title, type, abstract, title of the section the table appeared in and any text appearing in that section. Portions of the article which were saved are illustrated in Figure 3.1.

[1] https://wiki.dbpedia.org/
[2] https://developers.google.com/freebase/

Louise of Hesse-Kassel

From Wikipedia, the free encyclopedia

Louise of Hesse-Kassel (German: *Luise Wilhelmine Friederike Caroline Auguste Julie von Hessen-Kassel*, Danish: *Louise Wilhelmine Frederikke Caroline Auguste Julie*; 7 September 1817 – 29 September 1898) was Queen of Denmark by marriage to King Christian IX of Denmark.[1]

Contents [hide]

Louise of Hesse-Kassel

Queen consort of Denmark

Tenure	15 November 1863 – 29 September 1898

Children [edit]

Louise had the following six children with Christian. Eventually, they had thirty-nine grandchildren.

Name	Birth	Death	Spouse	Children
				Christian X of Denmark Haakon VII of Norway Louise, Princess Frederick of Schaumburg-Lippe Prince Harald of Denmark
Frederick VIII of Denmark	3 June 1843	14 May 1912	Princess Louise of Sweden	Princess Ingeborg, Duchess of Västergötland

Figure 3.1: Article fields saved by the parser for the Wikipedia article for "Louise of Hesse-Kassel". Each dashed red box contains one field we saved. These include (from top to bottom) the title of the article, abstract, section title, section paragraph, table headers, and cell values. Note that this table does not have a caption.

3.1.1 Relations

For the set of relations we chose to model, we combined those used in two related works. The first of these was Mintz et al., 2008 which used the 102 largest Freebase relations at the time for building models and prediction [31]. The authors provide a table listing the 23 largest of those 102, which we use in this work. We also used the relations provided by Cannaviccio et al., 2018 who create language models for 9 Freebase relations [7]. After removing duplicates for relations appearing in both sets, one of each reflexive relations (e.g. we had *person-parents* and *person-children* and kept only *person-parents*), and those no longer present in Freebase, the result is 24 relations. We also add 4 more relations (*actor-character*, *film-production_company*, *film-music*, *actor-film*) not previously researched. We chose to add more relations for films as there are an abundance of Wikipedia tables concerning films. The final result is 28 relations which are given in Table 3.1.

3.2 Table Collection

To collect tables from the Wikipedia corpus that contained a pair of columns related through one of the 28 relations, we used two methods. The first was a modification of distant supervision using existing knowledge graphs, explained in Section 3.2.1. The second was querying using headers, section titles and other attributes identified to be indicative of one of the relations. This is explained in Section 3.2.3.

3.2.1 Using Distant Supervision

We first used a form of distant supervision to collect training and testing data. This involved gathering a list of entity pairs (e_1, e_2) that were related through a relation r from our set of 28 relations. The number of pairs was highly variable across our set of relations so we sought to find pairs another way. We settled on identifying the equivalent DBpedia relations for each Freebase relation and collecting those entity pairs as well. DBpedia is another public knowledge much like Freebase but containing slightly different information due to its

Relation	Freebase Pairs	DBpedia Pairs	Total
person-place_ of_ birth	1,048,577	955,617	1,613,709
person-nationality	1,163,196	51,665	1,196,259
location-contains	433,219	771,909	1,056,940
sports_ team-player	120,407	714,001	750,411
person-profession	532,612	0	532,612
film-country	470,722	25,298	476,745
person-place_ of_ death	336,859	245,846	475,669
actor-film	0	283,702	283,702
film-genre	270,947	0	270,947
musician-album	131,119	60,124	142,252
person-spouse	128,643	22,490	137,575
film-language	124,038	34,554	132,879
musician-origin	88,610	51,053	119,877
director-film	63,928	87,124	99,484
writer-film	39,489	84,430	98,590
person-graduate	0	98,150	98,150
producer-film	33,360	57,982	69,265
political_ party-politician	0	65,420	65,420
book-genre	53,969	20,938	62,459
person-parents	54,512	20,350	58,090
person-religion	27,873	29,060	50,046
film-music	49,144	0	49,144
author-works_ written	32,747	31,454	41,841
company-industry	0	38,112	38,112
football_ position-player	7,259	29,480	36,739
film-production_ company	11,979	1,636	13,573
actor-character	5,812	4,028	9,221
award-nominee	0	223	223

Table 3.1: Total number of entity pairs collected.

different construction method [23]. Not only did this increase the number of entity pairs collected, but DBpedia includes more recent data whereas the pairs we gathered from Freebase only contained information up to 2015. Table 3.1 shows for each relation the number of entity pairs collected from Freebase, the number of pairs collected from DBpedia and the number of distinct pairs obtained once the two were combined.

Then for each entity pair (e_1, e_2) collected for a relation r we searched for Wikipedia tables with e_1 and e_2 appearing in cells of the same row but different columns. We then assume that relation r holds between the two columns. We also searched for Wikipedia tables with either e_1 or e_2 as the subject of the

Relation	# Tables	Accuracy
sports_team-player	20,028	84%
person-place_of_birth	16,146	3.5%
person-nationality	14,297	64%
political_party-politician	9,010	78%
location-contains	7,747	89%
director-film	5,874	80%
writer-film	5,357	5.5%
musician-album	4,988	95%
actor-film	4,483	72%
producer-film	3,799	9.5%
person-spouse	2,487	15%
film-language	2,236	89%
football_position-player	2,120	100%
film-music	1,333	74%
film-genre	1,114	9.0%
person-place_of_death	1,070	6.0%
author-works_written	959	57%
film-production_companies	602	60%
actor-character	575	48%
film-country	572	73%
person-parents	390	27%
musician-origin	299	49%
person-profession	284	71%
company-industry	106	54%
person-graduate	83	37%
book-genre	28	78%
person-religion	18	78%
award-nominee	4	25%

Table 3.2: Statistics on how many tables were collected with column pairs for each relation.

article and the other entity appearing in the cell of a table in the article. We then assumed the article subject was related to the corresponding column through relation r. The total number of tables returned by these queries are shown in Table 3.2 and Table 3.3. The first column lists a relation, the second gives the number of tables returned on average for each pair. The last two columns show the total number of tables returned for each relation as well as the estimated accuracy for this method. The accuracy was computed by annotating a random sample of (up to) 200 tables per relation.

Relation	# Tables	Accuracy
actor-film	20,369	93%
sports_ team-player	3,900	79%
musician-album	3,755	88%
director-film	3,712	55%
writer-film	3,658	33%
location-contains	3,312	99%
producer-film	2,774	30%
actor-character	2,729	49%
person-place_ of_ birth	2,008	0.0%
person-nationality	1,397	0.0%
film-music	783	15%
person-profession	582	55%
political_ party-politician	439	100%
person-spouse	407	4.5%
author-works_ written	331	79%
person-place_ of_ death	202	0.0%
film-country	125	0.0%
person-parents	75	1.4%
person-graduate	70	74%
film-language	30	67%
film-production_ companies	15	0.0%
company-industry	11	18%
award-nominee	8	71%
film-genre	7	14%
person-religion	2	0.0%
football_ position-player	1	100%
book-genre	0	–
musician-origin	0	–

Table 3.3: Statistics on how many tables were collected with article subject-column pairs for each relation.

Distant Supervision Errors

The estimated accuracy for many relations posed an issue for this method whch we managed to ameliorate using binary classifiers. The root of the problem with our distant supervision method can be illustrated by the table shown in Figure 3.2. This table lists films in the "Monsters, Inc." franchise along with their directors, producers, writers, composers and editors. There are multiple examples of directors who were also writers in this table. For example Dan Scanlon both directed and wrote Monsters University. These two triples are also in DBpedia (not Freebase) which means when we queried our tables for the pair ("Dan Scanlon", "Monsters, Inc.") for both *film-director* and *film-writer* we returned a relation between columns one and two and columns one and five. Therefore, our data for *film-director* and *film-writer* both included a pair of columns not actually related through these relations. This is because entities are frequently related to one another through more than one relation and is therefore an unavoidable problem when using distant supervision. This presents a bigger problem for some relations more than others and accounts for almost all of the erroneous tables annotated in Table 3.2 and Table 3.3.

3.2.2 Classification for Error Reduction

Therefore, to clean up the tables collected we experimented with classifiers to remove tables incorrectly annotatated. We chose relations which had an accuracy below 85% to clean up with the classifiers to not risk removing too many correctly annotated tables from those relations which are more accurate.

We created a feature vector for each table comprised of the column numbers of the subject and object columns, number of entity pairs in the columns present in Freebase or DBpedia, total number of pairs in all rows and percent of pairs in the table in the database. To convert the header text, section title, section text, and article title to vectors, we used the python library `spaCy`[3] which has pre-trained embeddings for about a million words. We experimented using two different lengths for our `spaCy` vectors for each field. For what we

[3]https://spacy.io/

Crew [edit]

Film	Director(s)	Producer(s)	Executive Producer(s)	Writer(s)
Monsters, Inc.	Pete Docter *Co-directed by:* Lee Unkrich & David Silverman	Darla K. Anderson	John Lasseter & Andrew Stanton	*Screenplay by:* Andrew Stanton & Daniel Gerson *Original Story by:* Pete Docter, Jill Culton, Jeff Pidgeon & Ralph Eggleston
Monsters University	Dan Scanlon	Kori Rae	John Lasseter, Pete Docter, Andrew Stanton & Lee Unkrich	*Screenplay by:* Daniel Gerson, Robert L. Baird & Dan Scanlon *Story by:* Dan Scanlon, Daniel Gerson & Robert L. Baird

Figure 3.2: The table in the Wikipedia article of "Monsters Inc. (franchise)".

Relation	kNN (k=1)	kNN (k=2)	NC	GNB
actor-character	**0.75**	0.60	0.53	0.51
actor-film	**0.94**	0.86	0.83	0.79
author-works_written	0.75	**0.76**	0.53	0.60
director-film	**0.90**	0.89	0.59	0.83
film-country	**0.86**	0.83	0.80	0.77
film-genre	0.31	0.0	0.18	**0.86**
film-music	**0.85**	**0.85**	0.48	0.81
film-production_company	0.68	0.61	0.43	**0.71**
musician-origin	**0.64**	0.59	0.58	0.34
person-nationality	**0.93**	0.91	0.90	0.88
person-parents	0.74	**0.75**	0.60	0.74
person-place_of_death	**0.50**	0.0	0.15	0.37
person-profession	**0.80**	0.74	0.53	0.77
person-spouse	0.34	0.22	0.33	**0.67**
political_party-politician	**0.88**	0.84	0.47	**0.88**
producer-film	**0.25**	0.0	0.24	0.24
sports_team-player	**0.95**	0.94	0.93	**0.95**
writer-film	**0.17**	0.0	0.16	0.12
Average	**0.69**	0.61	0.54	0.67

Table 3.4: F1 values of classifiers on column pair tables using 5,105-dimensional vectors.

refer to as short vectors we used the first 3 words of the headers, section title and article title and the first 5 words of the section paragraph. For long vectors, we used the first 8 words of the article title, first 3 words of each header, 4 words from the section title and 14 of the section paragraph. When using the 300-dimensional `spaCy` vectors, this created 5,105-dimensional vectors for short vectors and 9,605 for the long vectors.

Once these vectors were built we took 100 annotated tables to train the classifiers and evaluated using another 100 annotated tables. We chose 3 classifiers to experiment with - k-Nearest Neighbours (with $k = 1$ and $k = 2$), Nearest Centroid and Gaussian Naïve Bayes. These classifiers tend to perform well on NLP problems and are simple to train and interpret. Each classifier chose to keep or discard an input table based on the provided feature vector. We evaluated the classifiers by computing the F1 score of the tables kept by the classifier. In this context, "true positives" were the tables correctly kept, "false positives" were tables kept incorrectly, and "false negatives" were tables dis-

Relation	kNN (k=1)	kNN (k=2)	NC	GNB
actor-character	**0.72**	0.65	0.56	0.28
actor-film	**0.91**	0.81	0.83	0.80
author-works_ written	**0.79**	0.74	0.60	0.69
director-film	**0.91**	0.87	0.60	0.82
film-country	**0.86**	0.83	0.80	0.58
film-genre	0.17	0.0	0.18	**0.80**
film-music	0.85	**0.86**	0.48	0.83
film-production_ company	**0.78**	0.64	0.43	0.61
musician-origin	**0.72**	0.60	0.58	0.55
person-nationality	**0.91**	0.88	0.90	0.85
person-parents	0.75	**0.76**	0.60	0.54
person-place_ of_ death	**0.38**	0.0	0.15	0.25
person-profession	**0.80**	0.69	0.53	0.63
person-spouse	**0.46**	0.0	0.33	0.42
political_ party-politician	**0.88**	0.84	0.49	0.75
producer-film	0.23	0.17	**0.26**	0.18
sports_ team-player	0.93	0.93	0.93	**0.96**
writer-film	**0.44**	0.0	0.16	0.10
Average	**0.70**	0.60	0.55	0.61

Table 3.5: F1 values of classifiers on column pair tables using 9,605-dimensional vectors.

Relation	kNN (k=1)	kNN (k=2)	NC	GNB
actor-character	**0.86**	0.77	0.74	0.72
author-works_ written	**0.87**	0.80	0.68	0.78
director-film	0.66	0.42	0.62	**0.67**
film-music	0.74	0.64	**0.76**	0.38
musician-album	**0.98**	0.95	0.89	0.95
person-profession	0.53	0.27	**0.72**	0.54
producer-film	**0.49**	0.31	0.30	0.41
sports_ team-player	0.92	0.89	0.88	**0.96**
writer-film	0.24	0.15	**0.53**	0.39
Average	**0.72**	0.62	0.68	0.66

Table 3.6: F1 values of classifiers on article subject-column pair tables using 5,105-dimensional vectors.

Relation	kNN (k=1)	kNN (k=2)	NC	GNB
actor-character	**0.87**	0.75	0.74	0.70
author-works_written	**0.86**	0.80	0.68	0.62
director-film	**0.65**	0.40	0.62	0.44
film-music	0.52	0.0	**0.70**	0.44
musician-album	**0.97**	0.94	0.89	0.95
person-profession	0.51	0.34	**0.72**	0.48
producer-film	**0.47**	0.28	0.30	0.37
sports_team-player	**0.90**	0.84	0.88	0.69
writer-film	0.27	0.10	**0.53**	0.34
Average	**0.69**	0.51	0.67	0.57

Table 3.7: F1 values of classifiers on article subject-column pair tables using 9,605-dimensional vectors.

carded that should have been kept. We trained and ran classifiers separately for tables with column pairs and tables with article subject-column pairs.

The results of these experiments are given in Tables 3.4, 3.5, 3.6, and 3.7. The tables show that for both groups, kNN with $k = 1$ performed best. Longer vectors seemed to provide some extra information for the column pairs, while the shorter vectors performed better for the article subject-column pairs. kNN likely outperformed nearest centroid because the two classes form multiple different clusters based on differently formatted tables which the nearest centroid can not manage as it computes an average for the whole class.

3.2.3 Querying

We also wrote a number of queries to find tables for each relation which may have no entity pairs in any rows stored in Freebase or DBpedia. Most queries looked for table headers and section titles that were identified through inspection of numerous tables to be characteristic of one of the relations. For example, the pair of headers "Name" and "Children" (shown in our running example, the Wikipedia table for "Louise of Hesse-Kassel") relates children with their parents. The results of running over 150 queries and annotating up to 100 tables for each of the 28 relations are shown in Table 3.8. We provide a full list of these queries in section A.3. These results gave us far fewer tables but with a much higher accuracy meaning we did not need to apply a classifier

after collecting them.

For the final dataset, we also needed samples of tables not representative of any relation. To accomplish this, we also queried for random tables, randomly selected a pair of column numbers (including the article title) and added this to our negative samples. This step assumes that most randomly selected tables will not be indicative of one of our relations, a common, albeit imperfect, assumption used when selecting negative samples for training classifiers [31], [42].

We finally combined the tables collected by distant supervision with those collected by querying for headers and section titles and present the numbers in Table 3.9. These are the tables we use to train and evaluate our neural network for relation prediction. This dataset also represents one major contribution of our work - a large set of annotated tables for relation extraction on Wikipedia tables[4].

We also gathered statistics on how many tables actually contained the fields we were saving. Table 3.10 gives a breakdown by relation of what percent of tables contained a caption, headers, abstract and section paragraph. We concluded that tables that didn't have an abstract were likely a bug in the Wikipedia parser caused by irregularities in the Wikipedia markup as all articles on Wikipedia contain an abstract. Tables without headers were also mostly as a result of the parser not handling the markup properly. Tables without either a caption or a section paragraph are likely intentional as these are not a requirement for tables on Wikipedia and are often omitted. This is indicative of how much contextual information the LSTM is provided with to deduce relations.

3.3 Prediction

The next step in our method was using the tables we collected to train a neural network to perform relation prediction on an unseen table. For our neural

[4]The dataset of tables in JSON format and of entity pairs collected from Freebase and DBpedia (in CSV) is available for download here: https://doi.org/10.7939/DVN/SHL1SL [26]

Relation	Column Pairs		Article Subject-Column Pairs	
	# Tables	Accuracy	# Tables	Accuracy
sports_team-player	22,041	98%	61	100%
political_party-politician	17,969	99%	45	100%
football_position-player	13,076	87%	0	–
person-nationality	7,002	100%	0	–
musician-album	5,528	100%	3,032	91%
location-contains	5,384	100%	737	98%
award-nominee	5,725	100%	0	–
person-graduate	5,408	100%	0	–
director-film	3,816	99%	688	88%
film-language	2,952	100%	0	–
author-works_written	1,372	89%	340	90%
actor-film	1,231	96%	26,946	98%
person-profession	716	97%	0	–
musician-origin	632	100%	0	–
producer-film	456	99%	508	92%
film-music	399	97%	0	–
film-country	380	100%	0	–
film-production_company	275	98%	0	–
person-parents	272	100%	459	96%
film-genre	203	94%	0	–
writer-film	199	80%	0	–
person-place_of_birth	180	100%	0	–
company-industry	102	100%	0	–
person-spouse	90	100%	0	–
actor-character	57	88%	21,405	97%
person-religion	42	100%	0	–
book-genre	9	89%	0	–
person-place_of_death	4	100%	0	–

Table 3.8: Number of tables collected and estimated accuracy of query method.

Relation	Distant Supervision	Querying	Total	Accuracy
sports_team-player	21,440	22,102	38,897	96%
actor-film	22,580	28,177	38,319	96%
political_party-politician	8,642	18,014	25,965	96%
actor-character	1,724	21,462	22,157	96%
location-contains	11,059	6,121	15,682	95%
football_position-player	2,121	13,076	14,360	89%
musician-album	8,049	8,560	14,058	96%
person-nationality	8,865	7,002	13,433	95%
No relation	–	–	12,491	–
director-film	7,019	4,504	9,067	89%
award-nominee	6	5,725	5,730	100%
person-graduate	83	5,408	5,486	100%
film-language	2,256	2,952	3,522	95%
author-works_written	588	1,712	2,031	87%
producer-film	668	964	1,526	75%
writer-film	1,036	199	1,233	40%
film-music	932	399	1,137	86%
person-profession	420	716	1,129	88%
person-parents	122	731	801	94%
film-country	474	380	760	92%
musician-origin	112	632	743	96%
film-production_company	446	275	654	81%
person-spouse	305	90	390	58%
film-genre	8	203	208	91%
person-place_of_birth	0	180	180	100%
company-industry	59	102	137	100%
person-place_of_death	133	4	137	40%
person-religion	14	42	49	100%
book-genre	22	9	31	97%

Table 3.9: The total number of tables in our dataset for each relation split by collection method. Note that the total number of tables given in column four can be less than the sum of distant supervision and querying tables due to overlap. The estimated accuracy is given in the last column.

Relation	Abstract	Caption	Paragraph
actor-character	95%	16%	0%
actor-film	93%	15%	1%
author-works_ written	97%	9%	21%
award-nominee	98%	2%	9%
book-genre	100%	3%	0%
company-industry	98%	9%	39%
director-film	95%	4%	9%
film-country	98%	2%	27%
film-genre	99%	8%	10%
film-language	96%	1%	8%
film-music	96%	1%	16%
film-production_ company	98%	6%	15%
football_ position-players	90%	1%	5%
location-contains	97%	11%	27%
musician-album	96%	8%	6%
musician-origin	99%	2%	24%
No relation	95%	7%	19%
person-nationality	95%	1%	10%
person-graduate	86%	7%	9%
person-parents	89%	1%	24%
person-place_ of_ birth	96%	7%	27%
person-place_ of_ death	97%	5%	35%
person-profession	95%	4%	20%
person-religion	97%	4%	42%
person-spouse	89%	3%	22%
political_ party-politician	98%	5%	19%
producer-film	96%	10%	4%
sports_ team-player	96%	4%	11%
writer-film	97%	5%	4%

Table 3.10: Statistics on what fields were present in the tables collected.

network we use an LSTM, an architecture which has been shown to perform well in NLP problems [22], [50], [51]. Our LSTM consisted of an input layer which took what we refer to as article embeddings (discussed in Section 3.3.1) built from the table and BERT word embeddings. We tune weights using specificity and sensitivity rather than accuracy because our classes (relations) have such skewed samples making accuracy impractical in this case. The input was then sent into an LSTM layer which outputted a 29 dimensional vector. This was followed by a dense layer and the output layer both inputting and outputting 29 dimensions. The structure of our prediction method is shown in Figure 3.3.

We explain in the following two sections in more detail what input the network reads followed by the output vector.

3.3.1 Input

Starting with an article and a pair of column numbers, we first created a vector to represent the input. To begin we tokenized the headers, section title, section text and table caption. We refer to these as the fields. We then concatenated the BERT vectors for all the fields. If no BERT vector existed for a term, we generated a random vector so as not to skew the vector too far in any dimension. Then, we padded each of the 4 fields to a specified length (which we experiment with in Section 4.1.1) and combined all the BERT vectors into a single vector for the article.

3.3.2 Output

Once the input vector had been sent through the LSTM later, dense layer, and the activation function, the output was a 29-dimensional vector. Each index of this vector represented one of our 28 relations or no relation. The value for a given index was the probability the network predicted for the table representing that relation. To predict a relation for the input table, we chose the most probable relation predicted (including no relation). Once a relation was predicted, we could then create tuples for the rows of the input table.

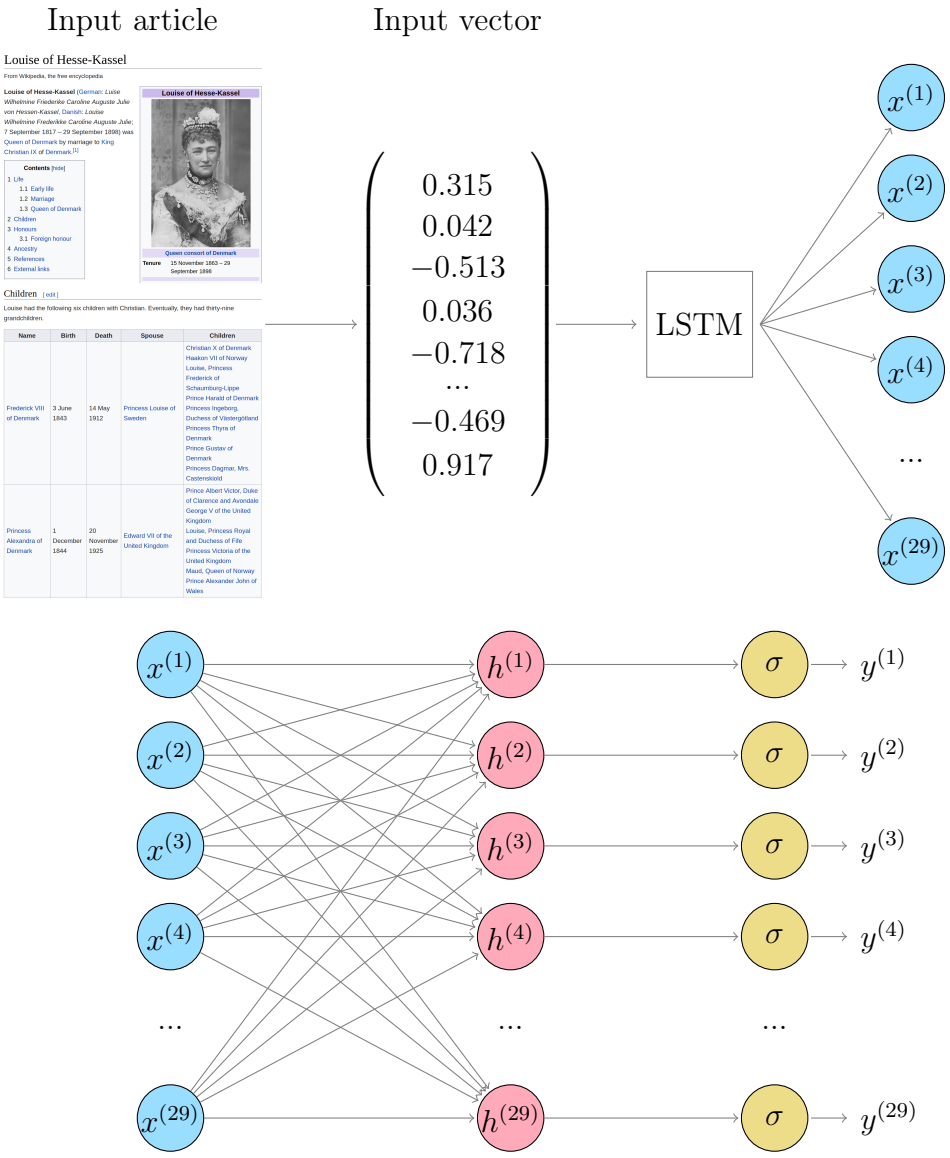

Figure 3.3: The architecture of our prediction model. Starting with an unseen article containing a table, we use BERT embeddings taken from the table headers, caption, section title and section text to create a vector for the article. This input vector is sent first to an LSTM unit followed by a dense layer. Each value is then sent through an activation function. The output is a 29-dimensional vector. Each value in the vector corresponds to a predicted probability that the input article and column numbers are representative of the relation represented by that index of the vector.

Chapter 4

Experiments

Our experiments included parameter tuning to optimize our neural network (in Section 4.1) followed by a comparison with a baseline explained in Section 4.2. We then ran some ablation studies (Section 4.3.1) and performed an error analysis in Section 4.3.2. We tuned our parameters on a development set comprising of 40% of our data or 92,130 tables. When the best set of parameters was found, we then trained a new network on another 40% of the data (92,118 tables) and tested on the remaining 20% (46,077 tables). We report the results of this final network along with two related works (Muñoz et al. and Cannaviccio et al.) in Section 4.3.

4.1 Parameter tuning

Using 40% of our data for development, our first step was to find an optimal set of parameters for our network. Because of the imbalance in class samples, we trained the network using sensitivity and specificity instead of accuracy. Parameters to tune included the vector length mentioned in Section 3.3.1 and parameters for the network. Network parameters were the loss function, optimizer, activation function, number of epochs, batch size, and learning rate. For each parameter setting, we created five networks using different splits of the provided development data for training (80%) and testing (20%). We report the accuracy for each parameter setting by computing the average of these five in Section 4.1.1.

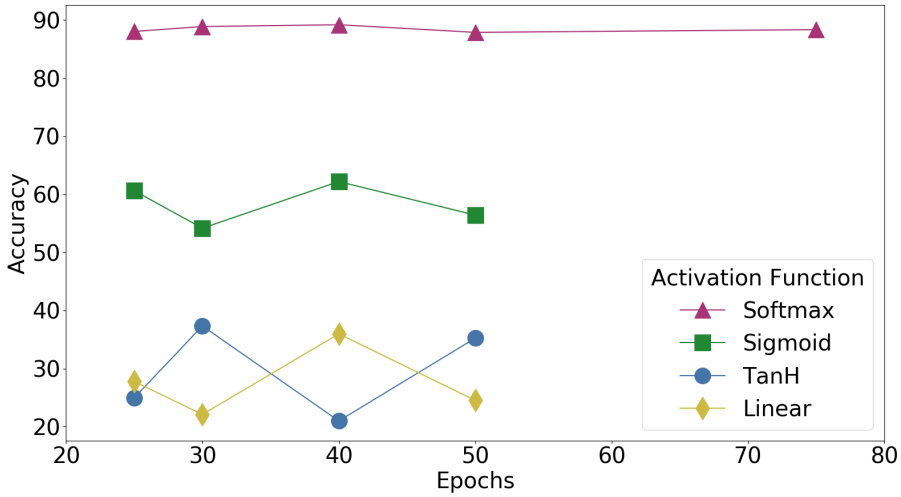

Figure 4.1: Accuracy scores for four different activation functions. Other parameter settings were categorical crossentropy for the loss function, adam as an optimizer, batches of 128 samples, a learning rate of 0.001, and a vector length of 19,968.

4.1.1 Results

We tuned parameters sequentially starting with the activation function and report the accuracy of the network for five different epoch values.

Activation Function

Using defaults for the other parameters, Figure 4.1 shows how using softmax, tanh, sigmoid and linear activation functions each fared when trained for 25, 30, 40, 50, and 75 epochs. Based on the results of this test we concluded that softmax was the activation function best suited to this task. Softmax managed to outperform every other function consistently for each epoch value.

Loss Function

Using softmax as the activation function, we then tested the loss function. Here, the options were categorical crossentropy (tested above as the default), binary crossentropy and Kullback-Leibler divergence (KL divergence). Figure 4.2 shows how the three functions performed. Analysing this figure, the results were slightly less clear than those obtained when tuning the activation

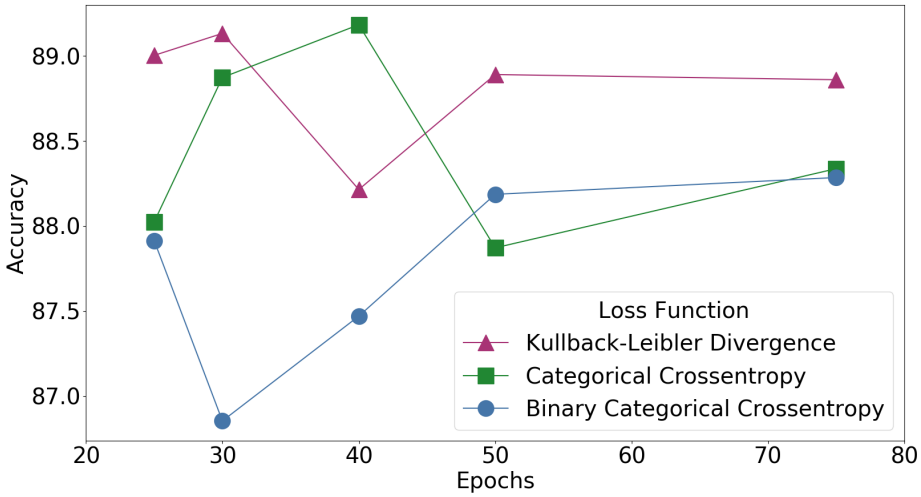

Figure 4.2: Accuracy scores for four different loss functions using softmax as the activation function. Other parameters were adam as an optimizer, batches of 128 samples, learning rate of 0.001, and a vector length of 19,968.

function (note the difference in scale), though binary categorical crossentropy seemed to perform slightly worse than the other two. Considering this, we performed the next set of experiments using both KL divergence and categorical crossentropy.

Optimizer

Our options for optimizers were adam, rmsprop, adagrad and adamax. The experiments are shown in Figure 4.3 with solid lines denoting the use of KL divergence as the loss function and dashed lines the use of categorical crossentropy. The four colours correspond to the four optimizers.

This comparison led us to a few conclusions. First, adagrad and adamax performed worse than adam and rmsprop regardless of loss function. Second, there was not notable difference between the accuracy values of adam and rmsprop using either loss function. With this in mind, we chose the pairing that produced the highest accuracy for further experiments, categorical crossentropy and rmsprop.

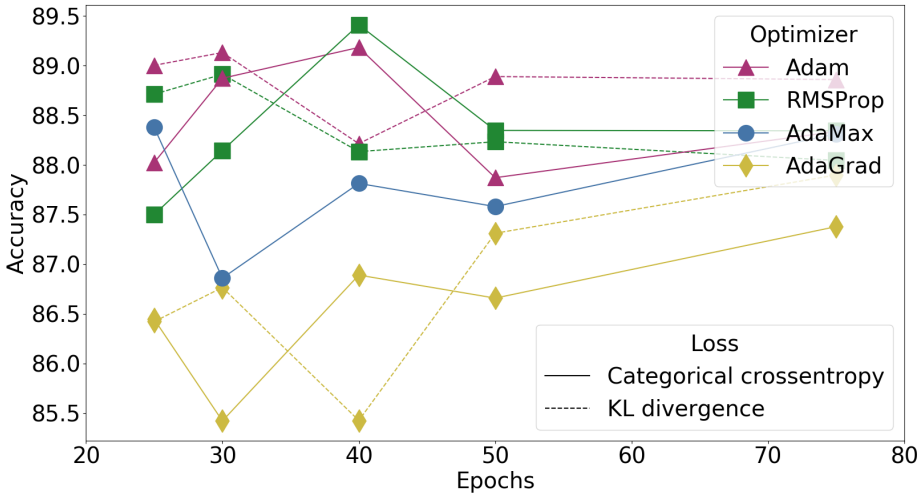

Figure 4.3: Accuracy scores for four different optimization functions using softmax as the activation function and two different loss functions. Other parameters were batches of 128 samples, learning rate of 0.001, and a vector length of 19,968.

Batch Size

Next, we investigated how batch size impacted the accuracy by testing sizes of 128 (the default), 64, 32, 16, and 8. We found that smaller batch sizes boosted the accuracy, shown in Figure 4.4. Based on this figure, we proceeded using the smallest batch size we could train in a reasonable amount of time, 16.

Learning Rate

Once the batch size was chosen, we could tune the learning rate to the most approprate value. These two parameters are closely related so we checked what learning rate worked best for batch sizes of 16 samples. Figure 4.5 shows how learning rates of 0.01, 0.001 (the default, used up until this point), and 0.0001 performed over five epoch values. From this, we concluded that the default value of 0.001 was slightly better-suited for a batch size of 16 than the other two.

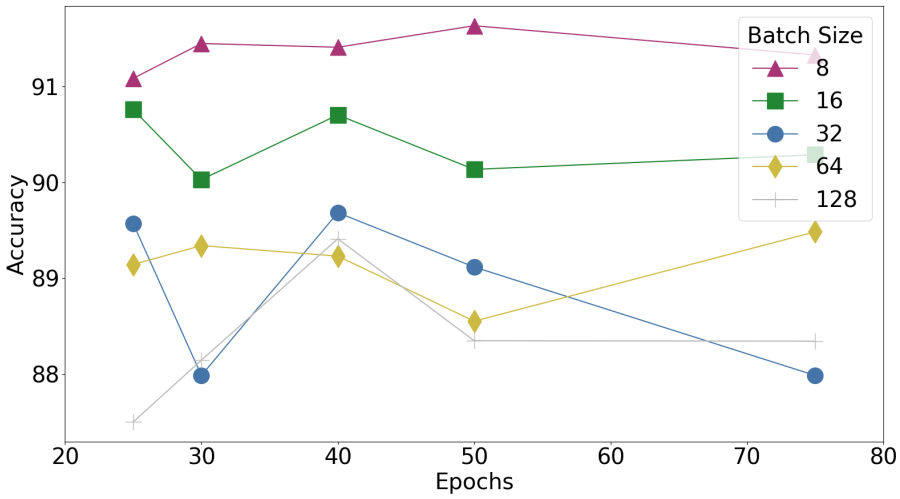

Figure 4.4: Accuracy scores of five different batch sizes using softmax, categorical crossentropy as the loss function and rmsprop as the optimizer. Untuned learning rate was 0.001 and vector length was 19,968.

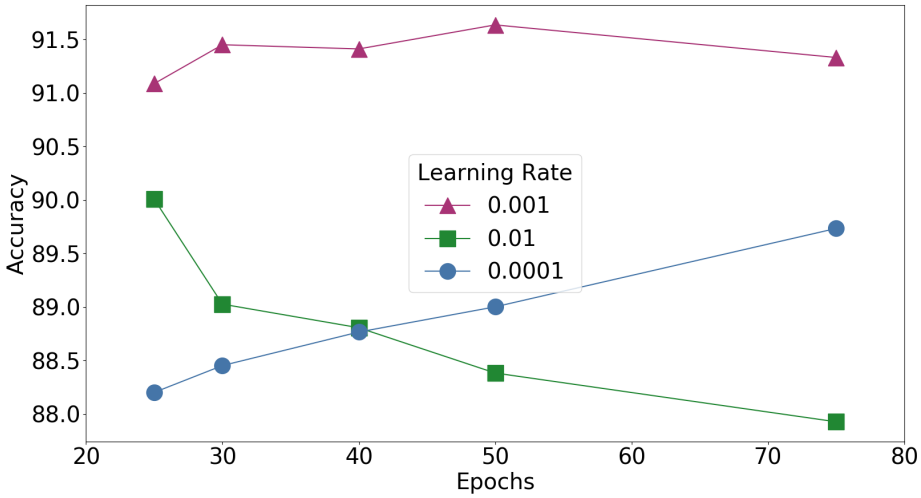

Figure 4.5: Accuracy scores of three different learning rates using softmax, categorical crossentropy, RMSProp, and batches of 16 samples. Untuned vector length was 19,968.

Vector Length

Finally, we compared three different feature vector lengths. Short vectors used the first 2 words of a caption, the first word of each header, the first 2 words of the section title and the first 20 of the section paragraph. This created vectors with 19,968 dimensions for each article when using BERTs 768 dimension vectors for each term. We used these short vectors as the defaults for the earlier experiments. Medium vectors used 3 words of the caption, the 2 first words of each header, first 3 words of the section title and the first 24 words of the section paragraph. This created vectors with a length of 26,112. Finally, long vectors took the first 5 words of the caption, the first 4 words of each header, first 5 of the section title and the first 35 words of the section paragraph. Long vectors were 40,704 dimensions. We chose these values based on the median length of each field: 2 words in a caption, 1 in headers, 2 in section title, and 23 in the section paragraph.

We report how each of these three lengths performed in Figure 4.6. The results of this experiment show that using more words in the features does not improve the accuracy and likely decreases it slightly due to the addition of more parameters the network has to tune. Therefore, we conclude that the best vector has a length of 19,968 dimensions.

When testing the five epoch values used previously (25, 30, 40, 50, and 75), we found that 50 epochs performed best but were forced to use 40 in the interest of time. This concluded our parameter tuning. Moving forward, all reported results use softmax as the activation function, categorical crossentropy to compute loss, RMSProp as the optimizer, batch sizes of 16, a learning rate of 0.001 and vectors of 19,968 dimensions.

4.2 Baseline

Because no work has had a similar enough usage to ours, we compared with a baseline and loosely compare with previous work discussed in Section 2.3. Our baseline method took all entity pairs available given a table and columns (or column and article subject) and queried Freebase for the set of relations

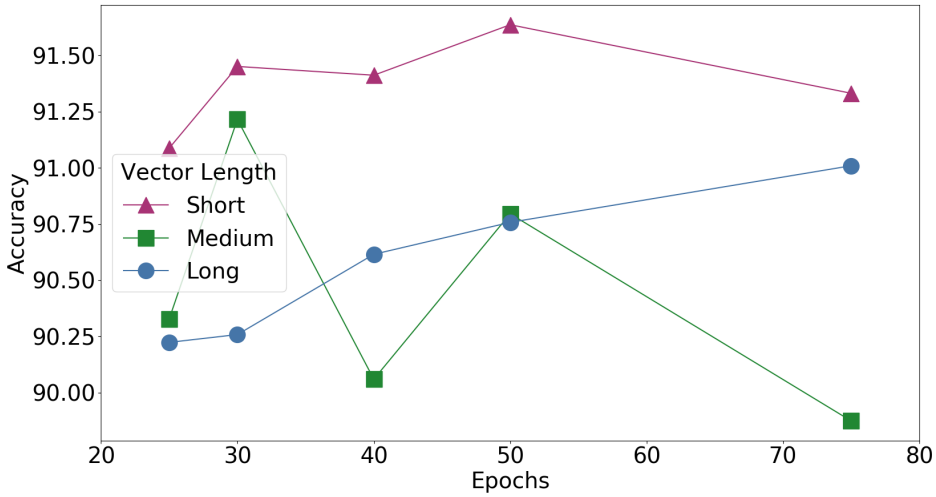

Figure 4.6: Accuracy scores of three different vector lengths using softmax, categorical crossentropy, RMSProp, batches of 16, and a learning rate of 0.001.

any of these pairs were related through. For example, to predict a relationship between the article subject and first column of the table in Section 1.2 the baseline queried for Freebase relations connecting the following list of entity pairs:

```
dbr:Louise_of_Hesse_Kassel - dbr:Frederick_VIII_of_Denmark
dbr:Louise_of_Hesse_Kassel - dbr:Princess_Alexandra_of_Denmark
dbr:Louise_of_Hesse_Kassel - dbr:George_I_of_the_Hellenes
dbr:Louise_of_Hesse_Kassel - dbr:Princess_Dagmar_of_Denmark
dbr:Louise_of_Hesse_Kassel - dbr:Princess_Thyra_of_Denmark
dbr:Louise_of_Hesse_Kassel - dbr:Prince_Valdemar_of_Denmark
```

To predict a relation we then took the most frequently returned relation from the queries, breaking ties alphabetically. For the example above no relations would be returned for any of the pairs meaning the baseline would predict no relation for that pair of columns of the table. We ran this baseline on the 20% of our tables set aside for testing and report results in Section 4.3.

4.3 Results

Once we chose the best parameter settings for the development set, we trained a network using the same parameters on the training set and tested on the testing set. We compared our results on the test set with the baseline described in Section 4.2. The accuracies of the two methods both broken down by relation and on average across the whole test set are shown in Table 4.1. This table shows that while our network does perform poorly for some relations, the overall accuracy for each relation is much better than the baseline. For the majority of relations, our network is able to predict the correct relation over 90% of the time and averages over 87%. In Section 4.3.2 we discuss the reasons our network fails for some relations.

We then performed a loose comparison with related methods discussed in Section 2.3.2. Table 4.2 has the results of this comparison and shows that, although the datasets are different for each of these systems, our accuracy is consistently higher than the two non-neural methods [5], [32]. In total, we were also able to extract an average of 582,691.4 new triples not already present in Freebase. From this, we can conclude that not only are neural networks applicable to this problem but LSTMs are well-suited despite being created to parse running text.

4.3.1 Ablation Studies

We ran a number of ablation studies to test how different parts of the method contributed to the accuracy. We started by testing how BERT embeddings performed compared to GloVe. Since its release in 2019, BERT has overtaken GloVe as the top embedding system [8], [35]. Building vectors for articles using GloVe was done the same as is described in Section 4.1.1 but instead of 768 dimensions, we used 300 dimension vectors trained on Common Crawl[1]. We created vectors the same number of tokens from each field of the article as the "small" vectors described in Section 4.1.1 producing 7,800 dimension vectors. The results of this test are shown in Figure 4.7 and show that using

[1]https://github.com/stanfordnlp/GloVe

Relation	Baseline	Network
actor-character	0.4%	99.5%
actor-film	1.7%	83.8%
author-works_ written	24.6%	87.5%
award-nominee	2.4%	100%
book-genre	42.9%	80%
company-industry	21.4%	96.4%
director-film	0%	78.8%
film-country	0%	83.5%
film-genre	19.0%	95.7%
film-language	52.9%	98.1%
film-music	49.6%	90.7%
film-production_ companies	48.1%	0%
football_ position-players	0%	96.6%
location-contains	0%	95.1%
musician-album	32.8%	95.7%
No relation	100%	63.1%
person-nationality	0%	95.9%
person-graduate	7.9%	98.5%
person-parents	0%	74.5%
person-place_ of_ birth	0%	99.4%
person-place_ of_ death	0%	8.6%
person-profession	17.7%	79.6%
person-religion	10.0%	58.0%
person-spouse	6.4%	53.8%
political_ party-politician	0%	96.5%
producer-film	0%	0%
sports_ team-player	12.4%	95.5%
writer-film	0%	38.6%
(Micro)Average	15.3%	87.7%

Table 4.1: Prediction accuracy on the test split achieved by the baseline and our method (trained on the training split).

	Accuracy	Precision	Recall	F1
Muñoz et al. [32]	78.13%	81.54	77.37	79.40
Cannaviccio et al. [5]	–	82	68	74
Present work	87.7%	98.71	91.59	95.02

Table 4.2: Comparison of our system on accuracy, precision, recall and F1 against two other methods.

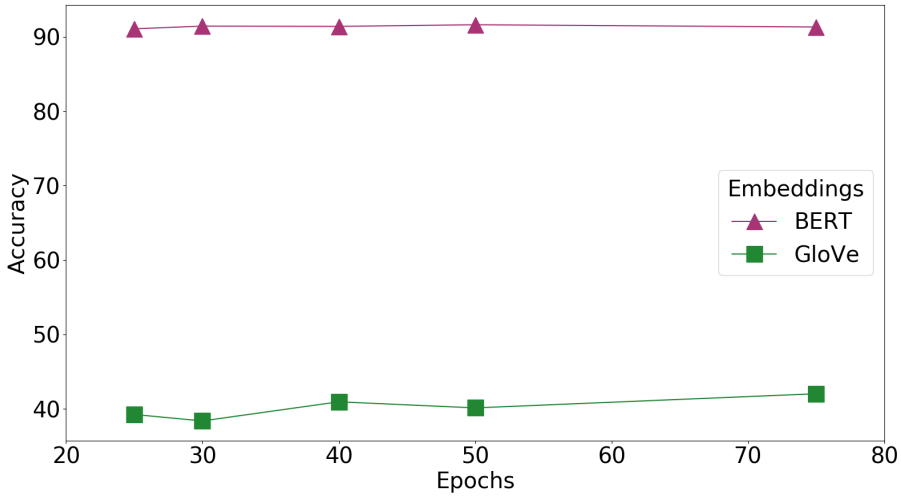

Figure 4.7: Accuracy values of our method using BERT embeddings and GloVe embeddings.

GloVe embeddings significantly impacts the accuracy. We conclude that BERT far outperforms GloVe likely due to the higher dimensionality, better vector encoding, use of contextual information, and larger vocabulary.

We also tested how each part of the Wikipedia article impacts the accuracy. To do this, we ran one set of tests for each field (caption, headers, section title and section paragraph) removing a different field each time. The results in Figure 4.8 show which fields provided the most information to the network. Removing captions reduced the accuracy slightly but not significantly. This was to be expected as most tables contained no caption. Removing the headers and section information impacted the accuracy much more. Therefore, these three features are likely weighted similarly in the network and provide the most information used for prediction, with the section title contributing slightly more than the other two.

4.3.2 Error Analysis

For the baseline, errors were almost always a prediction of no relation when a relation did exist between the pairs. This is caused by an absence of information in the knowledge graph and helps illustrate the importance of this

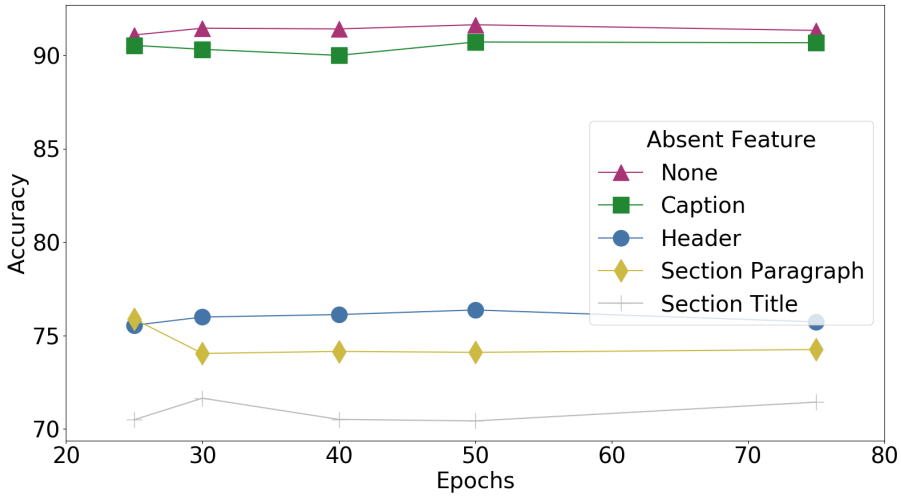

Figure 4.8: Accuracy values of our method removing one feature each time.

work. A small number of errors for twelve of the relations were predictions of a different relation which can be attributed to entities being related through more than one relation, explained in Section 3.2.1.

For our method, most errors were caused be a prediction of very similar relations like *film-writer* instead of *film-director*. This is likely because the majority of the information passed to the network points to the relation concerning a person and a film, but discerning a writer versus director is difficult and is done incorrectly in some cases. In addition, having fewer training examples gives the network less of a chance to tune the weights for that relation. As a result, relations with few examples like *person-place_of_death* had a low accuracy whereas relations like *sports_team-player* had a higher accuracy.

Both of these errors combine to adversely impact the accuracy for the relations *film-production_companies* and *producer-film*. First, this is a case where the structure and content of the articles is very similar to others concerning films (of which there are many). For example, section titles for many of the articles for these relations are "Filmography" and headers are often "Film" or "Title". Second, there are only 654 and 1,526 total samples for these relations respectively. This is not only a small number of samples overall, but a very small number when considering how many samples there are for *actor-film*,

actor-character, and *director-film*.

Finally, the accuracy of the collection of tables has a large impact on the accuracy of the network prediction. By examining Table 3.9 along with Table 4.1, we see that relations which had a low accuracy for collecting tables also had a lower accuracy in the network. Good examples of this are *writer-film* and *film-music*. These have a similar number of tables collected and tables with a very similar format; however, *film-music* had a much higher collection accuracy (86%) than *writer-film* (40%). This is carried forward to the network and explains the low accuracy for relations like *writer-film*, *producer-film*, *person-place_ of_ death*, and *person-spouse*.

Chapter 5

Conclusion

5.1 Overview

In this thesis, we presented a method for annotating tables using a combination of information extraction, distant supervision, binary classification and neural networks. Beginning with a set of twenty eight relations, we created a corpus of 200,000 Wikipedia tables annotated with relations. We used two methods to annotate this data in order to balance precision and recall.

The first, modeled after distant supervision, used a set of entities known to be related in a knowledge graph to find tables mentioning any of these pairs in either a pair of columns or the article subject and a column. Because of the amount of noise introduced by this method, we tested four different classification methods to clean up this dataset after hand annotating 200 tables for each relation. We showed that k-Nearest Neighbours performed best and improved the accuracy of this dataset significantly.

We also created custom queries for each relation to pull additional tables missed by the method above that had pairs absent in the knowledge graph. This step achieved much higher precision (above 95% for most relations) but returned far fewer tables.

This produced our dataset, which is publicly available for future methods to take advantage of. We split this dataset into 40% development to tune the parameters of our neural network, 40% to train and 20% to test the final network. After tuning six parameters, the network was able to achieve over 90% accuracy on the development set and 87.7% accuracy when trained on

the training set and tested on the test set.

We showed in our ablation studies that BERT far exceeds GloVe as an embedding tool for our work. In addition, we investigated how the different features of the Wikipedia article contributed to the accuracy of the network. In this set of tests, we showed that while the captions of the tables are not often informative, the other three features (section title and paragraph and headers) are weighted approximately equally.

Finally, we tested how many new triples could be added using our method from just the tables in the test set. On average for the test set, our method was able to extract 582,691 new triples.

In conclusion, we were able to prove in this thesis that a neural method for relation extraction on tables on Wikipedia is both appropriate and accurate. The accuracy of our network is competitive with state-of-the-art relation extraction methods for text without the semantic and syntactic information. Our work also achieved higher accuracy and F1 than other work for relation extraction on tables. Finally, our method for gathering tables to produce a dataset is the first of its kind and the first corpus of annotated tables to be released publicly.

5.2 Limitations

The main limitation to our work is the availability of training data. As we explained in subsection 4.3.2, when a relation has a small number of samples to train the network with, the prediction accuracy suffers. With this in mind, because of the nature of neural networks, there are not many other limitations for our method. As long as an article has headers, a section title and some text in the section, our system will be able to produce an accurate prediction. Our system is also not specific to the set of relations used in this thesis. As long as an appropriate number of training tables can be gathered, the network can be trained to predict any other relation.

5.3 Future Work

There are a number of directions for future work for this thesis. First of all, this method is applicable to other datasets of tables. For example, works like WebTables scan a corpus of tables from the web, not just Wikipedia [4]. The method described in this thesis can easily be applied to similar annotated datasets.

Alternative unsupervised approaches to annotating articles like feature learning with the help of libraries such as snorkel[1] could also be applied to this problem [37]. Methods like this annotate a corpus using predefined functions and learn the best weights for each function to create the training and testing data. It would be interesting to study how using feature learning to gather tables would perform in comparison to our method.

In addition, our method for selecting negative examples is imperfect and can be improved upon. The number of negative samples (12,491) is low compared to some other relations which likely impacts the final accuracy of the method. The makeup of these tables is also likely too widely varied for the network to learn appropriate weights. Finding a more sophisticated way to collect more negative samples would likely improve the accuracy of the system.

Fourth, we created models and experimented on 28 relations which is an improvement on some earlier works which could predict between 10 and 15 relations. An interesting direction for future work would be a test of how many relations are predictable in tables.

Another direction this work could be steered is in predicting properties rather than relations. In DBpedia, properties describe an entity using literals. For example, a persons birthdate or height or the GDP or population of a country are all properties. As long as an appropriate dataset can be obtained using a method like the one described in section 3.2, a similar network can be trained to predict properties in tables as well.

We also did not experiment with the architecture of the neural network and made the assumption that an LSTM would be most appropriate based on

[1] https://www.snorkel.org/

their success in other natural language processing tasks. With this in mind, another path for future work would be a thorough experimentation on the structure of the network which has the potential to improve accuracy even further.

Finally, more features have the ability to provide more information that could be useful for the network. For example, we did not use the text present in the cells, any type information about the entities in the columns or the article title as features in the network. We also did not analyse the abstract of the article, features of the entities in the table or other available text in the article.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. 15, 19

[2] F. BG and C. WA, "Simulation of self-organizing systems by digital computer," *Information Theory, IRE Professional Group on*, vol. 4, pp. 76–84, Oct. 1954. DOI: 10.1109/TIT.1954.1057468. 10

[3] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, ACM, 2013, pp. 18–26. DOI: 10.1145/2501511.2501516. 20

[4] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, Aug. 2008. DOI: 10.14778/1453856.1453916. 5, 20, 54

[5] M. Cannaviccio, L. Ariemma, D. Barbosa, and P. Merialdo, "Leveraging wikipedia table schemas for knowledge graph augmentation," in *Proceedings of the 21st International Workshop on the Web and Databases*, ACM, 2018, 5:1–5:6. DOI: 10.1145/3201463.3201468. 20–22, 47, 48

[6] M. Cannaviccio, D. Barbosa, and P. Merialdo, "Accurate fact harvesting from natural language text in wikipedia with lector," Jun. 2016, pp. 1–6. DOI: 10.1145/2932194.2932203. 19

[7] ——, "Towards annotating relational data on the web with language models," in *Proceedings of the 2018 World Wide Web Conference*, International World Wide Web Conferences Steering Committee, 2018, pp. 1307–1316. DOI: 10.1145/3178876.3186029. 20–22, 25, 64

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019. 15, 47

[9] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program - tasks, data, and evaluation.," in *LREC*, European Language Resources Association, 2004. 16, 17

[10] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, Sep. 1999, pp. 850–855. DOI: 10.1049/cp:19991218. 14

[11] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *Processing*, vol. 150, Jan. 2009. 18

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org. 10–12, 14

[13] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 1462–1471. 10

[14] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, pp. 8–12, 2009. 17, 18

[15] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, Association for Computational Linguistics, 2010, pp. 33–38. 16, 17

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Mar. 2015. 10

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735. 10, 14

[18] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Mar. 1991. DOI: 10.1016/0893-6080(91)90009-T. 12

[19] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., 2009, ISBN: 0131873210. 3, 4, 16, 17

[20] S. C. Kleene, "Representation of events in nerve nets and finite automata," in *Automata Studies*, Princeton University Press, 1956, pp. 3–41. 10

[21] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, JMLR.org, 2017, pp. 1885–1894. 15

[22] J. Lee, S. Seo, and Y. S. Choi, "Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing," *Symmetry*, vol. 11, 2019. 12, 38

[23] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer, "Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web Journal*, vol. 6, Jan. 2014. DOI: 10.3233/SW-140134. 26

[24] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 1338–1347, Sep. 2010. DOI: 10.14778/1920841.1921005. 5, 20

[25] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2016, pp. 2124–2133. DOI: 10.18653/v1/P16-1200. 18

[26] E. Macdonald and D. Barbosa, *Relational wikipedia tables*, 2019. DOI: 10.7939/DVN/SHL1SL. [Online]. Available: https://doi.org/10.7939/DVN/SHL1SL. 7, 34

[27] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. DOI: 10.1007/BF02478259. 10

[28] F. Mesquita, M. Cannaviccio, J. Schmidek, P. Mirza, and D. Barbosa, "Knowledgenet: A benchmark dataset for knowledge base population," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Nov. 2019, pp. 749–758. DOI: 10.18653/v1/D19-1069. 16

[29] T. Mikolov, K. Chen, G. S. Corrado, and J. A. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. 15

[30] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek, "Distant supervision for relation extraction with an incomplete knowledge base," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2013, pp. 777–782. 18

[31] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, 2009, pp. 1003–1011. 18, 25, 34, 64

[32] E. Muñoz, A. Hogan, and A. Mileo, "Using linked data to mine rdf from wikipedia's tables," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, ACM, 2014, pp. 533–542. DOI: 10.1145/2556195.2556266. 5, 20–22, 47, 48

[33] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor, "Industry-scale knowledge graphs: Lessons and challenges," *Queue*, vol. 17, no. 2, 20:48–20:75, Apr. 2019. DOI: 10.1145/3329781.3332266. 1

[34] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, pp. 489–508, Dec. 2016. DOI: 10.3233/SW-160218. 1

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. 15, 19, 47

[36] M. Purver and S. Battersby, "Experimenting with distant supervision for emotion classification," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2012, pp. 482–491. 18

[37] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proc. VLDB Endow.*, vol. 11, no. 3, pp. 269–282, 2017. 54

[38] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, Springer-Verlag, 2010, pp. 148–163. 16, 17

[39] D. Ritze, O. Lehmberg, and C. Bizer, "Matching html tables to dbpedia," in *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, ACM, 2015, 10:1–10:6. DOI: 10.1145/2797115.2797118. 20

[40] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *IRE Trans. Information Theory*, vol. 2, pp. 80–93, 1956. 10

[41] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0. 12

[42] M. Saeidi, R. Kulkarni, T. Togia, and M. Sama, "The effect of negative sampling strategy on capturing semantic similarity in document embeddings," in *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, Association for Computational Linguistics, Sep. 2017, pp. 1–8. 34

[43] Y. Sekhavat, F. Di Paolo, D. Barbosa, and P. Merialdo, "Knowledge base augmentation using tabular data," vol. 1184, Apr. 2014. 22

[44] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Hierarchical neural network generative models for movie dialogues," *CoRR*, 2015. 10

[45] T. P. Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher, "From freebase to wikidata: The great migration," in *World Wide Web Conference*, 2016. 2

[46] A. Uyar and F. M. Aliyu, "Evaluating search features of google knowledge graph and bing satori: Entity types, list searches and query interfaces," *Online Information Review*, vol. 39, no. 2, pp. 197–213, 2015. 1

[47] S. Vashishth, R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar, "RESIDE: Improving distantly-supervised neural relation extraction using side information," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018, pp. 1257–1266. DOI: 10.18653/v1/D18-1157. 18

[48] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu, "Recovering semantics of tables on the web," *Proc. VLDB Endow.*, vol. 4, no. 9, pp. 528–538, Jun. 2011. DOI: 10.14778/2002938.2002939. 5, 20

[49] L. Wang, Z. Cao, G. de Melo, and Z. Liu, "Relation classification via multi-level attention cnns," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Aug. 2016, pp. 1298–1307. DOI: 10.18653/v1/P16-1123. 12

[50] S. Wu and Y. He, "Enriching pre-trained language model with entity information for relation classification," *CoRR*, 2019. 12, 38

[51] P. Xu and D. Barbosa, "Connecting language and knowledge with heterogeneous representations for neural relation extraction," *NAACL-HLT*, vol. 1, pp. 3201–3206, 2019. 12, 38

[52] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *Document Analysis and Recognition*, vol. 7, no. 1, pp. 1–16, Mar. 2004. DOI: 10.1007/s10032-004-0120-9. 20

[53] D. Zelenko, C. Aone, and A. Richardella, "Kernel methods for relation extraction," *J. Mach. Learn. Res.*, vol. 3, pp. 1083–1106, Mar. 2003. 17

[54] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Sep. 2015, pp. 1753–1762. DOI: 10.18653/v1/D15-1203. 18

[55] Q. Zhang, M. Chen, and L. Liu, "A review on entity relation extraction," *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pp. 178–183, 2017. 17

[56] S. Zhang and K. Balog, "Ad hoc table retrieval using semantic similarity," in *Proceedings of the 2018 World Wide Web Conference*, International World Wide Web Conferences Steering Committee, 2018, pp. 1553–1562. DOI: 10.1145/3178876.3186067. 21

[57] ——, "On-the-fly table generation," in *The 41st International ACM SIGIR Conference on Research; Development in Information Retrieval*, ACM, 2018, pp. 595–604. DOI: 10.1145/3209978.3209988. 21

[58] ——, "Auto-completion for data cells in relational tables," *CIKM*, 2019. DOI: 10.1145/3357384.3357932. 21

[59] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2017, pp. 35–45. DOI: 10.18653/v1/D17-1004. 16, 17

Appendix A

Background Material

A.1 Relation Names

We used simplified names for Freebase relations in the tables throughout this thesis to keep tables and explanations more concise. In Table A.1 we provide the exact relations matched to the shortened terms used above. We also show in this figure the equivalent DBpedia relations used to gather entity pairs in Section 3.2.1.

In Table A.2 we list the related work we got each relation from.

A.2 Annotations

In this section we discuss some of the assumptions made when annotating tables to check the accuracy of the collection steps explained in Section 3.2. For a pair of columns or an article subject and column to be annotated as correct for some relation, the annotator had to be able to deduce the relationship based solely on the headers and context of the table. For example, consider a table listing film titles and people involved in the film with the header "Director(s)" and every person also wrote the film in their row. If this table was assigned the relation *film-writer*, we label it as incorrect. We annotate this way because our method relies on the information provided by the table and does not analyse existing relations between the entities.

For some relations, we make simplifications about the timing. For *sports_team-player*, we do not take into account former teams. This means we annotate

Freebase Relation	DBpedia Relation	Shortened Name
fb:award.award_nominated_work.award_nominations.. award.award_nomination.award_nominee		*award-nominee*
fb:book.author.works_written	is dbo:author of	*author-works_written*
fb:book.book.genre	dbo:literaryGenre	*book-genre*
fb:business.company.industry	dbo:industry	*company-industry*
fb:education.educational_institution.students_graduates.. education.educational_institution.students_graduates	is dbo:almaMater of	*person-graduate*
fb:film.actor.film..film.performance.character	dbo:portrayer	*actor-character*
fb:film.film.country	dbo:country	*film-country*
fb:film.director.film	is dbo:director of	*director-film*
fb:film.film.genre		*film-genre*
fb:film.film.language	dbo:language	*film-language*
fb:film.film.music		*film-music*
fb:film.film.production_companies	dbo:company	*film-production_company*
fb:film.performance.actor..film.performance.film	dbo:starring	*actor-film*
fb:film.producer.film	is dbo:producer of	*producer-film*
fb:film.writer.film	is dbo:writer of	*writer-film*
fb:government.political_party.politicians_in_this_party	is dbo:party of	*political_party-politician*
fb:location.location.contains	is dbo:isPartOf of	*location-contains*
fb:music.artist.album	is dbo:artist of	*musician-album*
fb:music.artist.origin	dbo:hometown	*musician-origin*
fb:people.deceased_person.place_of_death	deathPlace	*person-place_of_death*
fb:people.person.nationality	dbo:nationality	*person-nationality*
fb:people.person.parents	dbo:parent	*person-parents*
fb:people.person.place_of_birth	dbo:birthPlace	*person-place_of_birth*
fb:people.person.profession		*person-profession*
fb:people.person.religion	dbo:religion	*person-religion*
fb:people.person.spouse_s..people.marriage.spouse	dbo:spouse	*person-spouse*
fb:soccer.football_position.players	dbo:position	*football_position-player*
fb:sports.sports_team.roster.. sports.sports_team_roster.player	is dbo:team of	*sports_team-player*

Table A.1: Freebase relations used in our dataset and the shortened forms of their names we used throughout the thesis.

Relation	Source Literature
person-nationality	Mintz et al. [31]
location-contains	
person-profession	
person-place_of_birth	
film-genre	
film-language	
film-country	
writer-film	
director-film	
producer-film	
person-place_of_death	
musician-origin	
person-religion	
author-works_written	
football_position-player	
book-genre	
film-music	
company-industry	
person-parents	Cannaviccio et al. [7]
person-graduate	
sports_team-player	
political_party-politician	
award-nominee	
person-spouse	
actor-film	None
musician-album	
film-production_company	
actor-character	

Table A.2: Source literature of each relation.

an out-of-date table listing the roster of a sports team is correct even though many of the players may not play for that team anymore. Similarly, with *person-graduate*, we include tables listing people who we know attended the school (based on the information in the table), but may not have graduated.

Some relations are ambiguous and we err on the side of generality in these cases. *musician-album* is an example of this in which artists could contribute to a song in an album which we annotate as correct. Likewise, *film-writer* could include the screenwriter or story creators.

A.3 Queries

We ran 193 queries to collect tables for each relation based on the headers, section title and article type. Below we list all the queries broken down by relation:

actor-character

- (Headers = "Actor" & "Character") & (Section title = "Filmography" | "Films")
- (Headers = "Actor" & "Role") & (Section title = "Filmography" | "Films")
- (Headers = "Actress" & "Character") & (Section title = "Filmography" | "Films")
- (Headers = "Actress" & "Role") & (Section title = "Filmography" | "Films")
- (Headers = "Title" & "Role") & (Section title = "Filmography" | "Films" | "Film" | "Selected filmography")
- (Headers = "Film" & "Role") & (Section title = "Filmography" | "Films" | "Film" | "Selected filmography")

actor-film

- Headers = ("Cast" & "Film")
- Headers = ("Actor" & "Film")
- Headers = ("Actress" & "Film")
- Headers = ("Cast" & "Movie")
- Headers = ("Actor" & "Movie")
- Headers = ("Actress" & "Movie")
- Headers = ("Cast" & "Title") & (Section title = "Filmography" | "Films")
- Headers = ("Actor" & "Title") & (Section title = "Filmography" | "Films")
- Headers = ("Actress" & "Title") & (Section title = "Filmography" | "Films")
- Headers = ("Cast" & "Title") & (Section title = "2010s" | "2000s")
- Headers = ("Actor" & "Title") & (Section title = "2010s" | "2000s")

- Headers = (“Actress” & “Title”) & (Section title = “2010s” | “2000s”)

author-works_written

- Headers = (“Author” & “Title”)
- Headers = (“Author” & “Book”)
- Headers = (“Writer” & “Book”)
- Headers = (“Title”) & (Section title = “Bibliography” | “Novels” | “Fiction” | “Non-fiction” | “Books”) & (Article type = “person” | “writer”)

award-nominee

- (Headers = “Category” & “Recipient”)
- (Headers = “Award” & “Recipient”)
- (Headers = “Award” & “Winner”)

book-genre

- (Headers = “Title” & “Genre”) & (Section title = “Bibliography”)
- (Headers = “Title” & “Genre(s)”) & (Section title = “Bibliography”)
- (Headers = “Work” & “Genre”) & (Section title = “Bibliography”)
- (Headers = “Work” & “Genre(s)”) & (Section title = “Bibliography”)

company-industry

- (Headers = “Company” & “Industry”)
- (Headers = “Company” & “Sector”)
- (Headers = “Employer” & “Industry”)
- (Headers = “Name” & “Industry”) & (Section title = “Companies”)

director-film

- (Headers = “Director” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Director” & “Film”)
- (Headers = “Director” & “English Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Director(s)” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Director(s)” & “Film”)
- (Headers = “Director(s)” & “English Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Film”) & (Section title = “As director”) & (Article type = “person”)
- (Headers = “Title”) & (Section title = “As director”) & (Article type = “person”)
- (Headers = “Movie”) & (Section title = “As director”) & (Article type = “person”)

- (Headers = “Film”) & (Section title = “Director”) & (Article type = “person”)
- (Headers = “Title”) & (Section title = “Director”) & (Article type = “person”)
- (Headers = “Movie”) & (Section title = “Director”) & (Article type = “person”)
- (Headers = “English title”) & (Section title = “Director”) & (Article type = “person”)

film-country

- (Headers = “Title” & “Country”) & (Section title = “Filmography” | “Films”)
- (Headers = “Film” & “Country”)
- (Headers = “English title” & “Country”) & (Section title = “Filmography” | “Films”)
- (Headers = “Title” & “Production Country”) & (Section title = “Filmography” | “Films”)
- (Headers = “Film” & “Production Country”)
- (Headers = “English title” & “Production Country”)

film-genre

- (Headers = “Film” & “Genre”)
- (Headers = “Title” & “Genre”) & (Section title = “Filmography” | “Films”)
- (Headers = “Movie” & “Genre”)
- (Headers = “English title” & “Genre”) & (Section title = “Filmography” | “Films”)

film-language

- (Headers = “Film” & “Language”)
- (Headers = “Title” & “Language”) & (Section title = “Filmography” | “Films”)
- (Headers = “Movie” & “Language”)
- (Headers = “Name of Film” & “Language”)

film-music

- (Headers = “Film” & “Music Director”)
- (Headers = “Film” & “Composer”)

film-production_ companies

- (Headers = “Title” & “Studio”) & (Section title = “Filmography” | “Films”)
- (Headers = “Title” & “Studio(s)”) & (Section title = “Filmography” | “Films”)
- (Headers = “Film” & “Studio”)
- (Headers = “Film” & “Studio(s)”)
- (Headers = “Movie” & “Studio”)
- (Headers = “Movie” & “Studio(s)”)

football_position-player

- (Headers = “Position” & “Name”) & (Section title = “Name”) & (Article type = “football club season” | “Football club season” | “football club”)
- (Headers = “Position(s)” & “Name”) & (Section title = “Name”) & (Article type = “football club season” | “Football club season” | “football club”)
- (Headers = “Pos.” & “Name”) & (Section title = “Name”) & (Article type = “football club season” | “Football club season” | “football club”)
- (Headers = “Position” & “Name”) & (Section title = “Player”) & (Article type = “football club season” | “Football club season” | “football club”)
- (Headers = “Position(s)” & “Name”) & (Section title = “Player”) & (Article type = “football club season” | “Football club season” | “football club”)
- (Headers = “Pos.” & “Name”) & (Section title = “Player”) & (Article type = “football club season” | “Football club season” | “football club”)

location-contains

- (Headers = “Country” & “City”)
- (Headers = “State” & “City”)
- (Headers = “Province” & “City”)
- (Headers = “Country” & “City”)
- (Headers = “Country” & “County”)
- (Headers = “Country” & “State”)
- (Headers = “Country” & “Province”)

musician-album

- (Headers = “Artist” & “Album”)
- (Headers = “Other artist(s)” & “Album”)
- (Headers = “Artist(s)” & “Album”)
- (Headers = “Album”) & (Section title = “Albums”) & (Article type = “musical artist”)
- (Headers = “Title”) & (Section title = “Albums”) & (Article type = “musical artist”)
- (Headers = “Album”) & (Section title = “Discography” | “Studio albums”) & (Article type = “musical artist”)
- (Headers = “Title”) & (Section title = “Discography” | “Studio albums”) & (Article type = “musical artist”)
- (Headers = “Album”) & (Section title =) & (Article type = “musical artist”)

musician-origin

- (Headers = “Artist” & “Hometown”)
- (Headers = “Artist” & “Origin”)

person-graduate

- (Headers = “COLLEGE” & “ROSTER”)
- (Headers = “University” & “Name”)
- (Headers = “College” & “Name”)
- (Headers = “College” & “Player”)

person-nationality

- (Headers = “Name” & “Nationality”)
- (Headers = “Name” & “Nation”)
- (Headers = “Name” & “Nat”)
- (Headers = “Name” & “Nat.”)

person-parents

- (Headers = “Name” & “Father”)
- (Headers = “Name” & “Parents”)
- (Headers = “Children” & “Name”)

person-place_ of_ birth

- (Headers = “Name” & “Place of Birth”)
- (Headers = “Name” & “Birthplace”)

person-place_ of_ death

- (Headers = “Name” & “Place of Death”)
- (Headers = “Person” & “Place of Death”)

person-profession

- (Headers = “Name” & “Occupation”)
- (Headers = “Celebrity” & “Occupation”)

person-religion

- (Headers = “Name” & “Religion”)
- (Headers = “Chaplain” & “Denomination”)
- (Headers = “Patriarch” & “Church”)
- (Headers = “Senator” & “Religion”)

person-spouse

- (Headers = “Name” & “Husband”)
- (Headers = “Name” & “Wife”)
- (Headers = “Name” & “Spouse”)

political_ party-politician

- (Headers = “Party” & “Representative”)
- (Headers = “Party” & “Name”)
- (Headers = “Party” & “Candidate”)
- (Headers = “Party” & “Member”)

producer-film

- (Headers = “Producer” & “Film”)
- (Headers = “Producer” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Producer” & “English title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Producer” & “Movie”)
- (Headers = “Producer(s)” & “Film”)
- (Headers = “Producer(s)” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Producer(s)” & “English title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Producer(s)” & “Movie”)

sports_ team-player

- (Headers = “Club” & “Player”)
- (Headers = “Team” & “Player”)
- (Headers = “Club” & “Driver”)

writer-film

- (Headers = “Writer” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Writer” & “Film”)
- (Headers = “Writer” & “Movie”)
- (Headers = “Writer(s)” & “Title”) & (Section title = “Filmography” | “Films”)
- (Headers = “Writer(s)” & “Film”)
- (Headers = “Writer(s)” & “Movie”)

A.4 Confusion Matrix

Below we present the average confusion matrix obtained by our final network on the test data. This is an average of 5 runs. Network predictions are represented by the top axis while target relations are along the leftmost axis. For example, the network incorrectly predicted *actor-character* an average of 10.4 times when the proper relations was *actor-film*. Diagonals (correct predictions) are bolded.

	None	*actor-character*	*actor-film*	*author-works_ written*	*award-nominee*	*book-genre*
None	**1,975.8**	9.4	10.4	4.6	1.2	0.2
actor-character	4.4	**4,408.2**	1.8	0.2	0	0
actor-film	48.4	11.6	**6,420.8**	19.4	0.4	0
author-works_ written	9.2	0.4	7.0	**356.0**	0	0
award-nominee	0	0	0	0	**1,145.8**	0
book-genre	0.2	0	0	0	0	**5.6**
company-industry	0.8	0	0	0	0	0
director-film	18.8	0	226.4	9.6	0	0
film-country	4.2	0.2	0.4	0	1.0	0
film-genre	1.4	0	0.2	0	0	0
film-language	4.0	0	1.6	0	0	0
film-music	6.4	0.2	0.6	0.2	0.8	0
film-production_ companies	43.8	0	1.0	0	0	0
football_ position-player	10.4	0	0	0.2	0	0
location-contains	89.4	0.2	0.4	1.0	0	0
musician-album	37.2	0	12.2	4.6	0.8	0
musician-origin	4.0	0	0	0	0	0
person-graduate	3.6	0	0	0	0	0
person-nationality	48.2	0.6	0.6	1.2	0.2	0
person-parents	3.6	0.4	0.8	0.4	0	0
person-place_ of_ birth	0.2	0	0	0	0	0
person-place_ of_ death	2.6	0.2	1.0	0.2	0	0
person-profession	9.4	19.0	0.8	0.2	0	0
person-religion	0.6	0.2	0	0.2	0	0
person-spouse	12.6	0	0.2	0.2	0	0
political_ party-politician	119.6	0.6	0.4	0.2	0.6	0
producer-film	2.4	0	43.8	0.6	0	0
sports_ team-player	139.8	0.4	0.4	0	0.2	0
writer-film	0	0.2	77.4	2.2	0	0

	company-industry	*director-film*	*film-country*	*film-genre*	*film-language*	*film-music*
None	2.4	27.6	3.2	0.4	3.6	13.6
actor-character	0	2.2	0	0.4	0	0
actor-film	0	578.8	2.8	1.2	0.8	2.8
author-works_written	0	6.6	1.2	0	0.2	0
award-nominee	0	0	0	0	0	0
book-genre	0	0.2	0	1.0	0	0
company-industry	**27.0**	0	0	0.2	0	0
director-film	0	**1,429.8**	2.4	0	1.0	4.0
film-country	0	2.0	**127.0**	0.4	0	5.2
film-genre	0	0	0	**40.2**	0	0
film-language	0	0.4	3.6	0	**691.6**	3.6
film-music	0	2.4	2.6	0.4	2.0	**206.8**
film-production_companies	0	8.2	15.2	2.8	6.6	34.0
football_position-player	0	0	0	0	0	0
location-contains	0	2.2	1.6	0	0	0.2
musician-album	0	31.6	0.2	0	0.2	1.6
musician-origin	0	0	0.8	0	0	0
person-graduate	0	0	0	0	0	0
person-nationality	0	2.0	5.0	0	0.2	0.4
person-parents	0	1.0	0	0	0	0
person-place_of_birth	0	0	0	0	0	0
person-place_of_death	0	1.0	0	0	0	0
person-profession	0	3.2	0	0	0.2	0
person-religion	0	0	0	0	0	0
person-spouse	0	0.6	0	0	0	2.0
political_party-politician	0.2	1.4	0	0	0	0.4
producer-film	0	227.8	0	0.4	0	0
sports_team-player	0	1.0	0	0	0	0.2
writer-film	0	65.8	0	0	0	0

	film-prod_ companies	*football_ pos-player*	*location-contains*	*musician-album*	*musician-origin*	*person-graduate*
None	0	10.2	77.4	42.8	0	4.4
actor-character	0	0	0.2	1.2	0	0
actor-film	0	1.2	2.6	46.6	0	0.4
author-works_ written	0	0.6	1.4	7.0	0	0.2
award-nominee	0	0	0	0	0	0
book-genre	0	0	0	0	0	0
company-industry	0	0	0	0	0	0
director-film	0	0.2	1.4	9.0	0	0.2
film-country	0	0	0.8	0	0	0
film-genre	0	0	0	0	0	0
film-language	0	0	0	0	0	0
film-music	0	0	0.6	0.4	0	0
film-production_ companies	**0**	2.2	5.8	0.4	0	0.4
football_ position-player	0	**2,775.2**	0.6	0	0	1.2
location-contains	0	1.0	**2,984.4**	0.8	0	2.6
musician-album	0	0.6	1.4	**2,690.4**	0	0.2
musician-origin	0	0	1.4	104.0	**0**	0
person-graduate	0	0	1.0	1.0	0	**1,081.2**
person-nationality	0	3.2	7.4	1.2	0	0.8
person-parents	0	0.2	0	0.2	0	0
person-place_ of_ birth	0	0	0	0	0	0
person-place_ of_ death	0	0	5.2	0	0	0
person-profession	0	0	0.8	3.8	0	0.4
person-religion	0	0	2.2	0	0	0
person-spouse	0	0.2	4.4	4.2	0	0
political_ party-politician	0	1.6	19.0	1.4	0	0.2
producer-film	0	0	0	4.6	0	0.2
sports_ team-player	0	29.0	21.0	1.0	0	109.6
writer-film	0	0	0.8	4.0	0	0

	person-nationality	*person-parents*	*person-place_of_birth*	*person-place_of_death*	*person-profession*	*person-religion*
None	55.8	4.2	0.2	1.4	9.2	0
actor-character	0	0	0	0	11.2	0
actor-film	1.8	0	0	0	15.2	0
author-works_written	2.2	0	0	0	0.4	0
award-nominee	0.2	0	0	0	0	0
book-genre	0	0	0	0	0	0
company-industry	0	0	0	0	0	0
director-film	2.8	0	0	0	0.6	0
film-country	7.4	0	0	1.0	0	0
film-genre	0	0	0	0	0	0
film-language	0	0	0	0	0	0
film-music	1.0	0	0	0	1.2	0
film-production_companies	0.8	1.0	0.2	0.2	0	0
football_position-player	46.4	0	0	0	0	0
location-contains	12.8	0.8	0.4	2.6	0.2	0
musician-album	1.2	0.4	0	0.2	0.6	0
musician-origin	11.2	0.2	0	0	0.2	0
person-graduate	0.2	0	0	0	0.4	0
person-nationality	**2,580.2**	2.6	2.6	1.4	1.0	0
person-parents	1.4	**148.6**	0	0	0	0
person-place_of_birth	0	0	**35.8**	0	0	0
person-place_of_death	11.2	0.4	1.0	**2.4**	0.2	0
person-profession	1.2	0	0.4	0.2	**180.0**	0
person-religion	0	0	0	0	0	**5.8**
person-spouse	1.2	0.8	0	1.6	0	0
political_party-politician	4.2	0.4	0	0	3.4	0
producer-film	0.4	0	0	0	1.0	0
sports_team-player	11.4	0.2	0	0.2	1.8	0
writer-film	0	0	0	0	1.2	0

	person-spouse	*political_party-politician*	*producer-film*	*sports_team-player*	*writer-film*
None	10.2	85.6	0	113.0	32.2
actor-character	0.6	0	0	0.4	1.2
actor-film	1.2	1.6	0	4.6	501.8
author-works_written	0.8	1.0	0	3.4	9.4
award-nominee	0	0	0	0	0
book-genre	0	0	0	0	0
company-industry	0	0	0	0	0
director-film	1.8	0.6	0	1.4	104.0
film-country	0	0.6	0	0	1.8
film-genre	0	0	0	0	0.2
film-language	0	0	0	0	0.2
film-music	0.6	0	0	0.2	1.6
film-production_companies	1.6	0.4	0	1.0	5.4
football_position-player	0	2.6	0	34.6	0.8
location-contains	1.2	11.8	0	17.0	6.4
musician-album	1.2	0	0	0.4	27.0
musician-origin	26.2	0.2	0	0	0.8
person-graduate	0	1.2	0	9.0	0.4
person-nationality	1.0	9.4	0	14.2	5.6
person-parents	2.2	1.6	0	0	0.6
person-place_of_birth	0	0	0	0	0
person-place_of_death	0.4	0.6	0	0.2	1.4
person-profession	0.4	1.4	0	0.8	3.8
person-religion	0.2	0.2	0	0.2	0.4
person-spouse	**42.0**	3.4	0	2.8	1.8
political_party-politician	3.8	**5,009.4**	0	14.6	11.6
producer-film	0	0.2	**0**	0	24.6
sports_team-player	2.6	20.2	0	**7,427.0**	14.0
writer-film	0	0	0	0	**95.4**