Relation Extraction With Synthetic Explanations And Neural Network

by

Rozan Chahardoli

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

Master of Science

Department of Computing Science

University of Alberta

UNIVERSITY OF ALBERTA

Abstract

Faculty of Graduate Studies and Research
Department of Computing Science

Master of Science

by Rozan Chahardoli

Relation Extraction, which is defined as the detection of existing relations between a pair of entities in a sentence, has received a large interest lately, including more recent work on using neural methods. Since neural systems need a large number of annotated sentences to build effective models, Distant Supervision has been a preferred choice for collecting training labeled data. However, recent published work has shown that, training classifiers via a small number of annotated data and some explanation of why a sentence expresses a relation performs as accurate as distant supervision methods working with a large number of annotated sentences. In this thesis, we show that we can generate synthetic explanations, based on a small number of trigger words, for each relation in a way that the resulting explanations achieve comparable accuracy to human produced explanations by training a neural classifier. Our system is evaluated on five relation extraction tasks with different entity types (person-person, person-location, etc.) and the results show that synthetic explanations can work as precise as human generated explanations for the task of relation extraction. The proposed system also has the ability to classify noisy data coming from distant supervision methods with a reasonable accuracy.

Preface

I, Rozan Chahardoli, declare that this thesis titled, 'Neural Relation Extraction With Synthetic Explanations' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"No two things have been combined better than knowledge and patience."

Prophet Muhammad (peace be upon him)

To:

The great savior of the world

Acknowledgements

I would like to express my special appreciation and thanks to my advisor Professor Davood Rafiei, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been invaluable.

I would also like to thank Professor Denilson Barbosa for helping me with providing data and giving me suggestions to solve the problems. He gave an intersting hint for generating synthetic explanations using trigger words. His suggestion about usage of BabbleLabel labeling functions was also very helpful in my research.

I would also like to thank my committee members, professor Osmar Zaiane and professor Marek Reformat for serving as my committee members even at hardship.

A special thanks to my family. Words can not express how grateful I am to my mother-in-law, my mother and father. Your prayer for me was what sustained me thus far.

I would also like to thank to my beloved husband, Mojtaba Yeganejou. Thank you for supporting me for everything, and especially I can't thank you enough for encouraging me throughout this experience.

Finally I thank my God, for letting me through all the difficulties. I have experienced Your guidance day by day. You are the one who let me finish my degree. I will keep on trusting You for my future. Thank you, Lord.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	What (it) Stands For
KB	**K**kowledge **B**ase
IBM	**I**nternational **B**usiness **M**achines
KG	**K**kowledge **G**raph
RE	**R**elation **E**xtraction
NER	**N**amed **E**ntity **R**ecognition
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
DS	**D**istant **S**upervision
CFG	**C**ontext **F**ree **G**rammar
CBOW	**C**ommon **B**ag **O**f **W**ords
FIGER	**FI**ne **G**rained **E**ntity **R**ecognizer
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificiall **N**eural **N**etwork
ReLU	**R**ectified **L**inear **U**nit
LF	**L**abeling **F**unction
Bbl	**B**abbleLabel
FstT	**F**ast**T**ext
FGR	**F**IGER
StNER	**S**tanford **N**amed **E**ntity **R**ecognition
BoBigr	**B**ag of **B**igrams
MLP	**M**ulti-**L**ayer **P**erceptrons
Explan.	**E**xplanation
autom.	automatic

Physical Constants

Constant Name	Symbol	=	Constant Value (with units)
Euler's Number	e	$\cong$	2.71827

Chapter 1

Introduction

1.1 Knowledge Bases

Knowledge Bases (KBs) structure and maintain large volumes of useful information about the world and are used in support of many important applications with impressive success, including question answering, web search, etc. (see Figure 1.1 as an example). KBs have also been used in enriching documents and structured data with semantic tags, by linking both the entities mentioned in a document and the relationships between those entities expressed in text to entities and relationships that also exist in a reference KB. A well known example is the usage of YAGO KB [7], which resulted in defeating human experts in a trivia game [8].

However, KBs need regular maintenance over time to remain relevant. Any new knowledge that is harvested from different sources may be integrated with a KB while old knowledge maybe revised to account for changes over time. The inherent difficulties in these tasks motivate the development of automatic tools for extracting facts that are recognized by or can be integrated into a KB. This is often done by leveraging one or more *phrases* connecting a pair of entities in a text corpus, which in turn can be added to the KB. Taking a look at the well-utilized knowledge graph (graph based KB) of Google for "Ben Ripley", shown in Figure 1.2, we cannot find a mention of "Stanford university" under the education relationship. However, in the Wikipedia text of Ben Ripley page, we can find the sentence *"Ripley is a graduate of* ***Stanford University*** *and the University of Southern California's USC School of Cinema-Television."* Thus, we need algorithms

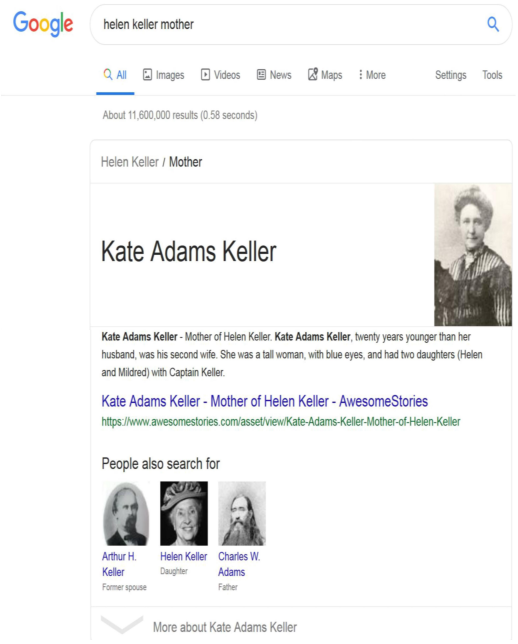


FIGURE 1.1: Question-Answering using Google KG.

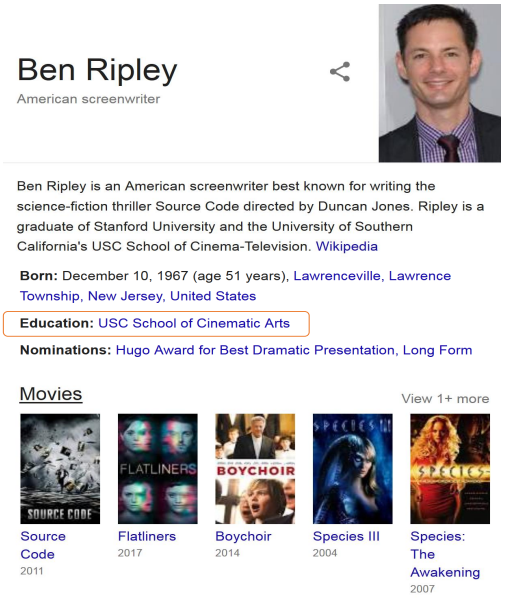


FIGURE 1.2: Google KG of Ben Ripley

and systems that can extract most (if not all) of the relations between entities in text to expand knowledge bases.

1.2 Relation Extraction

Relation Extraction (RE) aims to identify the relationship between two nominals, typically in the context of a sentence, making it essential to natural language understanding [9]. RE is a key component of many natural language processing applications, such as information extraction [10], knowledge base population [11], and question answering [12].

Early works on Relation Extraction (e.g., Bunescu and Mooney [13]) rely on supervised learning for two sub-tasks: detecting whether a relation is expressed in the text, and if so, classifying the relation into one of the known KB relations. The cost for human annotation has since been identified as prohibitively high, and many subsequent methods aim at reducing that burden. Lightly supervised methods (e.g., Agichtein and Gravano [14], Bollegala et al. [15]) require only seed tuples with entities known to be related in the KB and exploit regularities in language to train relation extractors.

Relation extraction from raw text is commonly divided into the three important sub-problems of Named Entity Recognition (NER), Entity Linking and Relation Prediction [16]. As a state-of-the-art work, Ren et al. [3] work on the problem as a whole and report performance results in all sub-problems that are comparable or better than others [17, 18]. In this thesis, we focus on the third sub-problem and develop a method for predicting the relations between pairs of entities. We compare our results with the results of relation prediction part of Ren et al. [3].

1.3 Distant Supervision

A major goal of Machine Learning algorithms may be identified as inductively creating models to predict, classify, and make decisions [19]. Achilles heel of this model development is having access to a set of training data. With the commence of deep learning and big GPU machines, the need for good or large datasets has shown its effect on the performance of ML Methods more than before. In particular, in some NLP tasks such as relation extraction, large training sets are shown to be very useful [20]. In order to make the required data, the first and most handy way is to employ some human experts to annotate the data. However, this process is expensive in terms of the time it takes and the resources that are needed [21]. Thus, the idea of making the required data using a ML algorithm has been an interesting direction. One idea is to have a purely unsupervised information extraction, which extracts the phrases between entities in large amounts of text, and clusters and/or simplifies these phrases to produce relations (see the methods developed by Shinyama and Sekine [22] and Banko et al. [23]). Although this approach makes a larger training set, compared to those annotated by experts, the resulting relations may not be easy to map to relations needed for a particular knowledge base [21]. Boot-strap learning is another approach that uses small number of patterns or seed instances to produces more instances and patterns. This is employed by Brin [24], Riloff et al. [25] and Rozenfeld and Feldman [26] in their methods.

Distant Supervision is an alternative approach for creating large training sets for relation extraction, first proposed by Mintz et al. [21]. This method requires as input a large number of pairs of entities known to be related. For each entity pair and their related relationship, the authors query a large text corpus to identify sentences that mention both entities in a pair of entities. The obtained sentences are tagged as the samples of the

relationship between the entity pair's relation. Distant supervision is far from perfect, as many pairs of entities belong to multiple relations, which in turn brings considerable noise to the training data [21].

1.4 Training classifiers with labels and explanations

A more recent line of work to address the high costs in obtaining human annotations, which we follow here, is to ask humans to provide both *labels* for the data as well as *explanations* in natural language that justify those labels [27]. Using explanations and labels together is expected to reduce the need for large annotated data. The following example from Hancock et al. [27] shows an annotated sentence and the corresponding explanation that justifies the positive label:

Example 1.1. "Brady's wife, Gisele Bundchen pictured above as she visits the sports therapy center at Gillette Stadium in Massachusetts in May."

- *relation:* spouse between 'Brady' and 'Gisele Bundchen'.

- *label:* True

- *explanation:* because the phrase "'s wife" are right before person 2.

An explanation in this context provides a reason for why the sentence expresses a relation. The structure of explanations can be different for each sentence, depending on the candidate sentence and the user writing the explanation.

A good explanation is meant to provide a way to obtain more training data for the relation, by identifying features that are determinant for the label. The BabbleLabel framework of Hancock et al. [27] uses a semantic parser for the explanations, which can mix natural language and some special purpose logical predicates. Many *labeling functions* are automatically derived from the explanations and are used as feature extractors for the actual sentences. For example, a boolean labeling function can check for the presence of the phrase "'s wife" immediately before the second *person* entity in a phrase.

Hancock et al. [27] show that good explanations, although require more effort from human annotators, generalize to many examples and lead to a drastic reduction in the

overall annotation effort. To quote them: they "find that users are able to train classifiers with comparable F1-score from 5–100 × faster by providing explanations instead of just labels." They develop a system for generating a large training set for relation extraction. The quality of their results depends on the quality of the small number of hand-labeled instances and explanations that are provided.

BabbleLabel has been used in this thesis for generating a set of features to classify the relations. We expand their idea by making the labeling process automatic. By this effort, we aim to skip the role of human annotators in distant supervision. On the other hand, the more explanations, the more accurate system. Since an automatic system is better equipped to find more explanations from training data, it can give a better performance in the labeling process. Automatically generating explanations makes the labeling efficient in terms of cost and performance.

To make the system more accurate, we make use of more natural language features like sentence embedding and types of entity mentions.

1.5 Contributions

Our goal is to further reduce the cost of obtaining training sentences for the relation extraction task. We achieve that by combining into a single method for relation extraction the strengths of both light/distant supervision approach and the use of natural language explanations for labeling data. More precisely, starting from a specific relation (e.g., spouse), a list of prominent *trigger* words for the relation (e.g., husband, wife, spouse, etc.), and a set of sentences mentioning entities of the appropriate type for the relation (e.g., sentences mentioning two people) we want to:

- derive synthetic explanations based on the presence or absence of a positive or negative trigger word (or a closely related synonym);
- process the explanations through the BabbleLabel framework to obtain labeling functions;
- train a neural classifier based on features extracted from the labeling functions and other linguistic features.

We hypothesize that such a pipeline can yield high precision extractions even with a small number of trigger words (and hence explanations). To be more clear, we expect that such a pipeline to be comparable to or outperform the state of the art in terms of precision with less than 15 trigger words, and that the trigger words can be automatically selected from training data (if not provided).

In this thesis we describe and evaluate a neural model for relation extraction that implements the pipeline just described. We compare our results against that of Hancock et al. [27], using all of the evaluation data they provide [5] (which is a subset of the data they use in the experiments they report). We use their provided codes and data in our experiments.

Our model differs from theirs in some important ways. First, we use a standard neural network for relation prediction, while they use a simpler model. Second, we use the raw output of their labeling functions as features, following an approach the authors discuss in their paper, although they do not evaluate. This comparison, albeit limited by the availability of data, shows that our approach is superior, achieving F1-score upwards of 70%, compared to 50% as reported in their paper.

We also evaluate our work on Wiki-KBP dataset [1, 2] and the results are compared with Ren et al. [3] under their relation extraction setting. We reach a 6% higher accuracy than them on the same test set. This improvement validates the positive impact of using synthetic explanations on relation extraction.

We further evaluate our framework on four other relations for which we obtain entity pairs, sentences, and explanations ourselves. The F1-score obtained by our approach is comparable or better than that of Hancock et al., which indicates that our approach performs better. We report our results with different feature extraction tools to show the strength of the selected tools. Finally, some experiments are done with different combinations of the features to evaluate the direct influence of the selected feature sets on the results.

Chapter 2

Background

Building an accurate automated relation extraction system often requires developing or employing accurate tools for sentence parsing and feature extraction. The developed system in this thesis makes use of sentence embedding techniques, entity types and some labeling functions. We extract and compare these three sets of features (and their combination) and feed them to a special type of neural network called family tree. In this chapter, we review the background material that is referenced in this thesis and may help provide a better understanding of our developed features and our automated relation extraction algorithm.

2.1 Dependency Parsing

Ambiguities in sentences make language interpretation hard. Parsing is a formal way for resolving structural ambiguities. There are two types of parsing: Dependency parsing which focuses on detecting the relations between words, and Phrase structure parsing, which focuses on identifying phrases and their recursive structure [28]. A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between "head" words and words which modify them. For more details look at Chapter 13 of Jurafsky and Martin [28].

In this thesis, we choose Stanford Dependency parser, developed by Chen and Manning [29], to find the dependency trees of our sentences. The Stanford Dependency parser is a fast transition based parser which produces typed dependency parses of natural

TABLE 2.1: Explanation of the notations used in the constituent tree example

Symbol	explanation
S	Sentence
NP	Noun Phrase; N stands for Noun
VP	Verb Phrase; V stands for Verb
DT	Determiner
PP	Prepositional Phrase

language sentences. The parser is powered by a neural network that accepts word embedding inputs [29]. In the following subsections, we will describe the step-by-step process of creating a dependency tree.

2.1.1 Dependency trees

Generating a dependency tree of a sentence can be done by making the constituency tree of the sentence, which is defined as "Breaking down a sentence into its constituent parts". The constituency tree of a sentence is generated using the rules of a Context-Free Grammar (CFG) where a set of production rules describe all possible strings in a given formal language. Production rules are simple replacements; for example, the rule $A \rightarrow \alpha$ replaces A with α. For more details about the notations and rules see Chapter 12 of Jurafsky and Martin [28]. The following example shows the constituency tree of the sentence "The man walked to the park."

$$\underbrace{S}_{\text{sentence}} \rightarrow \underbrace{NP}_{\text{The man}} \quad \underbrace{VP}_{\text{walked to the park}}$$

$$\underbrace{NP}_{\text{The man}} \rightarrow \underbrace{DT}_{\text{The}} \underbrace{N}_{\text{man}}$$

$$\underbrace{VP}_{\text{walked to the park}} \rightarrow \underbrace{V}_{\text{walked}} \quad \underbrace{PP}_{\text{to the park}}$$

$$\underbrace{PP}_{\text{to the park}} \rightarrow \underbrace{P}_{\text{to}} \underbrace{DT}_{\text{the}} \underbrace{N}_{\text{park}}$$

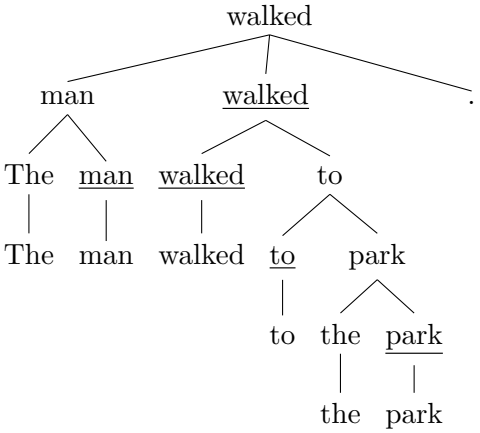
Table 2.1 explains the symbols used in our parsing example. When a phrase-structure parse contains additional information in the form of grammatical relations and function tags, these tags can be used to label the edges in the resulting tree. When applied to the parse shown above, this algorithm produces the dependency structure [30].

2.1.1.1 Generating dependency parse trees

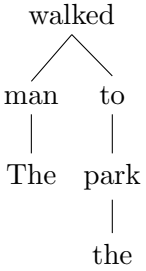
The first step in generating a dependency parse tree is to find the head words of the tree. Collins [31] provides some rules that can be followed for finding the head words. The following shows a subset of those rules. (The head word is underlined in each composition.)

- S → NP VP
- VP → V NP PP
- PP → P NP
- NP → DT N PP

Considering the mentioned rules, we can generate the dependency parse of the sentence "The man walked to the park." with the underlined head words.



Here is a path between the words "the man" and "the park" which gives the dependency relationship between these words.



2.2 Word and sentence embedding

Learning continuous representations of words has a long history in natural language processing [32]. These representations are typically derived from large unlabeled corpora using co-occurrence statistics [33], [34].

Today, word and sentence embeddings have become an essential part of any Learning-based natural language processing system. More specifically, there has been more interest in Universal Embeddings, which are pre-trained on a large corpus and can be used in a variety of tasks to automatically improve their performance by incorporating some general word/sentence representations learned on the larger dataset. While several works augment unsupervised approaches by incorporating the supervision of semantic or syntactic knowledge, purely unsupervised approaches have seen interesting developments in 2017–2018. Two notable works are FastText [35], an extension of word2vec [36], and ELMo [37], the state-of-the-art contextual word vectors.

Embedding approaches turn textual features into numerical ones, which is necessary when the features are fed into neural networks. In this thesis, we use sentence embedding to feed our neural network with numerical features of the sentences. We want to train our model on whole sentences instead of phrases, and we choose a system that provides sentence embedding. As a state-of-the-art embedding tool, FastText is shown to give one of the best pre-trained sentence embedding models [38]. In the following, we discuss the advantages of using sentence embeddings and briefly explain how FastText works.

2.2.1 Embedding Models

Word embedding systems can be obtained using two models (both involving Neural Networks): Common Bag Of Words (CBOW) and Skip Gram [39].

2.2.1.1 CBOW vs Skip-gram Model

Both Skip-Gram and CBOW methods learn the underlying word representation for each word using a neural network. Since learning word representations is essentially unsupervised, a way for creating labels to train the neural network is required. Skip-gram and CBOW are two ways of creating these labels.

CBOW is learning to predict a word by its context, and it maximizes the probability of a target word by looking at its context. Consider the sentence: “Have a [...] day”; the CBOW model will predict that the most probable word inside the brackets are “Great” and “nice”. Words like “delightful” will get much less attention in the model, because it is designed to predict the most probable word.

On the other hand, the Skip-gram model is designed to predict the context. Given the word “delightful” it must understand it and tell there is a huge probability that the context is “Have a [...] day”, or some other relevant context [40].

According to Rong [39], Skip-gram works well with small amounts of the training data, represents well even rare words or phrases. CBOW, on the other hand, is several times faster to train than the skip-gram and has a slightly better accuracy for frequent words.

2.3 Named Entity Recognition

Named Entity Recognition (NER), also known as entity extraction, classifies named entities that are present in text into pre-defined categories such as ‘individuals’, ‘places’, ‘organization’, ‘cities’, ‘dates’, etc.

A good Named Entity Recognizer can automatically scan an entire article and reveal which are the major people, organizations, and places discussed in it. Knowing the relevant types of named entities of an article can help in automatically categorizing the article in a predefined hierarchy.

The usage of NER in this thesis is to extract the types of subject and object entities in each sentence as some features. For example, in Spouse relation, the type of the entities is “Person”. To generate this set of features, we use FIGER [1] which is shown to work well for NER tasks [3].

2.3.1 The FIGER system

FIGER is a named entity recognition tool and we use its pre-trained model for our feature extraction. Ling and Weld [1] divide the whole process of the NER into a pipeline. Given a sentence in plain text as input, first, the sentence is segmented and candidates to be

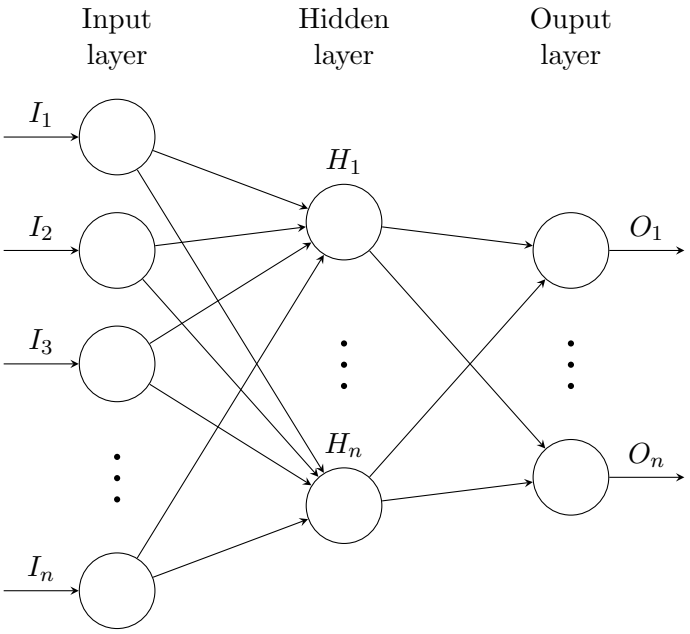


FIGURE 2.1: Simple neural network with one hidden layer.

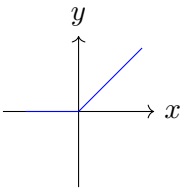
tagged are detected. Second, they apply a classifier to the identified segments and output tags of segments [1]. The most significant advantage of their system is using Freebase [41] tags. As Freebase has around thousands of types and its tags are comprehensive, they filter the tags by only keeping well-maintained types (the ones with curated names, e.g. /location/city).

2.4 Learning by Neural Networks

A neural network (NN) is a computational nonlinear model based on the neural structure of the human brain which is able to learn how to perform tasks like classification, prediction, decision-making, visualization, and others just by considering examples. A Neural network consists of artificial neurons or processing elements and is organized in three interconnected layers: input, hidden(s), and output. Figure 2.1 shows a simple neural network with one hidden layer. The input layer contains input neurons that send information to the hidden layer and the hidden layer sends data to the output layer.

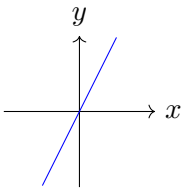
Tanh Function:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$



Logistic (Sigmoid) Function:

$$f(x) = \frac{1}{1 + e^{-x}}$$



Relu Function:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

Linear function:

$$f(x) = ax$$

FIGURE 2.2: Some of the common activation functions

Learning a neural network is categorized as supervised learning, which means the system uses the labels provided by the user and tunes model's parameters in order to improve its prediction performance. For this thesis, the model parameters are defined in Table 4.16 and the evaluation metrics are discussed in Chapter 4. The learning process is done by optimizing an objective or cost function which can be the distance between the desired targets and the outputs of the model. Minimizing the cost function is performed by changing the parameters of the model [42].

Every neuron has weighted inputs also known as synapses, an activation function (which defines the output given an input), and one output. Synapses are the adjustable parameters that convert a neural network to a parameterized system. The weighted sum of the inputs produces the activation signal that is passed to the activation function to obtain one output from the neuron. Commonly used activation functions are linear, step, sigmoid, tanh, and rectified linear unit (ReLu). Figure 2.2 gives some of the common activation functions, a couple of which are used in our models (see Table 4.16).

Different types of neural networks have been applied on natural language processing tasks, and they differ for each task in terms of speed and accuracy. The following shows a list of some of the neural networks which are applied on different NLP tasks.

- Multilayer Perceptron

A multilayer perceptron (MLP) has three or more layers. It utilizes a nonlinear activation function which lets it classify data that is not linearly separable. Every node in a layer connects to each node in the following layer making the network fully connected. Speech recognition [43] and Machine Translation [44] are examples of using MLP networks.

- Convolutional Neural Network (CNN)

  A CNN contains one or more convolutional layers, pooling or fully connected, and uses a variation of multilayer perceptron. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. This operation allows the network to be deeper with much fewer parameters. Semantic parsing [45] is an example use case for CNNs in NLP.

- Recurrent Neural Network(RNN)

  A recurrent neural network (RNN), unlike a feedforward neural network, is a variant of a recursive artificial neural network in which connections between neurons make a directed cycle. It means that the output depends not only on the present inputs but also on the previous step's neuron state. This memory lets users solve NLP problems like text classification [46] or text summarization [47].

In this thesis, we use a MLP (Multilayer Perceptron) Network. We design our architecture based on the network proposed by Hinton et al. [4] which is used for family tree relation prediction.

2.4.1 Family tree network

Family tree network is the name of the network proposed by Hinton et al. [4]. Their network is defined for finding the relations between people in a family tree. For example, given the family tree in Figure 2.3, the following list shows some outputs of the system:

- input: Roberto, Gina. output: parent
- input: Roberto, Marco. output: parent-in-law

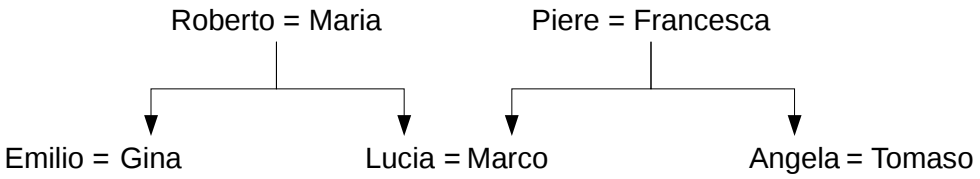


FIGURE 2.3: A sample of family tree. Symbol '=' means 'married to'. "Roberto" and "Maria" are the parents of "Gina".

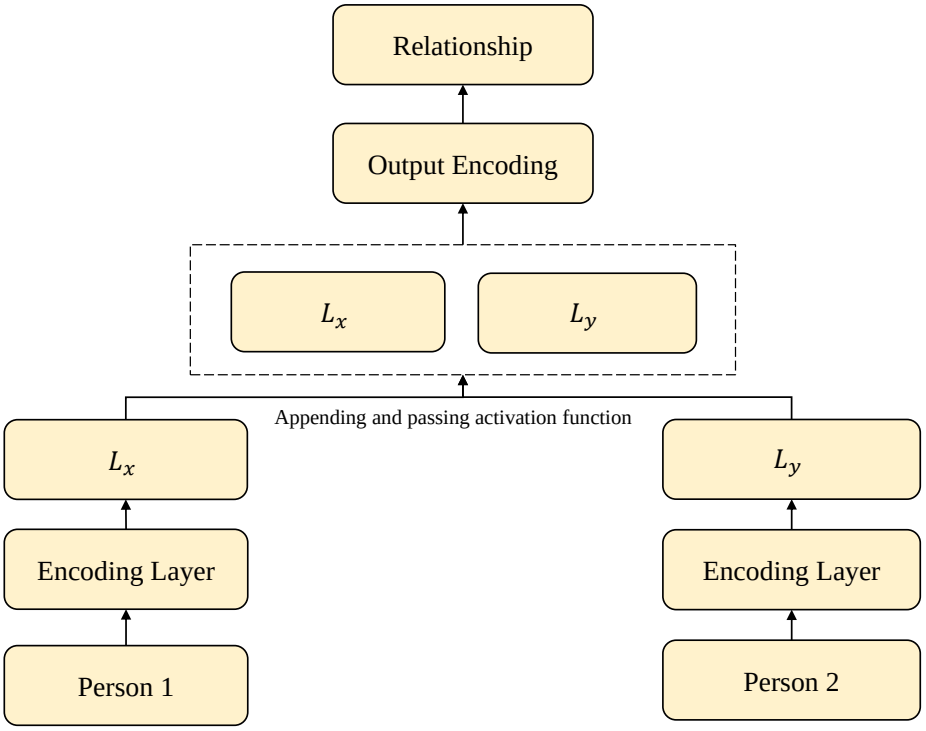


FIGURE 2.4: The architecture of the network proposed in Hinton et al. [4].

Figure 2.4 shows the architecture used in Hinton et al. [4] for their proposed network. In this architecture, two simple MLP networks with one hidden layer are trained independently on the inputs (Person 1 and Person 2) and then the outputs are appended to feed another MLP network for classifying the relationship. Training features separately is the most important characteristic of this architecture.

Chapter 3

The Proposed Method

We build a neural network for the Relation Extraction task that builds on a seminal work by Hinton et al. [4]. Although Hinton et al. [4] use phrases *between* named entities to find the relationship between people in a family tree, we use embeddings of *entire* sentences and find relationships between named entities.

Our work follows the standard practice of breaking the task into the steps of (1) feature generation from raw sentences and (2) classification based on those features. Figure 3.1 shows an overview of our system. For feature extraction, sentence embedding vectors from normalized sentences are extracted. Synthetic explanations are generated from the training data and BabbleLabel features are extracted for each sentence by applying the Labeling functions on them. Labeling functions are generated from the synthetic explanations. The other set of features, types of the entities, are extracted from the raw sentences. The classification part is a network called family tree that is fed by the extracted feature sets.

3.1 Feature Extraction

We extract three sets of features per sample: sentence embedding, types of the entities in the sentence, and the output of the labeling functions—which we call BabbleLabel features.

Most relation extraction approaches use the embedding of the phrase between a pair of entities as an input of their classifiers [3], however, in many cases the important

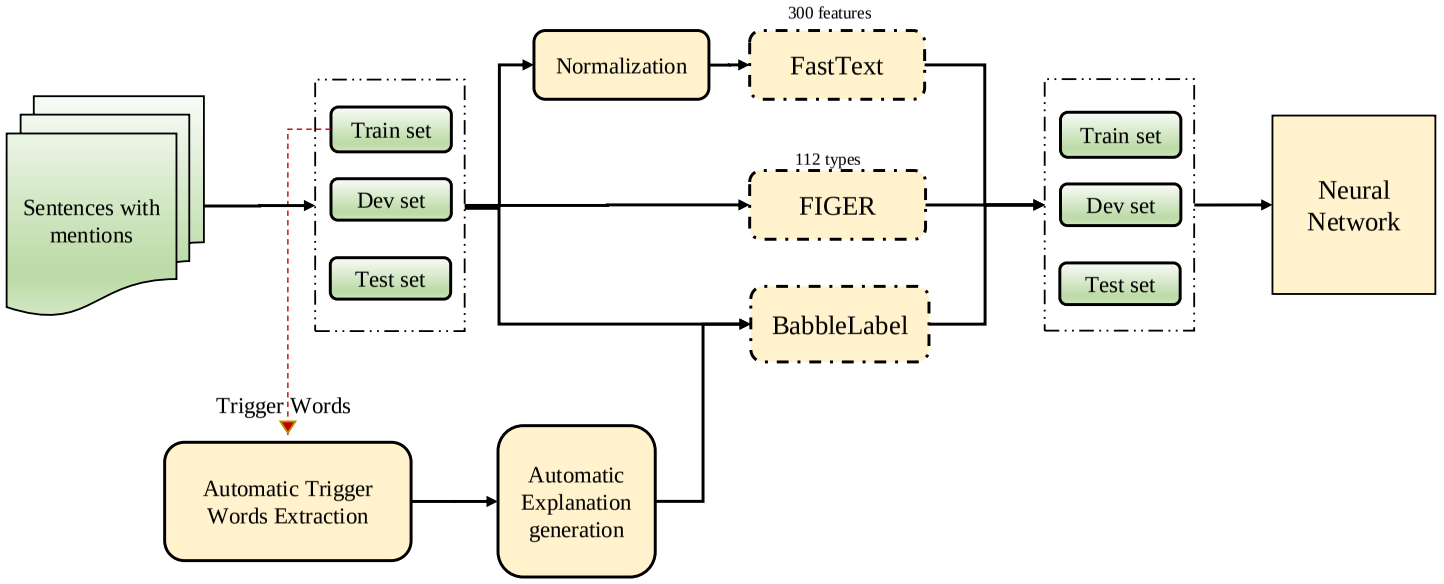


FIGURE 3.1: An overview of the system. Lined boxes are the programs written by us and dashed boxes show the features extraction tools.

words and phrases for detecting the relations come in other parts of the sentence. For example, in sentence "SUBJECT and OBJECT married on May 2012.", the important word for detecting "Spouse" relation is "married", which appears after the SUBJECT and OBJECT entities. For the same reason, we train our network on the whole sentence instead of the phrase between the pair of entities.

Entity Types of subject and object entities is another set of features which is extracted in this work. This feature set improves the system performance particularly when the entity types of the positive and negative training relations are different.

The output of the BabbleLabel labeling functions is the last set of features extracted for training our network. In [27], the authors show that BabbleLabel can achieve a reasonable F1-score with a small number of training samples. Their system performance motivated us to use their labeling functions for generating our features.

For each feature set, we compare alternatives when possible. For example, we compare the Stanford NER and FIGER [1] entity typing systems.

3.1.1 Normalization

As customary we normalize the text to increase the chances of generalization by the model. Besides standard tokenization, we also preprocess each sample as follows:

- All mentions of the words tagged as subject and object in the sentence are replaced with "SUBJECT" and "OBJECT" respectively.
- All named entities except subject and object entity mentions are replaced with "ENTITY".
- All numbers in the sentence are replaced with "NUM".
- All adjectives expressing nationality (e.g., American, Canadian, etc.) are replaced with "NAT".

3.1.2 FastText sentence Embedding

FastText [35] is a library for efficient learning of word representations and sentence classification. This tool represents a document by the average of its word vectors and allows the word vectors to be updated through Back-propagation during training.

In this thesis, we use a pre-trained FastText model as our sentence embedding tool. The input is the normalized sentence and the output is a vector of length 300.

3.1.3 FIGER entity type recognition

Entity types are known to help relation extraction as they can filter out candidate relations immediately [48]. Thus, we extract fine-grained types of the entities in the sentences with the FIGER [1] method, which provides 112 types. For comparison, we conduct our experiments with the standard Stanford NER type classifier as well and the results are compared.

3.1.4 BabbleLabel system

As mentioned earlier, BabbleLabel [27] is a framework for labeling large samples of sentences express a specific relation (e.g., Spouse), which in turn can be used to train classifiers. In the procedure described by the authors, human annotators provide a small number of natural language explanations (less than 50) for each label assigned to an example. These explanations are parsed into logical forms labeling functions that heuristically map examples to the binary labels. Figure 3.2 shows an overview of the BabbleLabel framework. This framework has training and testing part. Labeling functions are generated in training part and they are used for labeling large number of sentences in test part. An example of labeling functions is shown in Section 3.1.4.1.

Hancock et al. [27] found that using the output of their labeling functions as features to be *less effective* for relation extraction than using a bag of n-grams from the phrases between the related entities [27, Section 4.3]. Interestingly, we found that the output of the various Labeling Functions we obtain to be useful features. Moreover, they report to have found a non-neural classifier to be their best choice. We, on the other hand,

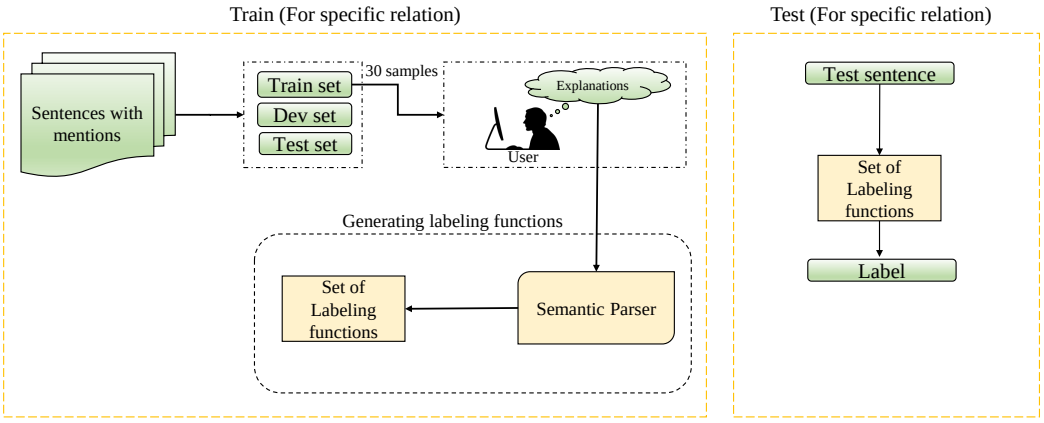


FIGURE 3.2: An overview of the BabbleLabel framework. Labeling functions are generated using the manually written explanations.

report much better results than them, on their dataset, using a neural relation extraction system.

3.1.4.1 Labeling Functions

BabbleLabel has a simple rule-based Semantic Parser that takes a natural language explanation and gives a set of Labeling Functions, each an executable Python code ready to be used for feature extraction. Their system is provided with some grammatical rules that makes the parser generate valid parses for each explanation. After generating the parses, there are some Filter Banks to sift the parses. For example, in the explanation in Example 1.1, the word "right" can be interpreted as either "immediately" or simply "to the right". The latter interpretation results in a function that is inconsistent with the associated example (since "'s wife" is actually to the left of person 2), so it can be safely removed. At the end, the system returns all valid and filtered parses corresponding to the entire explanation [27].

As mentioned in previous sections, we run the BabbleLabel system and extract the output of Labeling Functions as features for our system. An example of labeling functions for Spouse relation is shown in Example 3.1.

Example 3.1.

```
def LF(x):
return (1 if "'s wife" in
between(x.person1, x.person2)
```

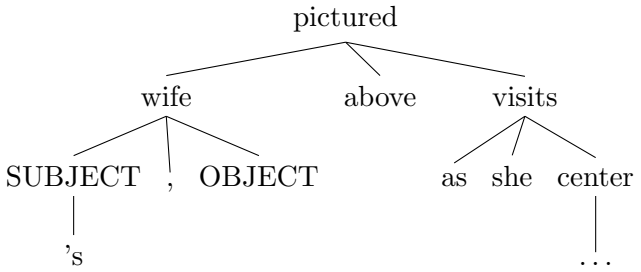


FIGURE 3.3: Parse tree of the sentence in Example 1.1 for finding the shortest path between the entities.

```
    else 0)
```

3.2 Generating Synthetic Explanations

To reduce the cost of procuring training data, we experiment with an automatic system for relations extraction. To achieve this aim, we generate the explanations automatically using the shortest path between the subject and object entities in the dependency tree of each sentence, filtering out those sentences in which we cannot find *trigger* words for the relation in the shortest path between the related entities in the sentence. Remain sentences are considered as the candidate sentence for generating explanations.

Parsing the sentences and requiring trigger words to appear in the shortest path between the annotated entities is expected to remove noise introduced by sentences that express different relations. This type of noise is common, especially for popular pairs of entities. For example, many politicians are born in the country they become president or prime minister of. Similarly, many soccer players become managers of teams they previously played for. In the pure distant supervision approach, all sentences mentioning a pair of entities are taken as expressions of all relations between those entities. In contrast, by requiring a trigger word (e.g., coach) we can filter out sentences expressing other relations (i.e., noise).

3.2.1 Standford Dependency parser

We use the dependency parser of the Stanford CoreNLP tool [49] to process each sentence. The Stanford typed dependencies representation was designed to provide a simple

description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. In particular, rather than the phrase structure representations that have long dominated in the computational linguistic community, it represents all sentence relationships uniformly as typed dependency relations [49].

3.2.2 Shortest path

We find the shortest path between the tokens referring to the subject and object entities in sentences. Using the shortest path between a pair of entities helps to reduce the chance of having trigger words of other relations in the path.

Looking at Figure 3.3, the shortest path between the subject and the object in the tree is [SUBJECT, wife, OBJECT], corresponding to the sentence in Example 1.1 on page 4. Notice that the shortest path between the subject and the object entities contains a trigger word for the Spouse relation, namely *wife*. Therefore, we take the entire phrase in the shortest path ("'s wife" in the example) and use it as part of the synthetic explanation.

3.2.3 Filtering

We performed the following filtering steps to reduce noise and to produce a more precise training set. Since the example given for each rule is a normalized sentence, the entity names are replaced with the token ENTITY.

Rule 1: Remove sentences when there is no node between SUBJECT and OBJECT in the shortest path. Here is an example of sentence being removed by this rule.

"In all seriousness, ENTITY1 sends best wishes to OBJECT and SUBJECT ."

Rule 2: Remove sentences that have trigger words from the negative relation sets in the shortest path. The following sentence is an example of a removed sentence by this rule:

"Son of SUBJECT and brother to OBJECT and father to ENTITY1."

Relation between SUBJECT and OBJECT is 'parent'. Positive trigger word is "Son" and negative trigger word is "brother".

Rule 3: Remove sentences when a trigger word of negative relations appears as a *sibling* of either the SUBJECT or OBJECT in the parse tree. Here is an example sentence where this rule applies.

"Family, Daughter of ENTITY1 executive OBJECT and actress ENTITY2, older brother SUBJECT."

Relation between SUBJECT and OBJECT is 'parent'. Positive trigger word is "Daughter" and negative trigger word is "brother" which is the sibling of SUBJECT in the parse tree of the sentence.

We arrived at these rules empirically by inspecting a sample of mistakes of our method on the development set.

3.2.4 Explanation Generation

For generating explanations automatically, in each relation extraction task, we use trigger words for the positive and negative sentences of a given relation and the shortest path of each sample. Trigger words can be defined by user or collected from the training data automatically.

For extracting trigger words from the training sentence without the human interference, we collect the 15 most frequent words (if possible) between the subject and object entities in the sentences related to each relation. Thus, for each relation, at most we have 15 trigger words.

Each explanation is derived from a template that requires a name, a condition involving a trigger word, a candidate for the condition from training sentences and a label. Candidates are the sentences that have the trigger word between the subject and object entities. The process is shown in Figure 3.4.

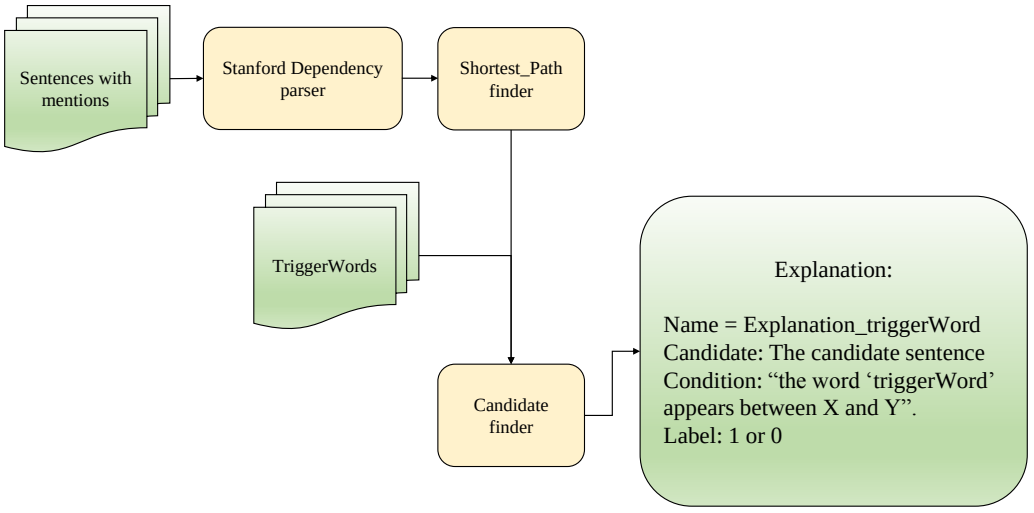


FIGURE 3.4: Finding candidate sentence for generating synthetic explanations.

For example, from the following ClueWeb sentence:

> "The Star of David is so named after King David, the father of King Solomon,"

one can generate a positive label for the relation Parent, because of the trigger word "father" in the path. Furthermore, the explanation for this positive label is:

> Name: explanation.1
> Candidate: "The Star of David is so named after King David, the father of King Solomon."
> Label: 1[1]
> Condition: "the word 'father' appears between X and Y"[2]

By following this procedure we are guaranteed to generate explanations that are compatible with the original explanation parser of Hancock et al. [27]. Thus, we can obtain labeling functions that work exactly as the ones derived from manually produced explanations.

[1] In the BabbleLabel language, positive and negative labels are shown with 1 and 2 respectively.
[2] In the BabbleLabel language, X and Y stand for placeholders for the subject and object entities.

Explanations for negative labels are of two forms: (1) the presence of a hand-picked negative trigger word or the absence of a positive trigger word. Hand-written explanations (by user) can have various types of structures depending on the user. However, our automatically generated explanations have the same structure as:

- Positive samples:

  "A positive trigger word appears between X and Y."

- Negative samples:

  "A negative trigger word appears between X and Y."

  "No positive trigger word appears between X and Y."

Note that this types of explanations are very simple in comparison with manually written explanations by user.

3.3 Binary Classification

We use a neural network for the Relation Extraction task that is inspired by the familyTree network of Hinton et al. [4]. Our network, depicted in Figure 3.5, takes three sets of features as input:

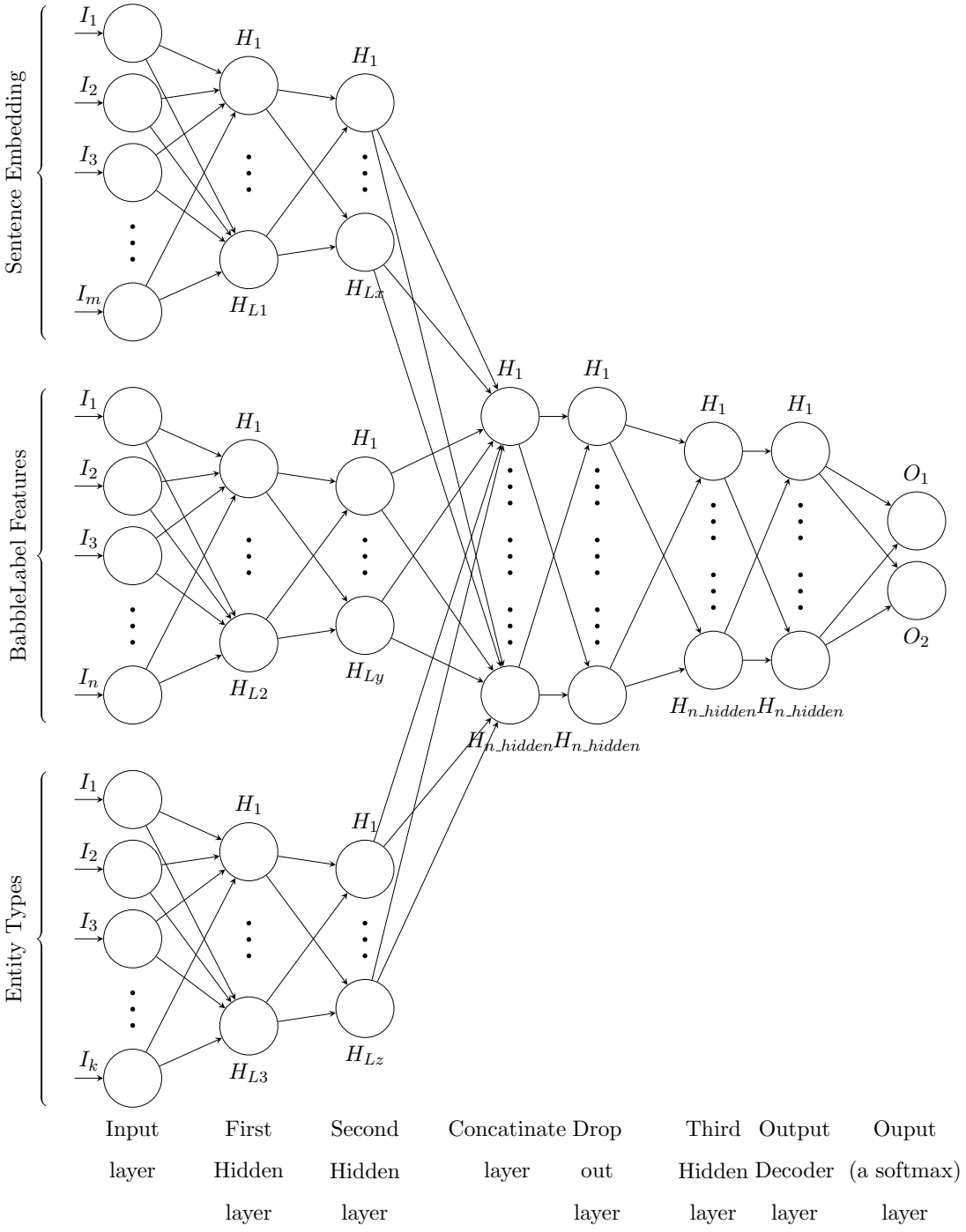
1. Output of BabbleLabel labeling functions (derived from the explanations) on the input sentence;
2. A representation of the entire sentence;
3. Embedding of entity types.

As it is shown in Figure 3.5, our network has two main parts. First, three individual networks with one hidden layer have been trained on the sets of features. The number of neurons are different for these networks. Second, a five layers fully connected network gets the combination of outputs of the first part. As the network is a binary classification network, the output layer of the network is a softmax layer and has two neurons.

As mentioned in Section 2.4.1, training the feature sets independently is the most significant property of this architecture which helps the network to find the importance of

each feature set. Our developed network is different from the network proposed by [4] in terms of the number of inputs, the number of layers and the hyper-parameters.

FIGURE 3.5: Architecture of our defined network which gives sentence embedding vector, out put of BabbleLabel Labeling functions and FIGER types of entities as inputs.



3.3.1 Learning by FamilyTree

Our proposed network named FamilyTree, contains four parts. The first three parts are the encoding layers for each feature set and the last part is the combination of layers for predicting the output. Each part has a similar architecture as MLP networks.

In MLPs, learning occurs in the neurons by changing the connection weights, based on the amount of error in the output compared to the expected result. MLP training process includes three main steps; Forward pass, Loss calculation and Backward pass. The following briefly describes each step.

- Forward pass

  In this step the input is passed to the model and is multiplied with wights at every layer.

- Loss calculation

  An output is generated and the loss is calculated based upon the predicted output and the truth label from dataset.

- Backward pass

  After calculating the loss, weights of the model are updated by backpropagating the loss using gradient. In this step, weights will adjust according to the gradient flow in that direction.

Chapter 4

Experimental Results

In this chapter, we first present our datasets and the experimental setup for our evaluation. This is followed by our evaluation of the proposed method.

Our evaluation is divided into a few parts. First, we report the results of our system with different tools that we use to generate features. This is to find the best tools or settings for feature extraction (Section 4.2.1). After choosing the best tools for generating the desired features, precision, recall and F1-score will be reported for the network trained on every possible combination of the features (Section 4.2.2). Third, we show that our automatically generated explanations are comparable with the human produced ones by comparing the results produced by our network on both automatic and manual explanations (Section 4.2.3). The next experiment is a comparison, in terms of F1-score, of our system and the BabbleLabel system published in [27] for 'Spouse' relation (Section 4.2.4). We investigate the false-positive instances of all relations to better understand why the network miss-classifies them (Section 4.2.5). In another experiment with two other relation extraction tasks, namely location-contain and place-of-birth, we further evaluate our method on those relations (Section 4.2.6). At the end, to compare our system with one other related work, Ren et al. [3], we use a public dataset (Wiki-KBP [1, 2]) and evaluate our best model on its test set (Section 4.2.7).

In the last section of the chapter, we provide some details for replicating the results.

4.1 Dataset Description

4.1.1 Relation extraction tasks

BabbleLabel Spouse Dataset. We test our approach on all data provided by the BabbleLabel authors. Hancock et al. [27] test their model on three relations but make available data for only the Spouse relation [5]. Moreover, although their paper mentions that they use 30 explanations for the Spouse relation, they make only 10 explanations available.

ClueWeb Dataset. We build a corpus of sentences, explanations, and labels for five relations: Spouse, Parent, Book-author, place-of-birth and location-contain. Our training data comes from ClueWeb sentences [6] annotated with Freebase entity identifiers [50]. For each relation, we gather all sentences mentioning pairs of Freebase entities belonging to the corresponding relation.

For the first three relations, we test with both manual explanations we produced ourselves, following the guidelines in Hancock et al. [27], and automatically generated explanations from sentences that contained trigger words for the relations. After getting results for those relations, we pick our best model and test other two relations, place-of-birth and location-contain, only with the best model.

Wiki-KBP Dataset. To have a comparison with related works in relation extraction, we choose a public dataset named Wiki-KBP [1, 2] and test our model on that. It uses 1.5M sentences sampled from around 780k Wikipedia articles [1] as training corpus and 14k manually annotated sentences from 2013 KBP slot filling assessment results [2] as test data.

This dataset has 7 relation types including: children, countries-of-residence, country-of-death, country-of-birth, parents, religion and None. We test our best model on this dataset and compare our results with Ren et al. [3].

4.1.1.1 Negative samples for training and testing

BabbleLabel Spouse Dataset. The BabbleLabel Spouse dataset contains labeled positive and negative samples for both training and testing sets.

TABLE 4.1: Statistics of BabbleLabel Spouse datasets [5]. The relations used as negative samples are shown in the last column.

Relation	Train	Test	Dev	negative relations
Spouse	8K	1K	1K	Parent, friend, ...

ClueWeb Dataset. To obtain negative samples for our data, we manually picked ClueWeb sentences relating entities of similar types belonging to other relations in Freebase. For example, for the Parent relation we gathered samples of Spouse and other familial relationships (e.g., cousin, brother, sister, etc.). Explanations in such cases amount to either the lacking of the corresponding positive trigger words of the relation or the presence of the negative trigger words for the relation. For the Book-author relation, we used sentences mentioning a person and other arts (e.g., a song).

Wiki-KBP Dataset. For each relation in this dataset, we considered the samples of the relation as positive sentences and the samples of other 6 relations as negative sentences.

4.1.1.2 Splitting data to training and testing sets

BabbleLabel Spouse and Wiki-KBP datasets. These two datasets have training and testing instances separately. We considered 10% of their training set as our validation sets.

ClueWeb Dataset. This data is divided in 80%,10% and 10% splits for training, validation and test sets.

For all datasets, each sample of the dataset includes a subject entity, an object entity, a sentence containing the subject and the object entities and the target relationship between subject and object entities. Statistics for the datasets are reported in Tables 4.1, 4.2 and 4.3.

4.1.1.3 Trigger Words

The following lists show the hand-picked trigger words for each relation extraction task on Clueweb and BabbleLabel Spouse datasets. These trigger words are provided by user without using training data.

TABLE 4.2: Statistics of Wiki-KBP [1, 2] dataset. The relations used as negative samples are shown in the last column. The number of Train, Test and Dev sentences are the same for all relations and are 1.35M, 14K and 150K respectively.

Relation	negative relations
parents	children, countries-of-residence, country-of-death, country-of-birth, religion, None
children	countries-of-residence, country-of-death, country-of-birth, parents, religion, None
country-of-birth	children, countries-of-residence, country-of-death, parents, religion, None
country-of-death	children, countries-of-residence, country-of-birth, parents, religion, None
countries-of-residence	children, country-of-death, country-of-birth, parents, religion, None
religion	children, countries-of-residence, country-of-death, country-of-birth, parents, None
None	children, countries-of-residence, country-of-death, country-of-birth, parents, religion

TABLE 4.3: Statistics of our generated dataset [6]. The relations used as negative samples are shown in the last column.

Relation	Train	Test	Dev	negative relations
Spouse	9.6K	1.2K	1.2K	Parent, family-members
Parent	8.8K	1.1K	1.1K	Spouse, family-members
Book-author	16K	2K	2K	politicians-party, artist-song
place-of-birth	17.8K	2.3K	2.3K	place-of-death
location-contain	24K	3K	3K	cities-of-states, states-of-country

- **Spouse**:

  Positive trigger words: 'married', 'wife', 'husband', 'marriage', 'widow', 'dating', 'fiance', 'spouse'.

  Negative trigger words: 'brother', 'sister', 'half-brother', 'sibling', 'half-sister','son', 'dad', 'mom', 'daughter', 'father', 'mother', 'child', 'parent', 'grandson', 'grand-daughter', 'adopted', 'adopt','friend'.

- **Parent**:

  Positive trigger words: 'son', 'dad', 'mom', 'daughter', 'father', 'mother', 'child', 'parent', 'adopted', 'adopt'.

  Negative trigger words: 'married', 'wife', 'husband', 'spouse', 'marriage', 'widow', 'dating', 'fiance','brother', 'sister', 'half-brother', 'sibling', 'half-sister','friend'.

- **Book-author**:

  Positive trigger words: 'author', 'book', 'novel', 'wrote', 'written', 'published', 'story', 'writes', 'writer', 'writing', 'poem'.

  Negative trigger words: 'album', 'released', 'release', 'recorded', 'song', 'record', 'recording', 'CD', 'music', 'band', 'leader', 'candidate', 'elected', 'president', 'chairman', 'head', 'leadership', 'politician', 'government'.

- **place-of-birth**:

  Positive trigger words: 'born', 'grew', 'native', 'birthplace'.

  Negative trigger words: 'died', 'killed', 'death', 'president of'.

- **location-contain**:

  Positive trigger words: 'place in', 'city in', 'country in', 'province in', 'located in', 'commune of', 'capital of', 'city of', 'province of', 'home to'.

  Negative trigger words: 'near to', 'south of','north of, 'west of', 'east of', 'miles of'.

To have a fair comparison with Ren et al. [3], we select our trigger words for the relations in Wiki-KBP dataset [1, 2] automatically from the training sentences. Positive trigger words for the 7 relations in Wiki-KBP dataset is shown in the list below. We consider trigger words of other 6 relations as negative trigger words for each relation.

- **parent**: 'son', 'daughter', 'mother', 'sons', 'Prince', 'children', 'child', 'grandson', 'daughters', 'parents', 'grandfather', 'Queen'.
- **children**: 'created', 'half-brother', 'half-sister', 'child'.
- **religion**: 'leader', 'cleric', 'Church', 'Muslim', 'religion', 'Christian', 'Muhammad', 'religious', 'Al-Azhar', 'Imam', 'God', 'philosophy', 'prophet', 'John', 'Islam'.
- **country-of-death**: 'city', 'killed', 'years', 'City', 'capital', 'life', 'located', 'Battle', 'age'.
- **country-of-birth**: 'native', 'town', 'birthplace', 'grew'.
- **countries-of-residence**: 'live', 'work', 'moved'.
- **None**: 'played', 'buy', 'laughed', 'paper'.

TABLE 4.4: Statistics of data cleaning part for our generated dataset training sentences

Relation	Original train set	Rule	number of samples removed	cleaned train set
Parent	8800	Rule 1	1580	6166
		Rule 2	682	
		Rule 3	372	
Book-author	16000	Rule 1	3723	11444
		Rule 2	97	
		Rule 3	736	
Spouse	9600	Rule 1	2378	6499
		Rule 2	213	
		Rule 3	510	
place-of-birth	17800	Rule 1	1846	15584
		Rule 2	114	
		Rule 3	256	
location-contain	24000	Rule 1	2121	21360
		Rule 2	327	
		Rule 3	192	

4.1.1.4 Filtering

Table 4.4 shows the statistics of the data removed from the training set of ClueWeb dataset by each rule discussed in 3.2.3. Since we compare our results for the BabbleLabel Spouse and Wiki-KBP datasets with others, we do not apply the filtering rules on those datasets.

4.2 Experiments

4.2.1 Choosing our feature extraction tools

In this section, we ignore the Babblelabel features. Thus, sentence embedding and entity types are the inputs of our network. These set of experiments are done to find the best sentence embedding and entity type recognition tools among some of the available tools. Table 4.5 shows the result of our system based on various features obtained with alternative tools for the same kind of feature.

For sentence representation, we tested with Bag_of_Bigrams as suggested by Hancock et al. [27] and also with FastText [35] sentence embedding. For the sake of efficiency, we prune infrequent bigrams, i.e. those occurring less than 30 times in the corpus. For

entity types, Stanford_NER [51], which is coarse-grained and only provides 9 types, and FIGER [1], which provides 112 types have been tested.

Table 4.5 shows the results of our evaluation. Four experiments are conducted for each relation extraction task, each with a different feature extractor tools. Models are Stanford_NER + Bag_of_Bigrams, Stanford_NER + FastText, FIGER + Bag_of_Bigrams and FIGER + FastText. Precision, Recall and F1-score are reported for every experiment. The maximum and minimum F1-score are used as the measurements for final comparison.

By comparing the F1-score between FIGER+FastText model (last part of the table) and FIGER+Bag_of_Bigrams model (third part of the table) we can decide about the impact of using FastText as the sentence embedding tool. In all of the three relations, F1-score of FIGER+FastText model is better than FIGER+Bag_of_Bigrams one. This superiority can be seen in comparing the results of Stanford_NER+Bag_of_Bigrams model (first part of the table) with Stanford_NER+FastText model (second part of the table) too. Thus, we can conclude that FastText improves performance of the system in comparison with Bag_of_Bigrams. This is not surprising since the output of FastText is the embedding vector of the whole sentence while the output of Bag_of_Bigrams is the bag of bigrams of the phrase between the pair of entities. Also, unlike the Bag_of_Bigrams which uses frequent bigrams, FastText is a pre-trained model on a large corpus. So the FastText model is more general than the Bag_of_Bigrams approach.

The other set of experiments is to compare Stanford_NER and FIGER as entity type recognizer tools. Note that although Spouse and Parent are the relations between entities of type Person, which is one of the most well supported types by NER tools, FIGER has a better performance than Stanford_NER tool on these two relations. Providing 112 types for entities is not the only benefit of FIGER. According to [1], FIGER uses more textual features than Stanford_NER tool which makes it more accurate. The F1-score of FIGER is around 10% better than Stanford_NER reported in [1]. By comparing FIGER+FastText with Stanford_NER+FastText results, we conclude that using FIGER as the entity type recognizer tool can improve the system performance. The advantage of using FIGER is more visible in Book-author relation. Since FIGER provides different tags for /person/author and /person/artist, this progress is reasonable in this relation.

TABLE 4.5: Precision, Recall and F1-score on Spouse, Parent and Book-author relations varying features

StNER+BoBigr			
Relation	Precision	Recall	F1-score
Spouse	70.25±3.25	78.12±2.88	73.98±2.97
Parent	68.75±3.25	81.12±2.75	74.43±3.07
Book-author	89.75±2.25	88.12±2.25	88.93±2.13
StNER+FstT			
Relation	Precision	Recall	F1-score
Spouse	73.00±3.12	84.25±2.50	78.22±2.80
Parent	69.50±3.13	82.88±2.62	75.60±3.02
Book-author	93.38±1.75	89.88±2.13	91.59±1.82
FGR+BoBigr			
Relation	Precision	Recall	F1-score
Spouse	73.50±3.00	85.88±2.38	79.21±2.83
Parent	71.12±3.12	84.25±2.62	77.13±2.89
Book-author	97.12±1.25	98.25±1.00	97.68±1.00
FGR+FstT			
Relation	Precision	Recall	F1-score
Spouse	87.62±2.25	91.25±2.00	89.40±2.07
Parent	88.25±2.25	92.50±1.88	90.33±1.95
Book-author	97.12±1.25	98.75±0.88	97.93±0.94

By these comparisons, as shown in Table 4.5, we find that FastText and FIGER are the best tools that can be used in all relation extraction tasks. Thus, we discard Stanford_NER and Bag_of_Bigrams tools in our further experiments.

4.2.2 Results and discussion over the relation extraction tasks

All relation extraction tasks are evaluated with seven variations of the network, each with a combination of one, two and three feature sets as the input of the network. Thus, various types of networks in terms of the type of features and the number of the inputs has been evaluated in this section to find the best model. To have a better explanations of those defined types of networks we call them as modes. In Tables 4.6, 4.7, 4.8 and 4.9, precision, recall and F1-score for all modes of the network for Spouse, Parent and Book-author relations are reported.

As our first set of experiments, the models trained on only one of the features are evaluated. As shown in Tables 4.6, 4.7, 4.8 and 4.9, in all of our relation extraction tasks, although the networks that are trained only on FIGER or FastText features

do not show high performance, models trained only on BabbleLabel features, have an impressive improvement in precision. However, the recall has suffered by at least 15%. A reason for this behavior is that using explanations helps to predict all samples that belong to those explanations. The problem appears when we encounter samples for which no explanation is provided. In general, although adding BabbleLabel labeling functions to the features, can increase the precision, it makes the recall of the system dependent on the number of explanations.

The next set of experiment is to evaluate the models trained on a combination of two features. The best F1-score for this relation extraction task belongs to Babblelabel+FastText model. The results in this set of experiment is different for each relation. For example, according to Table 4.9, in Book-author relation (unlike the Spouse and Parent relations in Tables 4.7 and 4.8) FIGER+BabbleLable shows a better performance than FastText+BabbleLabel. This observation shows that the impact of the features on the network is different for each relation.

The last set of experiments in this section is to train the system on all of the three feature sets. This model gives the best performance on all relation extraction tasks except the Spouse using Hancock et al data [5]. Fortunately, in this relation (Spouse), the results of the FastText+FIGER+BabbleLabel model is comparable with the best model and the difference is less than 1%. Thus, we can conclude that the FastText+FIGER+BabbleLabel model is the best model for all of the evaluated relations. We use FastText+ FIGER+ BabbleLabel as our best model for comparing with others.

As discussed in Chapter 1, expanding existing Knowledge Bases is one of the most important goal of relation extraction. To achieve this aim, a very precise system is required. Although, the precision is more significant in this problem, we compare our results with the maximum and minimum of F1-score for finding the best model because of our feature sets. BabbleLabel features can make the system precise (e.g.,Table 4.7, experiment with only BabbleLabel features) but the precision is dependent on the number of explanation. However, comparing based on F1-score helps to have a better comparison.

TABLE 4.6: Percentage of Precision, Recall and F1-score on BabbleLabel Spouse dataset [5].

Explan	Source	Method	Precision	Recall	F1-score
30	[27]	Hancok et al. [27]	–	–	50.1
none		FstT	81.73±2.87	54.18±3.11	65.16±2.98
		FGR	83.39±3.44	31.54±3.42	45.76±3.43
		FstT+FGR	57.88±3.37	60.25±3.38	59.04±3.37
10[1]	[27] (manual)	Bbl	77.4±3.22	59.83±3.37	67.49±3.33
		FstT+Bbl	66.25±3.25	72.38±3.12	69.18±3.13
		FGR+Bbl	69.12±3.25	67.50±3.25	68.30±3.19
		Bbl+FstT+FGR	63.88±3.38	73.38±3.00	68.30±3.28
12	autom.	Bbl	75.32±2.78	61.73±3.14	67.85±3.00
		Bbl+FstT	**66.25±3.37**	**73.88±3.00**	**69.86±3.15**
		FGR+Bbl	69.62±3.25	70.00±3.25	69.81±3.13
		Bbl+FstT+FGR	66.62±3.38	71.62±3.25	69.03±3.14

TABLE 4.7: Percentage of Precision, Recall and F1-score on Parent relation of ClueWeb dataset [6].

Explan.	Source	Method	Precision	Recall	F1-score
none		FstT	79.63±1.19	69.89±2.22	74.44±1.87
		FGR	76.80±1.65	62.76±1.17	69.08±1.34
		FstT+FGR	88.25±2.25	92.50±1.88	90.33±1.95
4	manual	Bbl	93.14±2.18	73.41±3.09	82.10±2.78
		FstT+Bbl	78.00±2.88	79.38±2.87	78.68±2.81
		FGR+Bbl	79.50±2.75	74.00±3.12	76.65±2.89
		Bbl+FstT+FGR	**91.62±1.88**	**99.25±0.63**	**95.29±1.25**
53	autom.	Bbl	98.11±0.71	62.20±0.96	76.13±0.83
		FstT+Bbl	69.88±3.12	81.12±2.75	75.08±2.93
		FGR+Bbl	62.88±3.38	77.88±3.00	69.58±3.14
		Bbl+FstT+FGR	90.11±1.89	97.56±1.73	93.68±1.82

TABLE 4.8: Percentage of Precision, Recall and F1-score of Spouse relation of ClueWeb dataset [6].

Explan.	Source	Method	Precision	Recall	F1-score
none		FstT	78.34±2.76	70.11±2.65	74.00±2.70
		FGR	75.76±2.45	63.19±2.25	68.91±2.39
		FstT+FGR	87.62±2.25	91.25±2.00	89.40±2.07
5	manual	Bbl	93.32±2.00	71.76±2.52	81.13±2.31
		Bbl+FstT	78.31±1.78	79.44±1.45	78.87±1.67
		Bbl+FGR	78.71±1.65	72.73±2.00	75.60±1.87
		FstT+FGR+Bbl	**89.97±0.98**	**98.11±1.04**	**93.86±1.00**
27	autom.	Bbl	97.87±1.16	60.01±2.00	74.40±1.65
		Bbl+FstT	70.21±1.67	78.02±1.44	73.91±1.50
		Bbl+FGR	61.98±2.11	77.12±2.11	68.73±1.98
		Bbl+FstT+FGR	90.65±0.67	97.08±0.78	93.75±0.76

TABLE 4.9: Percentage of Precision, Recall and F1-score of Book-author relation of ClueWeb dataset [6].

Explan.	Source	Method	Precision	Recall	F1-score
none		FstT	92.63±1.31	89.28±0.87	90.79±1.02
		FGR	94.25±0.89	87.85±0.94	90.94±0.93
		FstT+FGR	97.12±1.25	98.75±0.88	97.93±0.94
5	manual	Bbl	95.69±1.48	31.82±1.12	47.76±1.25
		FstT+Bbl	93.00±1.88	95.62±1.50	94.29±1.57
		FGR+Bbl	94.12±1.62	98.12±1.00	96.08±1.27
		Bbl+FstT+FGR	**97.25±1.25**	**98.38±0.88**	**97.81±0.94**
91	autom.	Bbl	97.14±1.78	55.81±1.34	70.89±1.56
		FstT+Bbl	95.12±1.62	94.88±1.62	95.00±1.44
		FGR+Bbl	92.88±1.87	97.12±1.25	94.95±1.45
		FstT+FGR+Bbl	95.62±1.50	98.12±1.00	96.86±1.13

4.2.3 Quality of automatically generated explanations

In this section, we compare the results of the network using synthetic explanations with the results of using manually generated ones. In general, it is fair to say that the difference between synthetic and manually generated explanations in our tests was minimal. According to the results of Section 4.2.2, the maximum difference, in terms of F1-score, between manual and automatically generated explanations belongs to the Parent relation and is about 7% for BabbleLabel+FIGER model. This observation supports our claim that automatically generating explanations can reduce the role of humans in labeling data without causing an impressive damage on the performance of the system.

Note that in this thesis, the simplest type of explanations has been generated automatically and the number of generated explanations depends on the number of trigger words and the number of candidates that are found for those trigger words of each relation. For example, we have 32 trigger words for the Spouse relation (including lowercase and uppercase) and the system can only find 12 candidates on BabbleLabel Spouse [5] training data. However, with the same trigger word set, the system is able to find 27 candidates on ClueWeb [6] Spouse training data.

TABLE 4.10: Statistical results (true-positive, false-positive, true-negative and false-negative) of **best model** for each relation extraction task.

Relation	system	TP	FP	TN	FN
Spouse - Hancock et al. [5]	FstT+Bbl- Automatic	147	51	727	74
Spouse - ClueWeb data [6]	FstT+FGR+Bbl- Manual	392	8	756	44
Parent	FstT+FGR+Bbl- Manual	460	2	592	46
Book-author	FstT+FGR+Bbl- Manual	1154	19	795	32

4.2.4 Comparing the result of the Spouse relation with BabbleLabel system

It is worth mentioning that we obtained better results than those reported by Hancock et al. (see Table 4.6), on their dataset, albeit with only a subset of the explanations they used. As mentioned, our method differs from theirs in two ways:

1. We use a neural classifier while they use logistic regression.
2. We use the output of their labeling functions while they use n-grams of relational phrases.

However, we improve the F1-score on their dataset around 18%.

Although in BabbleLabel [27], the authors did experiments on three relation extraction tasks, Spouse, Disease and Protein, the data and written explanations are available only for Spouse relation in [5]. Our system is able to generate 12 explanations for their dataset, and, as a matter of fact, achieve the better performance with those explanations than with the manually derived ones.

4.2.5 Investigating False Positives

In this section, we show some examples of false-positive samples for all mentioned relations to better understand why the network miss-classified them. Table 4.10 shows the statistical results of our system.

For each relation extraction task, we look into the false-positive samples of each class and divide them in some groups.

[1] Although Hancock et al. report using 30 explanations, the dataset they make available contains only 10, which are the ones we used.

Spouse relation

There are 51 samples in the false-positive instances of the network. Every sample is a member of one of the following groups.

1. Group1 (35 sentences). Sentences without any trigger word that shows the Spouse relation.

   Example 4.1. "Actor Jon Hamm and actress Jennifer Westfeldt have called it quits after 18 years, reports Us Weekly."

   SUBJECT: Jon Hamm

   OBJECT: Jennifer Westfeldt

   *Note: the last name of SUBJECT and OBJECT is different.

2. Group2 (4 sentences). There is at least one trigger word but the relationship between SUBJECT and OBJECT is not 'Spouse'.

   Example 4.2. "Widow Kath Rathband, who cared for the hero police officer after he was shot and blinded by killer Raoul Moat in Newcastle in July 2010, will **marry** prison guard John McGee next."

   SUBJECT: Widow Kath Rathband

   OBJECT: Raoul Moat

Parent relation

There are only two samples in the false-positive instances of the best network for this relation. Those two samples are shown in the following list.

1. Group1 (1 sentence): There is no trigger word to show the 'Parent' relationship between SUBJECT and OBJECT.

   Example 4.3. "Were it not for the senior Jennifer Bolt's persistence, Usain would probably not be here."

   SUBJECT: Usain

   OBJECT: Jennifer Bolt

2. Group2 (1 sentence): Since the network and automatic BabbleLabel system provided with only simplest type of explanations, the network cannot classify the second sample correctly (daughter is not between SUBJECT and OBJECT)

   Example 4.4. "Memories of movies past: Over the rainbow in Neverland One day, visual consultant John Napier took his friend, daughter of Judy Garland, Lorna Luft, and her son Jesse Cole, 7, on a walk through the sets he had designed."

   SUBJECT: Lorna Luft

   OBJECT: Judy Garland

Book-author relation

There are 19 false-positive instances in the results of the best network for this relation. Most of them have one of these structures which are common in both Book-author and music-artist relations.

1. Group1 (9 sentences): SUBJECT's OBJECT.

   Example 4.5. "Although the idea has been used since, most notably in L. Sprague de Camp's Rivers of Time stories, Bradbury's tale looks at the consequences of the most minor action of which a person may not even be aware."

   SUBJECT: L. Sprague de Camp

   OBJECT: Rivers of Time

   relation: Book-author

   Example 4.6. "Roy Harper sang the Jethro Tull song Up the 'Pool , on their 1996 tribute album, To Cry You A Song - A Collection Of Tull Tales, and in 1998, Jethro Tull singer Ian Anderson contributed flute to the song, These Fifty Years on Harper's The Dream Society , a concept album based on Harper's life, particularly his youth."

   SUBJECT: Roy Harper

   OBJECT: The Dream Society

   relation: music-artist

2. Group2 (4 sentences): OBJECT by SUBJECT.

   Example 4.7. "The Alchemist by SUBJECT Paulo Coelho is originally written in Portuguese and became a widely translated international bestseller."

   SUBJECT: Paulo Coelho

   OBJECT: The Alchemist

   relation: Book-author

   Example 4.8. "Für Elise by Ludwig van Beethoven was not published during his lifetime, only being discovered forty years after his death."

   SUBJECT: Ludwig van Beethoven

   OBJECT: Für Elise

   relation: music-artist

As we showed, most of the False-Positive samples (in all relations) can be considered as noisy data. This manual checking of false positives shows the strength of our features and the positive impact of using natural language explanations.

4.2.6 Relations involving entities of type location

To further evaluate our system, we did experiments on two other relations that involve entities of type location, known to be a challenging type for this task. We have used our best model (FastText+FIGER+BabbleLabel) with automatically generated explanations for these relations. Table 4.3 shows the statistics of data used for location-contain and place-of-birth relations.

4.2.6.1 Evaluation

The result of our best model (BabbleLabel+FastText+FIGER) on relations that involve locations is shown in Table 4.11. Our system shows a reasonable performance on these types of relations too.

The following shows some examples of the sentences that were miss-classified by the network (False-Positives). Two main problems can be noticed in the false-positives for

TABLE 4.11: Percentage of Precision, Recall and F1-score of extra two relations for the best automatic model (FstT+FGR+Bbl)

Relation	Explan.	Source	Method	Precession	Recall	F1-score
place-of-birth	18	autom.	Bbl +FstT +FGR	78.65±1.36	75.31±2.06	76.94±1.83
location-contain	28			72.31±1.17	73.17±0.82	72.74±0.98

these relation extraction tasks. First, some of the positive and negative examples are similar (see Example 4.12); second, trigger words are absent in some of the sentences.

place-of-birth relation

1. In the following sentence, we cannot find any trigger that shows Korman is born in Chicago:

   Example 4.9. "After a stint in the navy, Korman studied theater in Chicago before going to New York hoping to make it as an actor."

   SUBJECT: Korman

   OBJECT: Chicago

   relation: place-of-birth

2. There are two trigger words in following sentences that show the relation is place-of-birth. But none of those are the candidate for our simple type of explanation. By improving our explanations, these type of sentences can be easily classified with the network.

   Example 4.10. "Born in Paris as Vincent Crochon , he is the son of French actor Jean-Pierre Cassel and the brother of Rockin."

   SUBJECT: Vincent Crochon

   OBJECT: Paris

   relation: place-of-birth

3. The following is another example that we cannot find any trigger word showing the relation in the sentence.

Example 4.11. "An exhibition about Johann Georg Kohl has been prepared together with the University Library of Bremen, and displayed at Washington, Bremen and Dresden."

SUBJECT: Johann Georg Kohl

OBJECT: Washington

relation: place-of-birth

location-contain relation

1. In the following example, the relation is location-contain, but it is hard to predict that for the network due to the existence of 'south of' ('south of' is in the Negative trigger words list for this relation).

   Example 4.12. "Just south of the heart of charming Silverton is the Oregon Garden."

   SUBJECT: Silverton

   OBJECT: Oregon Garden

   relation: location-contain

2. In the following sentence, the relation is location-contain, but the trigger word comes between Sarawak Museum and Sarawak:

   Example 4.13. "Currently at the Dewan Tun Abdul Razak in Kuching, a gallery of the Sarawak Museum just across the road from the main building, is a display of ceramic relics excavated from various sites at the Sarawak river delta."

   SUBJECT: Kuching

   OBJECT: Sarawak Museum

   relation: location-contain

4.2.7 Evaluation on Wiki-KBP dataset [1, 2]

As discussed before, we evaluate our best model (Fasstext+FIGER+BabbleLabel) on the Wiki-KBP public dataset to compare our results with Ren et al. [3]. There are training and testing sentences for 7 different relations in the Wiki-KBP dataset. Our

TABLE 4.12: Precision, Recall, F1-score and Accuracy for all relations in Wiki-KBP dataset [1, 2] for the best automatic model (FstT+FGR+Bbl)

Relation	Precession	Recall	F1-score	Accuracy
parents	0.523	0.785	0.628	0.953
children	0.714	0.500	0.588	0.950
country-of-birth	0.327	0.703	0.447	0.833
country-of-death	0.719	0.771	0.744	0.843
countries-of-residence	0.012	0.109	0.021	0.804
religion	0.111	1.000	0.199	0.968

methods requires positive and negative samples for each relation tasks. For using this dataset in our method, we consider sentences of other 6 relations as negative samples for each of the relations (Section 4.1.1.1). Following [3], samples of 'None' relation are excluded. The precision, recall, F1-score and accuracy for each of the 6 relations are shown in Table 4.12.

4.2.7.1 Changing the task to a multi-class classification

Our system is a binary classification model. For comparing our results with Ren et al. [3], we need to change it to a multi-class classification.

As shown in Figure 3.5, the last layer of our network is a softmax layer which provides a probability for each node. A same test set is used for all of the 7 relations in our method which is a binary classification network. We find the network prediction for each test sample by comparing all of the softmax probabilities from all binary classification networks and setting the maximum one as the prediction of the network. For example, softmax probabilities for "Hakim, 59, died in Tehran." sentence is shown in Table 4.13. As the maximum probability for 'yes' label belongs to the "country-of-death" relation, we consider it as the network prediction.

TABLE 4.13: Softmax probabilities for label 'yes' and 'no' for each relation classified with binary classification network.

Relation	softmax prob. for 'yes'	softmax prob. for 'no'
parents	4.4089714e-07	9.9999952e-01
children	2.8041018e-09	1.0000000e+00
country-of-birth	0.05261817	0.94738185
country-of-death	**0.92989**	0.07011009
countries-of-residence	1.4825741e-07	9.9999988e-01
religion	3.5840893e-04	9.9964154e-01

TABLE 4.14: Accuracy for Wiki-KBP dataset [1, 2] for different methods. Our system achieves the best accuracy.

Method	Accuracy
DS+Peceptron [1]	0.543
DS+Kernel [13]	0.535
DeepWalk [52]	0.613
LINE [18]	0.617
DS+Logistic [21]	0.646
MultiR [17]	0.633
FCM [53]	0.617
Ren et al. [3] (CoType)	0.669
Our method	**0.731**

Following this strategy, we change our binary classification task to a multi-class classification method.

4.2.7.2 Comparing with Ren et al. [3]

To have another comparison with the related work, specially in relation classification task, we compare the results of our best model with Ren et al. [3], as published in their relation extraction results on the Wiki-KBP dataset [1, 2] and is compared with other related work. Their results, as published in [3], are shown in Table 4.14, which shows a comparison of their relation classification method with others, tested on Wiki-KBP dataset. The last row of the table shows our results.

4.2.7.3 Discussion over the results

Looking at table 4.14, the accuracy of our system is 6% higher than that of Ren et al. [3]. Our system has two main strong points. First, the classifier is developed using the family-tree network architecture which trains feature sets separately in the first three layers and then combines them. This characteristic helps the network to understand the influence of each feature set on the target more accurately. Second, we use BabbleLabel labeling functions as one of the features of our classifier. BabbleLabel is a powerful framework for labeling sentences and it could achieve 20% higher F1-score than a traditional Distant Supervision method [27].

To better understand the source of the improvements, we conducted some further experiments. First, we implement a MLP network with 8 layers (the same with the network

TABLE 4.15: Accuracy for Wiki-KBP dataset [1, 2] with different methods.

Network	Features	Accuracy
simpleMLP	FstT+FGR	0.581
simpleMLP	FstT+FGR+Bbl	0.629
family-tree	FstT+FGR	0.653
family-tree	FstT+FGR+Bbl	**0.731**

shown in Figure 3.5) and the same parameters as the network used for Table 4.12. This network does not use the family-tree architecture and its input is the concatenation of all feature sets. We called this model as simpleMLP. Second, we reduce the features into two feature sets and remove BabbleLabel features to see its impact on the result. The results are shown in Table 4.15.

According to Table 4.15, adding BabbleLabel features in simpleMLP and using the family-tree network increase the accuracy around 5% and 10% respectively.

4.3 Reproducibility

In this section, we provide some details of the system to make it easy for others to replicate our results. The set of the trigger words and the network hyper-parameters are required for replicating the results of the experiments. For the ClueWeb data, hand-written trigger words (without using training sentences) are used and they are shown in Section 4.1.1.3. For other datasets, trigger words are extracted automatically from the training set.

The reported hyper-parameters and the number of neurons for all modes are selected via a random search over 30 configurations on the same held-out development set. Tables 4.16 shows the selected hyper-parameters for each relation extraction task.

TABLE 4.16: Hyper-parameters of our network fed by all types of features (same notation as Figure 3.5). L1, L2 and L3 are the number of neurons for hidden layer of each set of inputs. The numbers are shown for the best model (BabbleLabel+FastText+FIGER) trained on 300 epochs.

Dataset	Relation	batch size	learning rate	activation function	L1	L2	L3	n_hidden
ClueWeb dataset	Spouse	20	0.005	sigmoid	50	60	20	20
	Parent	20	0.0005	ReLU	60	100	20	20
	Book-author	60	0.0005	sigmoid	50	60	20	60
	location-contain	60	0.005	ReLU	50	60	20	60
	place-of-birth	20	0.0005	sigmoid	50	50	100	20
BabbleLable Spouse dataset	Spouse	20	0.005	sigmoid	50	60	20	20
Wiki-KBP dataset	all relations	60	0.0005	sigmoid	50	60	20	60

Chapter 5

Conclusions

In this thesis we discussed and evaluated a method for producing labeled data for relation extraction with accompanying explanations that is based on focusing on specific trigger words, indicative of a relation in text. We achieved that by training a specific type of neural network on three effective feature sets; sentence embedding, fine grained entity types and labeling functions of synthetic natural language explanations generated using small number of positive and negative trigger words for each relation. Trigger words can be obtained from training data automatically or picked by users.

We collected ClueWeb data annotated with Freebase entity identifiers to obtain our desired training data, and produced simple annotations by inspecting positive examples containing the trigger words. ClueWeb data is also used for obtaining negative examples by focusing on sentences from different relations accepting the same entity types. We considered four entity types for experimenting on, person-person, person-location, person-art and location-location.

Our method has two main parts; feature extraction and relation classification. Desired features are extracted using accurate tools of sentence embedding, named entity recognition and semantic labeling functions. The classification part is a multiple-layer perceptron network named family-tree network which is trained in two steps. First, it is trained on each of the feature sets independently and second it is trained on the concatenation of the outputs of the first part.

We used the BabbleLabel [27] system to process the produced synthetic explanations and to obtain executable feature extractors. We experimented with the proposed neural

relation prediction architecture using multiple combinations of features and tried to find the best model. Our experiments can be divided into two phases. First, we did some experiment to find the best tools for feature extraction part (e.g., best tool for sentence embedding). Second, we tried to find out which combination of the features (produced by best tools) gives a best performance.

The system is evaluated on three datasets; BabbleLabel Spouse dataset [5], data collected from ClueWeb [6] and Wiki-KBP dataset [1, 2]. Hand-picked trigger words are used for the first two dataset experiments. However, to show that the proposed system can be completely automatic and can be applied on every new dataset, for the last dataset, all of the trigger words derived from the training sentences automatically.

The results support our hypothesis that high accuracy can be achieved by focusing on positive and negative trigger words to derive general albeit useful explanations. Our results show 6% higher accuracy than Ren et al. [3] on the same test and train dataset. By this experiment, we show that using labeling functions as the input of a family-tree based network can improve the system performance. Moreover, our system gets better accuracy than Hancock et al. [27] on their dataset, which can be explained by the differences in the way we use their labeling functions and the different classifier we apply. We have another evaluation where we investigate the False-Positive samples for each relation to find out why the system miss-classified them. As we expected, most of those sentences can be considered as noisy samples of the data.

We can identify a few avenues for future work, addressing the limitations of our work. First and foremost, our efforts are just the first attempt at validating the BabbleLabel approach, and by no means exhaustive. A systematic and large-scale validation is needed, including a comparison to other baselines and using more benchmarking data. Second, since we derived the synthetic explanations using a simple rule for our sentences, we need to evaluate different strategies for automatically deriving explanations (more complicated explanations) from the trigger words, and evaluate their impact on the effectiveness of the extractor. Third, we need to make our system more general. Currently, our system works on the data with defined subject and object entities. We need to generalize it for classifying completely raw sentences into relations. Finally, we have to add an ability to our system to make it an explainable system for evaluating

the exact impact of each feature on the result. This ability can pave the way for feature selection in relation extraction tasks.

Bibliography

[1] Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100, 2012.

[2] Joe Ellis, Xuansong Li, Kira Griffitt, Stephanie M Strassel, and Jonathan Wright. Linguistic resources for 2013 knowledge base population evaluations. In *TAC*, 2012.

[3] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conferences Steering Committee, 2017.

[4] Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[5] Braden Hancock. Babble labble. https://github.com/HazyResearch/babble, 2018.

[6] The lemur project. http://lemurproject.org/.

[7] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

[8] AI Magazine. The ai behind watson—the technical article. *AI Magazine*, 2010.

[9] Christoph Alt, Marc Hübner, and Leonhard Hennig. Improving relation extraction by pre-trained language representations. *arXiv preprint arXiv:1906.03088*, 2019.

[10] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[11] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 1148–1158. Association for Computational Linguistics, 2011.

[12] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*, 2017.

[13] Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics, 2005.

[14] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94. ACM, 2000. ISBN 1-58113-231-X.

[15] Danushka Tarupathi Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *Proceedings of the 19th international conference on World wide web*, pages 151–160. ACM, 2010.

[16] Ce Zhang. Deepdive: a data management system for automatic knowledge base construction. *University of Wisconsin-Madison, Madison, Wisconsin*, 2015.

[17] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics, 2011.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[19] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[20] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. Accurate fact harvesting from natural language text in wikipedia with lector. In *Proceedings of the 19th International Workshop on Web and Databases*, page 9. ACM, 2016.

[21] Mike Mintz, Steven Bills, Rion Snow, and Dan 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*, pages 1003–1011. Association for Computational Linguistics, 2009.

[22] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics, 2006.

[23] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[24] Sergey Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer, 1998.

[25] Ellen Riloff, Rosie Jones, et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479, 1999.

[26] Benjamin Rozenfeld and Ronen Feldman. Self-supervised relation extraction from the web. *Knowledge and Information Systems*, 17(1):17–33, 2008.

[27] Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. Training classifiers with natural language explanations. *arXiv preprint arXiv:1805.03818*, 2018.

[28] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[29] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[30] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[31] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Neurocomputing: Foundations of research. chapter learning representations by backpropagating errors, pages 696–699, 1988.

[33] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[34] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[35] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[37] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[38] Piotr Bojanowski, Edouard Grave, Armand Joulina, and Tomas Mikolov. Facebook research. https://research.fb.com/fasttext/.

[39] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[40] Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[41] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[42] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, 2009.

[43] Abdul Ahad, Ahsan Fayyaz, and Tariq Mehmood. Speech recognition using multilayer perceptron. In *IEEE Students Conference, ISCON'02. Proceedings.*, volume 1, pages 103–109. IEEE, 2002.

[44] Alexandre Patry and Philippe Langlais. Prediction of words in statistical machine translation using a multilayer perceptron. *Proceedings of the twelfth Machine Translation Summit (MT Summit XII), Ottawa, ON, Canada*, pages 101–111, 2009.

[45] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015.

[46] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[47] DS Tarasov. Natural language generation, paraphrasing and summarization of user reviews with recurrent neural networks. In *Materials of international conference" Dialog*, 2015.

[48] Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1271–1280, 2017.

[49] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

[50] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject. org/clueweb09/FACC1/Cited by*, 5, 2013.

[51] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[52] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[53] Matthew R Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding models.