Neural Sequential Transfer Learning for Relation Extraction

vorgelegt von
M. Sc.
Christoph Benedikt Alt
ORCID: 0000-0002-0500-8250

an der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Klaus Obermayer
Gutachter: Prof. Dr.-Ing. Sebastian Möller
Gutachter: Prof. Dr. Hans Uszkoreit
Gutachter: Prof. Dr.-Ing. Alan Akbik

Tag der wissenschaftlichen Aussprache: 30. November 2020

Berlin 2021

Declaration of Authorship

I, Christoph Benedikt ALT, declare that this thesis titled, "Neural Sequential Transfer Learning for Relation Extraction" 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.

Date

Signature

Abstract

Relation extraction (RE) is concerned with developing methods and models that automatically detect and retrieve relational information from unstructured data. It is crucial to information extraction (IE) applications that aim to leverage the vast amount of knowledge contained in unstructured natural language text, for example, in web pages, online news, and social media; and simultaneously require the powerful and clean semantics of structured databases instead of searching, querying, and analyzing unstructured text directly. In practical applications, however, relation extraction is often characterized by limited availability of labeled data, due to the cost of annotation or scarcity of domain-specific resources. In such scenarios it is difficult to create models that perform well on the task. It therefore is desired to develop methods that learn more efficiently from limited labeled data and also exhibit better overall relation extraction performance, especially in domains with complex relational structure.

In this thesis, I propose to use transfer learning to address this problem, i.e., to reuse knowledge from related tasks to improve models, in particular, their performance and efficiency to learn from limited labeled data. I show how sequential transfer learning, specifically unsupervised language model pre-training, can improve performance and sample efficiency in supervised and distantly supervised relation extraction. In the light of improved modeling abilities, I observe that better understanding neural network-based relation extraction methods is crucial to gain insights that further improve their performance. I therefore present an approach to uncover the linguistic features of the input that neural RE models encode and use for relation prediction. I further complement this with a semi-automated analysis approach focused on model errors, datasets, and annotations. It effectively highlights controversial examples in the data for manual evaluation and allows to specify error hypotheses that can be verified automatically. Together, the researched approaches allow us to build better performing, more sample efficient relation extraction models, and advance our understanding despite their complexity. Further, it facilitates more comprehensive analyses of model errors and datasets in the future.

Kurzfassung

Relationsextraktion (RE) befasst sich mit der Entwicklung von Methoden, die relationale Informationen in unstrukturierten Daten automatisch erkennen und abrufen können. Sie ist von entscheidender Bedeutung für Anwendungen der Informationsextraktion (IE), die darauf abzielen große Mengen an Wissen in unstrukturiertem natürlichsprachigem Text, z.B. in Webseiten und sozialen Medien, zu nutzen und gleichzeitig die leistungsfähige und klare Semantik strukturierter Datenbanken benötigen; statt unstrukturierten Text direkt zu durchsuchen, abzufragen und zu analysieren. In der Praxis ist die Anwendung von RE jedoch problematisch: Annotationskosten und Knappheit domänenspezifischer Ressourcen resultieren oft in einer begrenzten Verfügbarkeit von überwachten Daten. In solchen Szenarien ist es schwierig Modelle zu erstellen, die diese Aufgabe effektiv lösen können. Daher ist es wünschenswert Methoden zu entwickeln, die effizienter aus wenigen überwachten Daten lernen und eine bessere RE Gesamtperformanz aufweisen, besonderes in Domänen mit komplexer relationaler Struktur.

In dieser Dissertation schlage ich vor hierfür Transferlernen zu verwenden, d.h. erlerntes Wissen aus verwandten Aufgaben wiederzuverwenden um Modelle zu verbessern, speziell ihre Performanz und Effizienz aus wenigen überwachten Daten zu lernen. Ich zeige wie sequentielles Transferlernen, insbesondere unüberwachtes Sprachmodel-Vortraining, die Leistung und Dateneffizienz überwachter und distanzüberwachter RE verbessern kann. Angesichts der verbesserten Modellierungsfähigkeiten ist ein besseres Verständnis der auf neuronalen Netzen basierenden RE Methoden entscheidend um neue Erkenntnisse zu gewinnen, die ihre Leistung weiter verbessern. Hierzu stelle ich einen Ansatz vor um linguistische Merkmale der Eingabetexte aufzudecken, die von Modellen kodiert und für die Relationsvorhersage verwendet werden. Des Weiteren ergänze ich dies durch einen halbautomatischen Analyseansatz, der sich auf Modellfehler, Datensätze und Annotationen konzentriert. Zusammen erlauben es die erforschten Ansätze, leistungsfähigere und effizientere RE Modelle zu erstellen und unser Verständnis trotz ihrer Komplexität zu verbessern. Darüber hinaus erleichtert es in Zukunft umfassendere Analysen von Modellfehlern und Datensätzen.

Acknowledgements

This thesis would not have been possible without the encouragement, assistance, and support of many people. First, I would like to express my deepest gratitude to my supervisor Sebastian Möller for his unwavering support and for giving me the freedom to pursue my interests and follow my curiosity. I would also like to extend my deepest appreciation to Hans Uszkoreit for his profound belief in my abilities, which allowed me to conduct this research in the first place; his helpful advice; and for kindly agreeing to review this thesis. Also, I would like to extend my sincere thanks to Alan Akbik for reviewing this thesis and his constructive feedback and insightful suggestions.

I want to thank Charlene who has been my confidante and 'motivational coach' for her limitless support. I am also grateful for the support of my family and friends. Special thanks go to my parents Susanne and Karl-Heinrich.

I also want to thank my colleagues at DFKI. In particular, I am extremely grateful to Leonhard Hennig for his unparalleled support and guidance, and to my fellow research collaborators Marc Hübner, Aleksandra Gabryszak, Robert Schwarzenberg, David Harbecke, and Roland Roller for their invaluable contributions, inspiring discussions, and exchange of ideas. Thanks also to Arne Binder, Nils Rethmeier, Sven Schmeier, and Philippe Thomas for their helpful advice and practical suggestions over the years.

To my parents

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
GCN	**G**raph **C**onvolutional **N**eural **N**etwork
IE	**I**nformation **E**xtraction
KBP	**K**nowledge **B**ase **P**opulation
KB	**K**nowledge **B**ase
LM	**L**anguage **M**odel
LSTM	**L**ong-**S**hort **T**erm **M**emory
MLE	**M**aximum **L**ikelihood **E**stimation
MLP	**M**ulti **L**ayer **P**erceptron
ML	**M**achine **L**earning
NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork
OpenIE	**Open I**nformation **E**xtraction
POS	**P**art(s) **O**f **S**peech
RE	**R**elation **E**xtraction
RNN	**R**ecurrent **N**eural **N**etwork
TAC	**T**ext **A**nalysis **C**onference

Chapter 1

Introduction

1.1 Motivation

We live in a digital world full of data, and it is expanding at astonishing rates. As access to the internet increases, with a global internet population of over 4.5 billion people in 2019[1], natural language remains the primary way to exchange information – and thus knowledge. Each day, a tremendous amount of text data is generated in the form of web pages, online news, blog posts, tweets, texts, and emails: Each minute in 2019, users sent 188 million emails, wrote 18.1 million texts, published 510,000 tweets and 92,000 blog posts on Tumblr, and queried Google 4.5 million times[2]. In 2019, authors edited the English Wikipedia approximately 12 million times and created more than 200,000 new articles, now totaling 6 million[3]. Although these examples represent only a small fraction of data being created, they clearly show an increasing trend in available text data. Gantz and Reinsel (2012) expect this trend to accelerate even further and estimate that in 2020 a third of the data in the digital universe, including unstructured text, will be of value, *but only if it can be tagged and analyzed*. Tagging such quantities of natural language data, i.e., enriching it with structured information, is only possible with automated information extraction from text.

To illustrate the importance of information extraction from text and its challenges, let us assume our company manufactures a complex technical product, e.g., a car or a power plant. Producing

[1] https://www.internetworldstats.com/stats.htm
[2] https://www.domo.com/learn/data-never-sleeps-7
[3] https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

it involves thousands, sometimes even tens of thousands, of different parts and technologies. To ensure that each part is available in time, our buying department expects an up-to-date list of suppliers and possible alternatives. It, however, is infeasible to manually research, compile, and continuously update a database of all possible suppliers for so many different parts. In our fast paced world this information also changes rapidly, e.g., new companies enter the market while others shift their focus or go out of business. We thus want to automatically collect this information about companies of interest, e.g., their name, where they are located, and the parts and technologies they produce. The more accurate and up-to-date this information, the more cost savings can be realized.

In addition to the usefulness of automated information extraction from text, the example highlights its challenges. The information we desire is often readily available in web pages, in online news, and on social media; continuously monitoring these sources and automatically extracting the relevant details would satisfy our information need. The information, however, is only available in un- or semi-structured text, which, if untagged, can not be readily analyzed for our purpose. In other words, we are unable to search, filter, or sort unstructured natural language data for the specific information we seek, such as, companies, where they are located, what kind of technologies they produce, etc.

Relation extraction The objective of relation extraction is to automatically detect such relational information in unstructured text data, which is why it plays such an important role in the effort to extract structured information from text. For example, in our scenario we aim to find companies and where they are located. Thus, we first identify entities of interest mentioned in text, in this case companies and locations. Given the entities, we must detect whether the context allows the conclusion that a company is located in a particular location. In other words, we want to find instances of a LocatedIn relation between entities of type *company* and *location*. Figure 1.1 shows three examples with marked entities in question, e.g., companies ("Intel", "Texas Instruments") and locations ("Santa Clara", "Dallas"). The first two examples contain a relation instance LocatedIn(Intel, Santa Clara). In the context of the third one, however, "Intel" and "Dallas" do not express a LocatedIn relation, whereas "Texas Instruments" and "Dallas" do.

Crucially, the relation may be expressed in many different ways in text. The goal in relation extraction is to develop methods that reliably and effectively detect a relation of interest, no matter the wording or syntax. Relation instances extracted from text are structured information. Depending on the use case, they may be subsequently input to another system or model, or

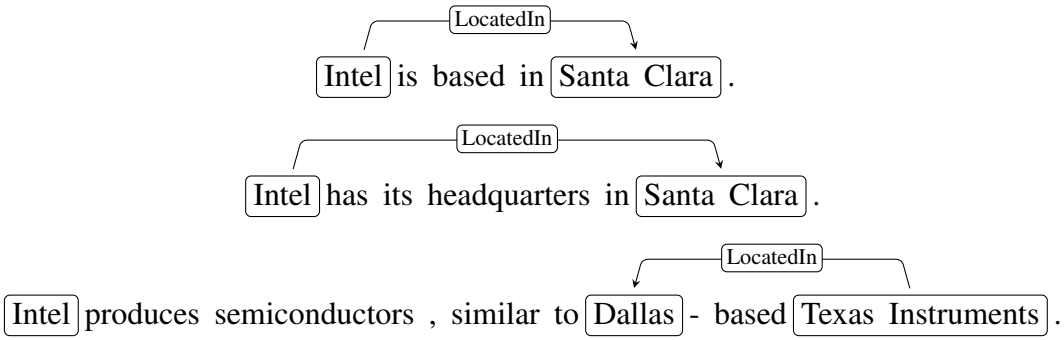


Figure 1.1 Examples for a LocatedIn relation. Entities of type *company* and *location* are marked. A link between two entities indicates that a relation is supported by the sentential context they appear in. Typically, links are directional, connecting head and tail entities, also called arguments.

indexed into a structured database where they can be analyzed, i.e., searched, sorted, or filtered for the desired attributes.

The literature shows two established categories of relation extraction methods:

Rule-based methods Rule-based methods use handcrafted extraction rules, or templates, similar to regular expressions (Yangarber and Grishman, 1998). The methods do not require annotated data and also allow to easily inspect the system's state, which is mandatory in many industrial applications. They, however, suffer from scalability issues when the set of rules grows large and becomes more complex, making it impossible to be managed effectively. Most research thus focuses on machine learning-based methods.

Machine learning-based methods Machine learning-based methods use mathematical models built from available task-specific data (Skounakis et al., 2003; Zeng et al., 2014). These methods exhibits superior scalability and performance on complex relations but require large quantities of supervised data, in particular neural network-based approaches (Smirnova and Cudré-Mauroux, 2018). A lot of research therefore focuses on addressing this issue, for example, via distant supervision (Mintz et al., 2009) or bootstrapping (Uszkoreit, 2011).

Applications that rely on relation extraction critically depend on the quality and accuracy of the extracted relation instances; making relation extraction performance most crucial to their success. In practical scenarios, both categories outlined above suffer from a number of limitations. As previously observed (e.g., Akbik, 2016), the overall problem is one of cost. Pattern-based methods require experts familiar with linguistics, machine learning, and natural

language processing (NLP) to manually create precise but often complex rules and templates. On the other hand, machine learning-based methods require large quantities of supervised data. In practice, however, only a limited number of labeled examples will be available due to the cost of manual annotation by human experts or crowd workers. A model is then created from scratch, given only the task-specific supervised data. This may result in sub-par performance, in particular for complex relations, as the number of examples is insufficient to reliably model robust syntactic and semantic patterns of a relation that generalize well to unseen data.

Transfer learning promises to mitigate this issue by transferring useful knowledge from related domains, tasks, and languages to the target setting (Howard and Ruder, 2018; Peters et al., 2018).

In this thesis, I argue that current relation extraction methods, specifically neural network-based approaches, are too data-intensive and exhibit insufficient performance on complex relations when trained solely on task-specific supervised data – which is typically limited. To overcome this problem, I develop methods based on transfer learning that reuse previously acquired knowledge for the relation extraction task. I demonstrate that the proposed methods outperform existing approaches under different types of supervision; often without additional linguistic features that state-of-the-art approaches rely on. To improve our understanding of complex neural network-based models, I develop a method to analyze which linguistic features of the input they capture and use for relation prediction, and further propose an approach to comprehensively analyze model misclassifications.

1.2 Research Objectives

This thesis studies the problem of transferring learned knowledge to the task of relation extraction. The main hypothesis of the thesis is the following:

Deep neural network methods for relation extraction that leverage existing knowledge from related tasks or domains outperform models not using this information.

The first objective is to develop methods that learn more efficiently from the same amount of supervised data, i.e., methods that are more data- or sample-efficient. I place the following desiderata on the researched methods:

- The acquisition of helpful (linguistic) knowledge should be independent of task-specific in-domain data. Instead, it should be acquired in advance, without supervision, on large and easy accessible text corpora.
- To further reduce the dependence on supervised data, it is desired that the method can be combined with other sources of supervision, e.g., distant supervision.

The second objective is to develop approaches that improve our understanding of neural relation extraction methods – including those based on transfer learning – with the goal to provide further insights and to identify areas for future research. I place the following requirements on the researched methods:

- Understanding neural network-based relation extraction methods should consider multiple aspects: model, dataset, annotations, and evaluation. It is desired to provide insights into a model's internal workings, e.g., properties of the input relevant to the decision making process, as well as external factors, such as linguistic phenomena (of the data) or incorrect annotations that result in model errors.
- Relation extraction methods, domains, and corpora may differ greatly, depending on the use case. It is desired that an approach is applicable independently of the aforementioned aspects.

1.3 Main Contributions

To address the requirements stated above, I propose a sequential transfer learning method for relation extraction. It uses language model pre-training to first acquire (linguistic) knowledge about the language from a large collection of text data. The knowledge is then transferred when creating a relation extraction model from the task-specific supervised data. Further, I propose two methods to improve our understanding of trained neural network-based relation extraction models. In more detail, the main contributions of this thesis are:

1. Sequential transfer learning for relation extraction I develop a sequential transfer learning method for supervised relation extraction to increase data efficiency and thus performance in settings with limited labeled data. The method is based on unsupervised pre-training

of language representations (Section 2.4.3.1). A model is first trained on a large collection of text to acquire knowledge about the language. In a second step, the knowledge is transferred to the relation extraction task by further training the model on supervised in-domain data. This approach improves training efficiency, as the model already acquired reliable syntactic and semantic representations of the language. An extensive experimental evaluation on two popular supervised relation extraction benchmarks shows superior performance and data efficiency compared to earlier methods. In a second step, I use the resulting state-of-the-art transfer learning method and extend it to distantly supervised relation extraction, i.e., to build a model from noisily supervised data (Section 2.3.2.2). A comprehensive automated and manual evaluation demonstrates superior performance at higher recall levels and the models ability to predict a more diverse set of relations compared to previous approaches.

This research has resulted in two full paper publications:

- Improving Relation Extraction by Pre-Trained Language Representations. *Christoph Alt*[§], *Marc Hübner*[§] *and Leonhard Hennig.* 1st Conference on Automated Knowledge Base Construction, AKBC 2019.

  The authors contributed as follows: I proposed the idea of using transformer-based language model pre-training for relation extraction, conceived, implemented, and evaluated the performance, entity masking, and ablation experiments on both datasets. Marc Hübner implemented and evaluated the data efficiency experiments on TACRED. Leonhard Hennig supervised the work. The manuscript was written by all authors.

- Fine-tuning Pre-Trained Transformer Language Models to Distantly Supervised Relation Extraction. *Christoph Alt, Marc Hübner and Leonhard Hennig.* 57th Annual Meeting of the Association for Computational Linguistics, ACL 2019.

  The authors contributed as follows: I proposed the idea, implemented the model and selective attention mechanism, and also conceived and evaluated the held-out performance experiments. Marc Hübner implemented the evaluation and visualization of precision-recall curve. Leonhard Hennig supervised the work, coordinated the manual evaluation efforts, and we jointly analyzed the results. The manuscript was written by all authors.

2. Analyzing captured linguistic knowledge Analyses of the previously proposed transfer learning methods strongly demand a better understanding of neural network-based models

[§]Equal contribution.

and their decision process. The goal is to provide insights into model internals, identify new directions of research, and ultimately improve performance. To reveal linguistic features used by models for relation extraction, I develop an approach based on probing tasks, or diagnostic classifiers (Section 2.5.1). Each task targets a linguistic property of the input, e.g., the type of the head entity, and estimates how well this information is encoded in a model's internal representation – indicating how important it is to the model's prediction. In particular, I introduce 14 probing tasks that specifically target linguistic properties relevant to relation extraction. An extensive evaluation on two benchmark datasets and more than 40 different models finds that the bias induced by the neural network architecture and the inclusion of linguistic knowledge are clearly expressed in the probing task performance. To facilitate future research and development of probing tasks, I introduce two modular and extensible software libraries: *RelEx*[5], a comprehensive suite of state-of-the-art neural relation extraction methods; and *REval*[6], a framework to develop and evaluate probing tasks.

This research has resulted in a full paper publication:

- Probing Linguistic Features of Sentence-Level Representations in Neural Relation Extraction. *Christoph Alt, Aleksandra Gabryszak and Leonhard Hennig.* 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020.

  The authors contributed as follows: I proposed the idea to use probing tasks tailored to relation extraction and implemented all tasks except TreeDepth and SDPTreeDepth. I also conceived and evaluated the experiments. Leonhard Hennig supervised the work and implemented the TreeDepth and SDPTreeDepth tasks. The manuscript was written by all authors.

3. Fine-grained analysis of model errors and datasets Analyses by means of probing tasks focus on aspects of the input that models consider relevant for their prediction, but neglect circumstances where models fail – and why they fail. To enable analyses with a focus on model errors, datasets and annotations, I develop a semi-automated approach that selects examples for manual evaluation based on aggregated evidence from multiple models, followed by grouping of examples and counterfactual rewriting (Section 2.5.2) to formulate and verify error hypotheses. To demonstrate its effectiveness, I conduct an extensive evaluation of TACRED (Zhang et al., 2017), one of the largest and most widely used crowdsourced relation extraction datasets. Specifically, the goal is to understand where current state-of-the-art relation extraction methods

[5] https://github.com/dfki-nlp/RelEx
[6] https://github.com/dfki-nlp/REval

fail. In particular, I investigate how crowd annotations and linguistic phenomena in the dataset contribute to the errors. An extensive manual and automated evaluation shows that labeling errors and ambiguous relations account for a large fraction of errors. I further release a revised TACRED to improve the accuracy and reliability of future relation extraction model evaluations.[7]

This research has resulted in a full paper publication:

- TACRED Revisited: A Thorough Evaluation of the TACRED Relation Extraction Task. *Christoph Alt, Aleksandra Gabryszak and Leonhard Hennig.* 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020.

  The authors contributed as follows: I proposed the idea to thoroughly evaluate TACRED, implemented, created, and evaluated the 49 models to aggregate evidence for selecting examples for manual evaluation. Further, I conceived, implemented, and evaluated the automated analysis. Aleksandra Gabryszak coordinated the manual annotation efforts, analyzed the label errors, and created the misclassification categories. Leonhard Hennig supervised the work. The manuscript was written by all authors.

Together, the three contributions address the desiderata outlined above, and allow us to build better performing and more data-efficient relation extraction models. Similarly, they improve our understanding of complex relation extraction models, and allow for a more fine-grained and effective analysis of model errors, datasets, and annotations in the future.

1.4 Thesis Outline

This thesis is structured as follows: In Chapter 2, I provide an overview of background information that is necessary to understand the content of this thesis. I review fundamentals in machine learning, neural network-based methods, and natural language processing. I further discuss the main aspects of this thesis: relation extraction, transfer learning, and the analysis of methods in neural language processing.

I then present the three main contributions in the order presented above.

[7] https://github.com/dfki-nlp/tacrev

In Chapter 3, I present my work on sequential transfer learning for supervised relation extraction to increase performance and data efficiency in settings with limited supervised data (Alt et al., 2019b). Chapter 4 presents my work on extending the transfer learning method proposed in the previous chapter to distantly supervised relation extraction (Alt et al., 2019a). In Chapter 5, I introduce an approach based on probing tasks, or diagnostic classifiers, to reveal linguistic features of the input used by neural network-based models for relation extraction (Alt et al., 2020a). Chapter 6 then presents my research on analyzing relation extraction methods with a focus on model errors, datasets, and annotations; for which I develop a semi-automated approach based on evidence aggregation, grouping of examples, and counterfactual rewriting to formulate and verify error hypotheses (Alt et al., 2020b).

Chapter 7 completes this thesis with a conclusion, where I summarize my findings and provide an outlook into future directions of research.

Chapter 2

Background

This chapter provides background knowledge for the subsequent chapters. It introduces the fundamentals of machine learning and a particular type of machine learning model that will be used throughout the thesis, neural networks (Section 2.1). It then gives an overview over common tasks in natural language processing (NLP) (Section 2.2).

I subsequently review the main NLP task of this thesis: relation extraction, which is concerned with extracting semantic relations from text (Section 2.3). In the following, I discuss transfer learning methods that allow to transfer knowledge between tasks and domains (Section 2.4). Finally, I give an overview of methods to analyze trained models in neural language processing (Section 2.5).

2.1 Machine Learning

In machine learning, we aim to build mathematical models from data. In the typical setting we have an input that is represented as a vector $\mathbf{x} \in \mathbb{R}^d$ of d features. A feature is a property, or an attribute, of the data. An example is a single instance, or observation, represented by a *feature vector*, and it is assumed to be drawn independently from a distribution that generates the data $\hat{p}_{data}$. A dataset is a collection of examples, which can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of examples in the dataset.

In the supervised learning scenario, each example $\mathbf{x}_i$ is typically associated with a target, or label, y_i. In contrast, in unsupervised learning no labels are available. There exist two common

types of supervised learning: classification and regression. In classification the label belongs to a pre-defined set of categories, or classes; in regression the target is a continuous number. Classification can be further divided into three settings: binary classification, multi-class classification and multi-label classification. In binary classification, there are only two classes; in multi-class classification we deal with more than two classes; and in multi-label classification one example can be associated with multiple labels.

2.1.1 Neural Networks

Neural networks are a commonly used method to model various tasks in computer vision and natural language processing, which led to remarkable results in recent years. In this section, I give an overview over the building blocks of neural networks. In general, neural networks can be seen as a composition of functions. For example, *logistic regression* is one of the simplest forms of neural network:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{W}\mathbf{x} + \mathbf{b}\,, \\ a(\mathbf{z}) &= softmax(\mathbf{z})\,, \\ softmax(\mathbf{z})_i &= \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\,, \end{aligned} \tag{2.1}$$

where $\mathbf{W} \in \mathbb{R}^{C\times d}$, $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{b} \in \mathbb{R}^C$, $\mathbf{z} \in \mathbb{R}^C$, C is the number of classes, and d is the dimensionality of the input. Logistic regression is a composition $a(f(\cdot))$ of two functions f and a, where $f(\cdot)$ is an *affine function* and $a(\cdot)$ is a so called *activation function*. A neural network is a composition of multiple of these functions, with intermediate non-linear activation functions. The sigmoid (Equation 2.2) and softmax activation are commonly used at the final layer, or *output layer*, to obtain a Bernoulli or categorical distribution.

$$sigmoid(z) = \frac{1}{1+e^{z}} \tag{2.2}$$

Non-output layers are called *hidden layers*. Neural networks are typically named according to the number of hidden-layers. For example, a network with one hidden layer is commonly known as *feed-forward neural network*, or multilayer perceptron (MLP):

$$\begin{aligned} \mathbf{h}_1 &= a_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)\,, \\ \mathbf{y} &= softmax(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)\,, \end{aligned} \tag{2.3}$$

where a_1 is the activation function of the first hidden layer. Each layer l is parameterized by its own weight matrix $\mathbf{W}_l$ and bias vector $\mathbf{b}_l$. $\mathbf{h}_l$ is commonly referred to as the hidden state of the neural network at layer l.

Activation function Besides the softmax and sigmoid functions, which are mainly used at output layers, a common activation function for hidden layers is the *rectified linear unit* (ReLU):

$$ReLU(z) = max(0,\ z)\,. \tag{2.4}$$

2.1.1.1 Layers and Models

I will now describe the neural network layers and models that are typically applied to the natural language processing (NLP) tasks discussed in the next section, and which will be used frequently throughout this thesis.

Word embeddings When applying neural networks to NLP tasks, each word w_i in the dictionary, or vocabulary, V is typically mapped to a vector $\mathbf{x}_i$, which is known as the word embedding of w_i. The word embeddings are stored in an embedding matrix $E \in \mathbb{R}^{|V| \times d}$. A text, which consists of a sequence of words $w_i, \ldots, w_T$, is first represented by a sequence of word embeddings $\mathbf{x}_1, \ldots, \mathbf{x}_T$ before it is input to a neural network.

Recurrent neural network To model tasks for natural language, typically a sequence of words, we require models that are able to process sequential input. The recurrent neural network (RNN) (Elman, 1990) is one of the elementary building blocks for processing sequential input. An RNN can be seen as a feed-forward network unrolled over a number of steps. At each step t the network is applied to the input $\mathbf{x}_t$ and the hidden state $\mathbf{h}_{t-1}$ from the previous step. $\mathbf{h}$ can be seen as a "memory" of the previous steps in the sequence. Specifically, at every step the RNN computes the following operation:

$$\begin{aligned} \mathbf{h}_t &= a_h(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h)\,, \\ \mathbf{y}_t &= a_o(\mathbf{W}_o\mathbf{h}_t + \mathbf{b}_o)\,, \end{aligned} \tag{2.5}$$

where $a_h(\cdot)$ and $a_o(\cdot)$ are activation functions. The RNN, however, has difficulties to learn over long sequences, due to the problem of *vanishing* or *exploding gradients*. During the

forward pass, the hidden state is multiplied by the weight matrix at every step. The gradients are multiplied with the same values at each step during the backward pass, or back-propagation, which can cause the gradients to become very large, i.e., explode, or become very small, i.e., vanish. In both cases the model is unable to learn. Mitigating this issue is one of the motivations behind the development of long-short term memory networks.

Long-short term memory Long-short term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) are preferred when modeling sequential data as they are able to retain information for long time spans. This is beneficial for long range dependencies that are common in natural language. To achieve this, the LSTM extends the standard RNN with a mechanism to decide what information should be retained and what should be forgotten. In contrast to the RNN the LSTM contains a forget gate $\mathbf{f}_t$, input gate $\mathbf{i}_t$, and output gate $\mathbf{o}_t$, which are all functions of the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. The gates then interact with the previous cell state $\mathbf{c}_{t-1}$, the current input $\mathbf{x}_t$, and the current cell state $\mathbf{c_t}$ to selectively retain and overwrite information:

$$\begin{aligned}
\mathbf{i}_t &= a_g(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)\,, \\
\mathbf{f}_t &= a_g(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)\,, \\
\mathbf{i}_t &= a_g(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)\,, \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ a_c(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)\,, \\
\mathbf{h}_t &= \mathbf{o}_t \circ a_h(\mathbf{c}_t)\,,
\end{aligned} \tag{2.6}$$

where $a_c(\cdot)$ and $a_h(\cdot)$ are tanh activation functions, $a_g(\cdot)$ is a sigmoid activation, and $\circ$ is an element-wise multiplication.

Convolutional neural network Convolutional neural networks (CNN) (LeCun et al., 1998) are another commonly used type of neural network. In particular, we are interested in its application to natural language tasks (Kim, 2014).

Typically, a convolutional layer slides filters of different window sizes over the concatenated input word embeddings $[\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T \times d}$, where d is the dimension of the word embedding. Each filter with weight $\mathbf{W_c} \in \mathbb{R}^{kd}$ computes a new feature $c_i \in \mathbb{R}$ for each window of k words $[\mathbf{x}_i, \ldots, \mathbf{x}_{i+k-1}] \in \mathbb{R}^{kd}$ according to the following equation:

$$c_i = a(\mathbf{W}[\mathbf{x}_i, \ldots, \mathbf{x}_{i+k-1}] + b)\,, \tag{2.7}$$

where b is the bias, and $a(\cdot)$ is an activation function. Sliding a single filter over every window of size k produces a feature map $\mathbf{c} \in \mathbb{R}^{T-k+1}$:

$$\mathbf{c} = [c_1, \ldots, c_{T-k+1}] \tag{2.8}$$

Finally, pooling is applied to reduce a feature map to its most important feature:

$$\tilde{c} = pool(\mathbf{c}), \tag{2.9}$$

where pooling can be the *min*, *mean*, or *max* operator. The pooled values of all feature maps are then combine into a vector $\tilde{\mathbf{c}} \in \mathbb{R}^C$, where C is the number of filters. The vector is then passed on to the next layer or an output layer.

Transformer The transformer (Vaswani et al., 2017) is a more recent neural network architecture motivated by the desire to replace the inherently sequential computation of RNNs with a more parallelizable approach based on (self-)attention (Bahdanau et al., 2014). Many state-of-the-art methods for language modeling and transfer learning use this architecture, e.g., the OpenAI GPT (Radford et al., 2018) and BERT (Devlin et al., 2019). Its main building block

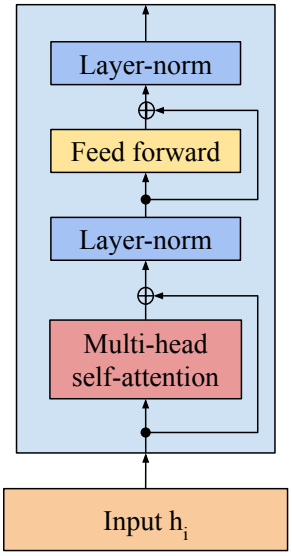


Figure 2.1 A transformer layer.

is the transformer layer, shown in Figure 2.1, which consists of two sub-layers: multi-head self-attention and a position-wise feed-forward neural network. A residual connection (He et al., 2016) is employed around each of the two sub-layers, followed by layer normalization (Ba et al., 2016). The main component is an attention function, which can be described as mapping

a query vector $\mathbf{q}$ and a set of key-value vector pairs ($\mathbf{k}_i$, $\mathbf{v}_i$) to an output. The output is computed as the weighted sum over value vectors, where the weight is obtained by a compatibility, or scoring, function between the query vector and the key vectors. We typically compute the attention on multiple queries at once, for example on a sequence of word embeddings $\mathbf{x}_i, \ldots, \mathbf{x}_T$. Queries, keys, and values therefore are concatenated into matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. Each query is then compared to all keys to compute an attention distribution over values, which is used to weight them as follows:

$$attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{2.10}$$

where d_k is the dimensionality of the queries and keys. For multi-head attention H different linear projections are first applied to queries, keys, and values before the attention is computed, where H denotes the number of heads. Intuitively, in each of the H subspaces the similarity between queries and keys has a different meaning, e.g., syntactic similarity or distance in the input sequence. Multi-head attention is computed as follows:

$$\begin{aligned} &multi\_head\_attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = ([head_1, \ldots, head_H])\mathbf{W}^Q, \\ &head_i = attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{Q}\mathbf{W}_i^V), \end{aligned} \tag{2.11}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_v}$, and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_m}$. d_m is the dimensionality defined for the model.

2.2 Natural Language Processing

Natural language processing (NLP) aims to teach computers to process and understand natural language. This is typically framed as a series of annotation tasks, i.e., mapping a text to linguistic structures that represent its meaning. We use machine learning, as presented in the previous section, to learn a model of this mapping. Our aim is to train a model that can map from an input text, a sequence of words[1] $w_1, \ldots, w_T$ represented by their numerical representation $\mathbf{x}_1, \ldots, \mathbf{x}_T$, to an output y.

I will now provide an overview of the tasks that are discussed throughout this thesis, except relation extraction which I review in detail in the next section (Section 2.3). Table 2.1 shows an example sentence and annotations for each task. Part-of-speech tagging, named entity

[1] While I use the notion of words for brevity, the examples also hold for other units, e.g., subwords or characters.

recognition, and dependency parsing are sequence labeling tasks that assign an output y_i to each word w_i.

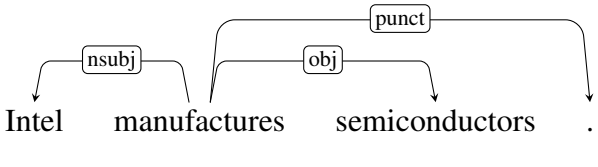
Task	Annotation			
Subwords	Intel	manufactures	semi ##conductor ##s	.
Words	Intel	manufactures	semiconductors	.
Part-of-speech tagging	NNP	VBZ	NNS	PUNCT
Named entity recognition	B-ORG	O	B-TECH	O
Dependency parsing				
Language modeling	Intel	manufactures	semiconductors	[?]
Masked language modeling	Intel	[MASK]	semiconductors	.

Table 2.1 Annotations for natural language processing tasks. NLP is typically framed as a series of annotation tasks modeled by a machine learning algorithm, e.g., a neural network. The input can be of different granularity, e.g., words, characters, or subwords (in the example '##' indicates subwords that do not start a word).

Part-of-speech (POS) tagging POS tagging assigns each word in a text its corresponding part-of-speech tag. A part-of-speech is a category of words that have similar grammatical properties, for example, noun, verb, adjective, adverb, pronoun, preposition, conjunction, etc. A word can also function as more than one part-of-speech when used in different context. The tag set used by the Penn Treebank (Marcus et al., 1993) is the most common and consists of 36 tags.[2] The tags, however, can be arbitrarily fine-grained and may also differ between languages. Multiple efforts exist to develop a "universal" tag set. For example, Petrov et al. (2012) proposed a set of 12 coarse-grained POS tags, and the current Universal Dependencies 2.0 tag set (Nivre et al., 2016) contains 17 tags.[3]

Named entity recognition (NER) NER detects and assigns types or categories to *entities* in a text, such as PERSON or LOCATION. Typically, the types are pre-defined and depend on the application they are used in. In our introductory example tags include ORGANIZATION

[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[3] http://universaldependencies.org/u/pos/

and TECHNOLOGY. Common types include locations, organizations, person names, and time expressions. Named entity recognition therefore is a common component of many information extraction systems. There exist various annotation schemes for the task. The most common notation is *BIO*, which indicates the begin, inside, and outside words of an entity span, e.g., B-ORG, as shown in Table 2.1.

Dependency parsing Dependency parsing determines the dependency structure, or *dependency parse*, of a sentence. The parse consists of words that are connected by directed links, representing the grammatical structure of the sentence. Each link connects a head word to its dependent (the child), which modifies the head according to the syntactic relationship indicated by the link. Dependency parsing is used in many applications, such as coreference resolution, question answering, and relation extraction.

Language modeling Language modeling aims to predict the next word, given its history of words, for example, in a sentence. It is typically considered an un- or self-supervised learning problem, as it only requires access to the raw text, and it is a fundamental part of recent advances in transfer learning for NLP, in particular, pre-training (Section 2.4.3.1). Language modeling is applied in many applications, such as intelligent keyboards and spelling autocorrection. More recently, masked language modeling evolved as a task. Instead of predicting the next word, the task involves first masking one or multiple words with the goal to predict them based on the remaining context. This is similar to a cloze style test, which aims to "fill in the blanks."

2.3 Relation Extraction

Relation extraction is a common task in NLP, like named entity recognition, and it is concerned with extracting semantic relations from text. It is a crucial part of information extraction from text, which aims to transform unstructured text into a structured representation so that it can be analyzed or further processed by subsequent applications.

In this section, I first introduce information extraction to ground relation extraction in its most common application (Section 2.3.1). I then review existing relation extraction methods, which can be divided into four categories depending on the degree of supervision (Section 2.3.2). Most recent methods are based on neural network models, which I subsequently describe

(Section 2.3.3). Finally, I give an overview over the datasets and metrics that I use to evaluate relation extraction methods in this thesis (Section 2.3.4).

2.3.1 Information Extraction

Let us return to the introductory example: collecting information for our supply chain application from large quantities of text. For example, we wish to collect facts about companies, e.g., where they are located and the technologies they produce. Information extraction aims to automatically extract the desired information from unstructured text data so it can be analyzed or subsequently used in higher-level NLP tasks, such as question answering (Xu et al., 2016a) or knowledge base population (Ji and Grishman, 2011).

Information extraction systems typically process the text input in multiple steps, similar to a pipeline, as shown in Figure 2.2. Each step adds increasingly abstract annotations to the input text so as to successively create a more abstract representation of it. The initial pre-processing

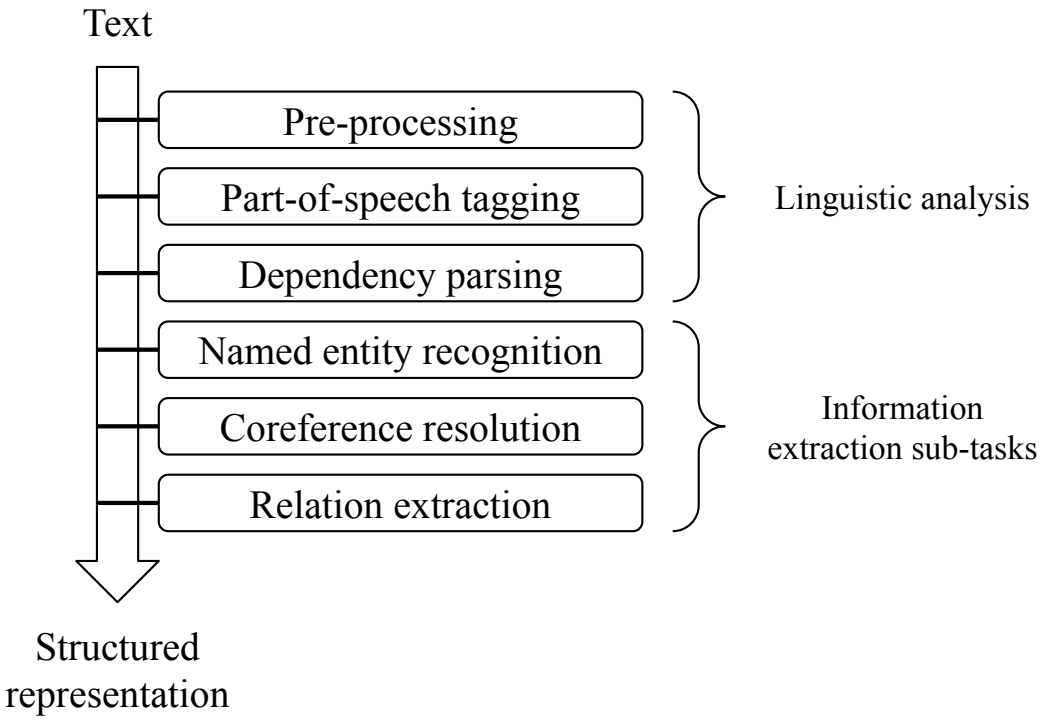


Figure 2.2 A typical information extraction pipeline. Each step adds increasingly abstract annotations to the input text, successively creating a more abstract representation of it.

step segments the text into sentences and tokenizes each sentence into a sequence of words, or

other linguistic units, such as sub-words (Sennrich et al., 2016). Subsequent linguistic analysis steps include part-of-speech tagging and dependency parsing. After the linguistic analysis, named entity recognition detects entity types of interest mentioned in the text, followed by coreference resolution to find all expressions that refer to the same entity. In some cases, a mentioned entity can be ambiguous, i.e., it refers to multiple entities. For example, "Paris" may refer to the capital or to the name, e.g., "Paris Hilton", depending on the context. Entity linking aims to resolve the mentions to unique entries in a structured knowledge base. At this point only the structure between entities is missing. The objective of relation extraction is to establish this structure by detecting relationships between the mentioned entities or concepts.

In most applications, entity types and relations are pre-defined (Piskorski and Yangarber, 2013), which is assumed to be the case throughout this thesis. There, however, exist scenarios without pre-defined entities types and relations, as we will see shortly. In case of our supply chain example, we first define entities of interest, such as *ORGANIZATION*, *LOCATION*, and *TECHNOLOGY*. Relations may include LocatedIn and ProducesTech. The first relation holds between an *ORGANIZATION* and a *LOCATION* and between two *LOCATIONS*; the latter holds between an *ORGANIZATION* and a *TECHNOLOGY*.

Let us take a look at the example sentence: "Intel, headquartered in Santa Clara, California, manufactures semiconductors.", shown in Figure 2.3. The linguistic analysis and named entity recog-

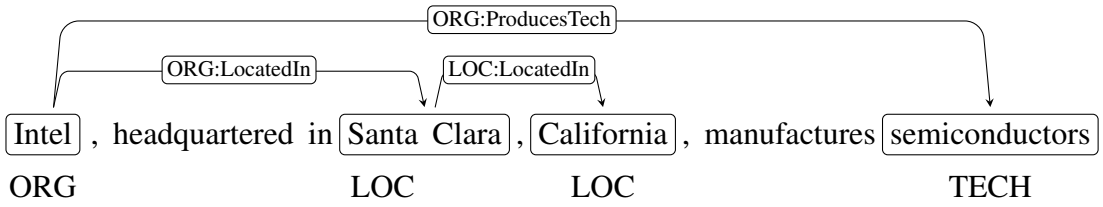


Figure 2.3 Relation extraction example. The goal of relation extraction is to detect semantic relationships between entities of different types, e.g., organizations (*ORG*), locations (*LOC*), and technologies (*TECH*).

nition identified "Intel" as an *ORGANIZATION*, "Santa Clara" and "California" as a *LOCATION*, and "semiconductors" as a *TECHNOLOGY*. This is equivalent to the setting of this thesis, which assumes that linguistic analysis and information extraction sub-tasks already completed and their results are available. I therefore focus on the task of relation extraction. The goal is to develop a system that is able to detect relationships in text, if there is textual evidence. In our example, a system is expected to identify the following relations: LocatedIn(Intel, Santa Clara), LocatedIn(Santa Clara, California), and ProducesTech(Intel, semiconductors). This, however, is often challenging due to the complexity and ambiguity of natural language.

In principle, relations can have an arbitrary number of arguments, which is commonly known as *n-ary relation extraction* (Akbik and Löser, 2012; Ling and Weld, 2010). For example, an Acquisition relation may comprise a buyer, seller, and price. In this thesis, I focus on methods for extracting relations between two arguments, i.e., *binary relation extraction*. In general, these methods can be readily applied to the n-ary setting as well.

2.3.2 Methods

Depending on the use case or domain, relation extraction systems can be built with a variety of methods. Typically, one must consider the availability of training data and structured knowledge sources, and the cost involved with manual annotation of examples. In this section, I provide an overview of relation extraction methods categorized into four scenarios, in order of difficulty of obtaining the training data:

- **Traditional supervision**: The traditional supervised setting requires manually labeling each training example.
- **Distant supervision**: Distant supervision (Mintz et al., 2009) uses heuristics to automatically obtain large quantities of noisily supervised data.
- **Semi supervision**: Semi-supervised methods uses a small amount of manually labeled data as a seed to iteratively expand the supervised data.
- **No supervision**: No supervision is the simplest setting, as it only requires raw data such as unlabeled text.

2.3.2.1 Supervised Relation Extraction

In the traditional supervised setting, we assume that sufficient labeled data is available which can be used to create a relation extractor, e.g., by training a machine learning-based relation extraction model. Figure 2.4 shows two annotated binary relation extraction examples. Each example consists of text with two of the mentioned entities marked, e.g., “Intel” and “motherboard chipsets”, and an associated label indicating the relation, such as ProducesTech. Labels are typically assigned by trained annotators or crowd workers.

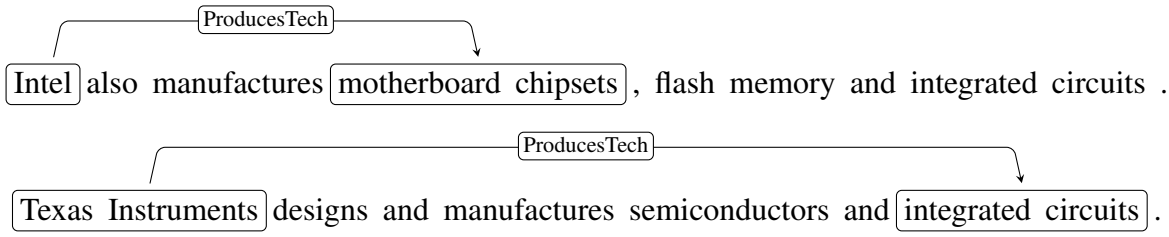


Figure 2.4 Supervised relation extraction examples. Both examples contain a ProducesTech relation between an *organization* and a *technology*.

Early work in supervised relation extraction mainly focuses on two categories of methods depending on how the input text and mentioned entities are represented: *Pattern-based methods* use explicit rules or templates; *feature-based methods*, on the other hand, represent the input as a feature vector, i.e., a set of manually defined attributes of the input.

Pattern-based methods Early pattern-based methods use hand-built extraction rules, similar to regular expressions (Yangarber and Grishman, 1998). An input is processed in subsequent steps via pattern matching on logical forms extracted from the input over previous steps. Typical steps include lexical analysis, named entity recognition, and parsing. Figure 2.5 shows an example for a logical form, which can be seen as an object with named slots.

Slot	Value
Class	Manufacturer
Company	Intel
City	Santa Clara
State	California
...	...

Figure 2.5 Example of a logical form for the sentence "Intel, headquartered in Santa Clara, California, manufactures semiconductors."

Later methods (Kim et al., 2011; Xu et al., 2007) use sentence templates, also called patterns or rules, learned from supervised data. The templates include surface-level word representations, dependency parse grammar, and entity types. After obtaining the templates, they can be used to extract relations from text via explicit pattern matching. An advantage of pattern-based methods is that it allows humans to handcraft rules and easily inspect the systems state, which is often required in industrial applications.

Feature-based methods In comparison to pattern-based relation extraction methods, feature-based methods represent the input text and mentioned entities as a set of attributes, or features, that can be extracted from the input. These features include:

Lexical features: Lexical features are in widespread use since the earliest work in relation extraction (Grishman and Sterling, 1993). These "shallow" features include properties of words, e.g., their surface form, lemma, and stem (Zhou et al., 2005). Another example is the use of trigger words, i.e., words that indicate a relation (Hendrickx et al., 2010). For example, the word "manufacturer" may indicate the presence of a Produces[...] relation. In addition to trigger words, a number of works use words before the first entity, after the second entity, and between mentions as a feature since they often relate the entities (Mooney and Bunescu, 2006). One of the most widely used features besides lexical ones are part-of-speech tags. POS tagging is a form of syntactic analysis that determines the syntactic category of a word, e.g., noun or verb. For instance, a phrase that consists of a preposition and a verb, e.g., "headquartered in", between entity mentions may indicate a LocatedIn relation.

Dependency parse features: Lexical features, however, are unable to capture long range dependencies or semantic relationships between constituents in a sentence. A number of works use dependency parsing to determine the dependency structure of a sentence. A frequently used feature is the shortest dependency path (SDP) connecting the two entity mentions (Mintz et al., 2009; Zhou et al., 2005). Considering the SDP is beneficial to relation extraction, because it focuses on the actions and agents in a sentence (Bunescu and Mooney, 2005b; Socher et al., 2014).

Entity mention features: Unsurprisingly, features related to the entity mentions are one of the most important. For example, Mintz et al. (2009); Zhou et al. (2005) used the named entity type of each mention as a feature. This is often helpful, because it constraints the set of relations to be considered, e.g., the LocatedIn only holds between organizations and locations. Other features are words surrounding the entities, and whether an entity is a nominal, name, or pronoun.

For each example, the desired features are extracted and stored in a feature vector, which is combined with the corresponding relation label to train a supervised machine learning model, e.g., a support-vector machine (Bunescu and Mooney, 2005a; Grishman et al., 2005; Zelenko et al., 2003), logistic regression/maximum-entropy model (Chieu and Ng, 2002; Kambhatla,

2004; Sun et al., 2011), or probabilistic graphical model (Ray and Craven, 2001; Rosario and Hearst, 2004; Skounakis et al., 2003).

In traditional feature-based methods, features are hand engineered and defined a priori. Recent deep learning methods aim to automatically learn useful input features for the task. These methods use neural network-based models to learn latent features, e.g., representing the input in a continuous vector space, on top of which a classifier is applied to predict the relation. We will see shortly how recent work uses neural networks of different architecture to extract relations from text. Despite the recent advances it is still common to explicitly provide neural network-based models with features, such as part-of-speech tags, named entities types, morphological features, and hypernyms to improve their performance.

2.3.2.2 Distantly Supervised Relation Extraction

Distant supervision was first proposed by Mintz et al. (2009) to train relation extraction models, where any sentence that contains a relation instance, i.e., a pair of entities that participates in a known relation, was assumed to be an example of that relation. Known relations are taken from existing knowledge bases (KB), which contain facts in the form of tuples, for example, PlaceLived(Jonathan Lethem, Brooklyn). In this setting the supervised data necessary for training can be created automatically without time-consuming and expensive manual annotation by human experts. For instance, in Figure 2.6 the tuple (Jonathan Lethem, Brooklyn) is used to assign any sentence that contains both entities the relation label PlaceLived. This is also commonly known as the *distant supervision assumption*. After obtaining the noisily supervised data, we can create a relation extraction model with the same methods and features as in the supervised setting. For example, Mintz et al. (2009) trained a multi-class logistic regression for relation classification.

A drawback of heuristically assigning labels is that it introduces noisy labels. For instance, under the distant supervision assumption we consider all three sentences in Figure 2.6 to be examples of the relation PlaceLived. Only one of them, however, truly expresses the relation (green check mark); the other two sentences do not (red check marks). In practice this often leads to poor relation extraction performance, because it is often impossible to accurately model the task based on inaccurate labels. Riedel et al. (2010) find that the distant supervision assumption is frequently violated and proposed *multi-instance learning* to train a model despite the noisy labels. This setting assumes that at least one sentence that contains a pair of entities which participates in a know relation is also an example of that relation. The authors use a

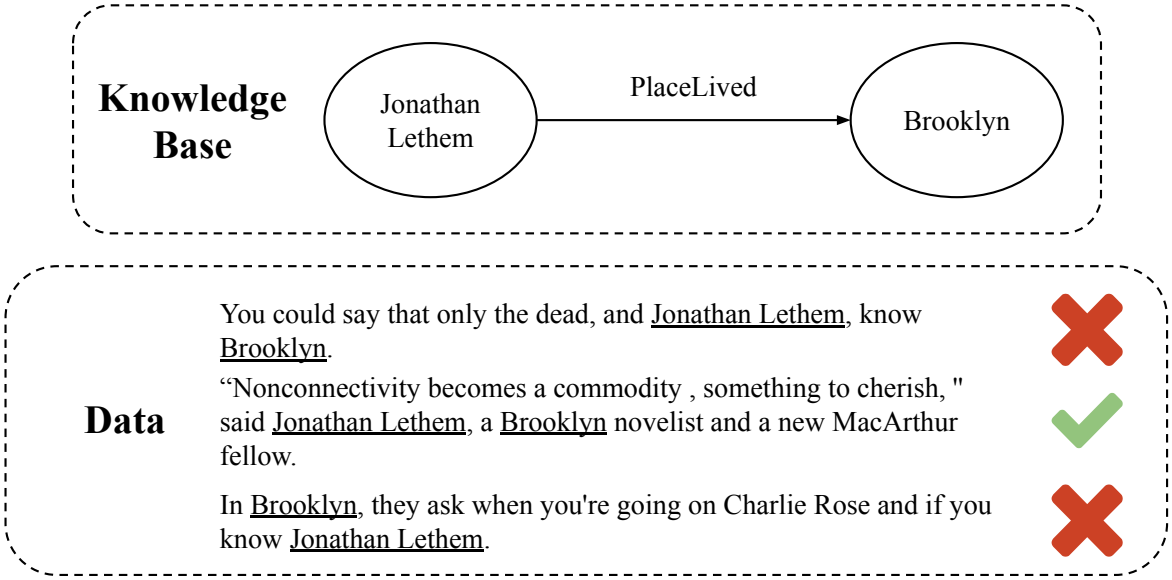


Figure 2.6 Distant supervision example. The underlying assumption is that all sentences that mention a known fact, e.g., Jonathan Lethem lived in Brooklyn, also express this relation. However, this assumption is often violated and thus introduces label noise, e.g. only the second sentence truly expresses the PlaceLived relation.

probabilistic graphical model to predict relations between entities and which sentence truly expresses the relation, which leads to more accurate results. Hoffmann et al. (2011) and Surdeanu et al. (2012) observe that previous multi-instance models assume that relations are disjoint, although entity pairs often participates in multiple relations, e.g., PlaceLived and PlaceOfDeath. The authors propose *multi-instance multi-label learning* which is based on the assumption that each relation mention, i.e., a pair of entities in a sentence, has exactly one label, but a pair is allowed to exhibit multiple relations across different mentions. Both works use probabilistic graphical models to model this assumption and classify relations.

Other limiting factors include: incomplete knowledge bases, i.e., knowledge bases lack facts that are true; and system's preprocessing errors, which violate the distant supervision assumption in many cases and further increase the label noise. Various works focus on mitigating this problem. For example, Angeli et al. (2014) use active learning, i.e., they integrate human feedback into the learning process; Takamatsu et al. (2012) and Roller et al. (2015) aim to automatically identify unreliable labels, e.g., via learned relation paths in the knowledge base and a generative model to assess the quality of the labeling process. Another drawback is the availability of knowledge bases that cover the relations of interest. Major sources of relations

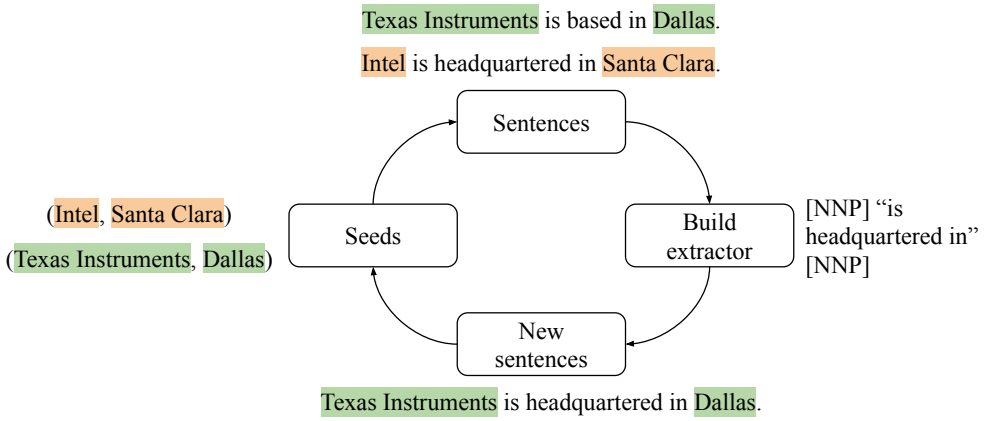


Figure 2.7 Semi-supervised relation extraction with bootstrapping example. Relation of interest is LocatedIn. Bootstrapping starts from an initial set of seed relation instances, e.g., (Intel Santa Clara), and iteratively expands it by first building a relation extractor from newly obtained sentences that mention these instances, followed by extracting new instances that can be added to the initial seeds.

are Freebase, DBPedia[4] and Wikidata,[5] which only cover a specific set of relations that may not be applicable to the domain of interest, such as healthcare.

Similar to the traditional supervised learning setting, most recent methods in distantly supervised relation extraction also use neural network-based models (Section 2.3.3).

2.3.2.3 Semi-Supervised Relation Extraction

In semi-supervised relation extraction, we aim to minimize the manual effort involved to build a model for the task (Agichtein and Gravano, 2000; Brin, 1998; Xu et al., 2007). Methods in this setting typically rely on *bootstrapping*, or bootstrap learning. In this scenario, we use a small set of relation instances, or seeds,[6] to initially collect sentences from a corpus that mention the instances. In a second step, we use the obtained sentences to build a relation extraction model, e.g., by learning a classifier (Agichtein and Gravano, 2000), or by learning extraction rules (Xu et al., 2007). Then, we apply the model to a corpus to retrieve further relation instances, which are added to the initial seed set. In the next iteration, we repeat the same procedure with the larger seed set to retrieve even more sentences, which we use to extend the seed set even further. Figure 2.7 shows a simplified bootstrapping iteration example. Here, we use the initial seed tuple (Intel, Santa Clara) to retrieve a sentence from a corpus that mentions the instance. Then

[4] https://wiki.dbpedia.org
[5] https://wikidata.org
[6] Seeds are not limited to relation instances and also include manually created patterns or annotated sentences.

we build an extractor from the sentence and use it to retrieve a new relation instance, in this case (Texas Instruments, Dallas). In the next iteration, we add the newly found relation instance to the seeds and repeat the process.

A drawback of this iterative approach is that it leads to semantic drift (Agichtein and Gravano, 2000; Xu et al., 2010). At each iteration there is a chance that we add false positive tuples to the seed instances. As we use these instances to retrieve new sentences, and the new sentences to learn a new extractor, the error propagates from iteration to iteration. To mitigate this issue, Agichtein and Gravano (2000) focus on overly specific patterns so only highly precise tuples get added to the seeds. Xu et al. (2010) instead estimate pattern confidence to decide which extracted tuples to add.

2.3.2.4 Unsupervised Relation Extraction

In unsupervised relation extraction our objective is to extract relations of a priori unknown semantic types, for example, in applications where the relations are new or unknown. A common approach is to use clustering methods on entity pairs and patterns. For instance, Shinyama and Sekine (2006) propose the paradigm of unrestricted relation discovery. They first cluster news articles and extract the mentioned entity pairs, followed by a step to construct patterns that indicate a relation. The method then uses the obtained patterns to cluster entity pairs extracted from other articles. More recent unsupervised relation extraction methods build on the latent relation hypothesis, which states that pairs of words that co-occur in similar patterns tend to have similar relations. Turney (2008) compute a pair-pattern matrix between entity pairs and distinct patterns to capture this hypothesis in a vector space model (Section 2.4.3.1). The resulting vector representations allow us to compute similarities between entity pairs based on the patterns they are observed with. Akbik et al. (2012) observe that the choice of patterns is crucial, as there often exist ambiguous or underspecified patterns. They propose a more informed feature generation strategy, e.g., to consider dependency parse information beyond the shortest path between two entities, and also to consider the distinctiveness of each pattern in a cluster to determine its ambiguity. Akbik et al. (2013) add selectional restrictions to patterns, i.e., they restrict the entity types between which a pattern is allowed to match so as to reduce ambiguous patterns.

Unsupervised relation extraction and the paradigm of open information extraction often go hand in hand. The TextRunner system (Etzioni et al., 2008; Yates et al., 2007) uses a self-supervised training approach based on a dependency parsed corpus, POS tags and identified

noun phrases to train a classifier, which is subsequently applied to extract relations from a corpus. A limitation of this system is that the semantics of a relation had to be determined in a separate step and the output was noisy. Later works address this, e.g., by adding an additional disambiguation step (Yates and Etzioni, 2009), or by using a language model to assess the quality of extractions (Downey et al., 2007). Subsequent works improved self-supervised learning, for example, by using heuristic matches between Wikipedia info box attribute values and corresponding sentences to construct training data (Wu and Weld, 2010).

Although unsupervised relation extraction methods do not require supervised data, the results are often difficult to interpret and hard to map to existing relations, schemas, or ontologies (Fader et al., 2011). This is a crucial shortcoming that prevents its use in applications such as knowledge base refinement (Smirnova and Cudré-Mauroux, 2018).

2.3.3 Neural Relation Extraction

Most recent relation extraction methods use neural networks to model the task, which led to a considerable increase in performance; mainly due to their ability to automatically learn useful input representations, and to effectively model sequential data. Figure 2.8 shows a typical architecture for neural relation extraction. As stated in the previous section, we assume linguistic analysis and named entity recognition already completed, and that we have access to their results, e.g., part-of-speech information, dependency parse, and named entities. Our goal is to model the probability of a relation $P(r \mid w_1, \ldots, w_T, \mathit{head}, \mathit{tail})$ given a sequence of words $w_1, \ldots, w_T$, e.g., a sentence, and two entity mention spans denoted *head* and *tail*. Distinguishing between head and tail is important, as it indicates the directionality of the predicted relation. Each word w_i is represented by its embedding $\mathbf{e}_i \in \mathbb{R}^c$, which we either learn during training or initialize with pre-trained embeddings (Section 2.4.3.1). It is common to use a positional embedding $\mathbf{p}_i \in \mathbb{R}^d$ to indicate the relative offset of the two mentions to other words in the input sequence, e.g., "headquartered" has an offset of -2 to "Intel", and 8 to "semiconductors". Intuitively, positional embeddings capture features related to the proximity of words and also serve as an indicator for the entity mentions in question. Additional word features are often represented as embeddings, denoted $\mathbf{f}_i \in \mathbb{R}^e$, such as part-of-speech tags or named entity types.

We then apply a neural network to the sequence of concatenated word representations $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, with $\mathbf{x}_i = [\mathbf{e}_i,\ \mathbf{p}_i,\ \mathbf{f}_i] \in \mathbb{R}^{(c+d+e)}$ to obtain a fixed size representation $\mathbf{s} \in \mathbb{R}^f$. For example, if the input is a sentence, $\mathbf{s}$ can seen as a representation of the sentence conditioned on the respective *head* and *tail* mentions. The encoder can be a neural network of arbitrary

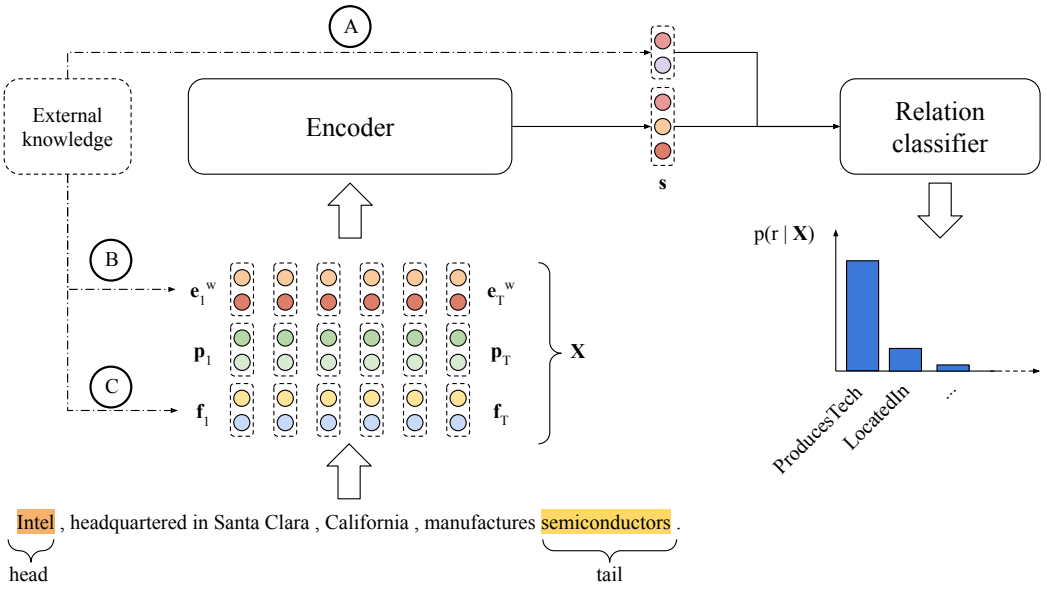


Figure 2.8 Typical architecture for neural relation extraction. We represent each word w_i in the input sentence by its corresponding word embedding $\mathbf{e}_i^w$, positional embedding $\mathbf{p}_i$, and additional word level features $\mathbf{f}_i$. A neural network-based encoder maps $\mathbf{X}$ to the sentence representation $\mathbf{s}$, which we then use to predict the relation. External knowledge can be added via sentence level features (A), distributed word representations (B), or word level features (C).

architecture, e.g., a convolutional neural network (CNN), a recurrent neural network (RNN), a graph convolutional neural network (GCN), or a self-attentive architecture. Finally, we apply a classification layer to $\mathbf{s}$, typically a linear projection followed by a softmax, to obtain the probability distribution $P(r|\mathbf{s})$ over the set of possible relations.

2.3.3.1 Supervised Neural Relation Extraction

In supervised neural relation extraction, methods mainly differ in the architecture of the encoder and the external knowledge or features they use. Socher et al. (2012) were the first to apply neural networks to the task. They use a matrix-vector recurrent neural network (MV-RNN) to encode the input token sequence according to its syntactic parse tree. Zeng et al. (2014) apply a CNN with max-pooling to pre-trained word embeddings and positional embeddings to obtain the sentence representation and combine it with lexical features of the entity mentions. Nguyen and Grishman (2015) use a similar approach but instead apply multiple convolutions of different filter sizes. Xu et al. (2015a) encode the words on the shortest dependency path and their dependency relations with a CNN into a sentence representation, similar to Xu et al. (2015b), who instead use an LSTM. Zhou et al. (2016) use an attention layer (Bahdanau

et al., 2014) to combine the final hidden states of a Bi-LSTM for each token into a sentence representation. To compute the attention weights, they use a static query vector learned during training. Zhang et al. (2017) instead proposed PA-LSTM, a position-aware LSTM, which is similar to the previous approach but uses the final state to control the attention, so the model is able to dynamically evaluate the contributions of each token after observing the entire input sequence. More recently, graph convolutional neural networks (GCN) for relation extraction garnered a lot of interest. For example, Zhang et al. (2018b) apply graph convolutions to the input token sequence. They use dependency tree information in combination with graph pruning to aggregate the token representations along the shortest dependency path. Similarly, self-attention gained a lot of interest recently. For instance, Verga et al. (2018) extend the transformer by a custom architecture for supervised biomedical named entity recognition and Relation Extraction.

2.3.3.2 Distantly Supervised Neural Relation Extraction

Methods in distantly supervised relation extraction mainly focus on multi-instance learning, multi-instance multi-label learning, and integration of additional knowledge to better learn with noisily supervised data. Neural encoder architectures are similar to the traditional supervised setting, however, methods predominantly rely on CNNs, in particular the piecewise convolutional neural network (PCNN) proposed by Zeng et al. (2015). The architecture is inspired by earlier work on feature-based methods, which observed that different parts of the context may contain important features to identify a relation. Instead of encoding the input sequence into a single representation by max-pooling, this method uses a piecewise max-pooling procedure to obtain a representation of the left context (to the left of the first entity mention), middle context (between mentions), and right context (to the right of the second mention). The representation is then combined with lexical features, such as, embeddings of entity mention tokens, their context tokens, and WordNet hypernyms, to form the final representation for relation prediction. Adel et al. (2016) further extended the PCNN. Rather than applying the separation into left, middle, and right context during pooling, the convolutional part is divided into three distinct parts.

In neural multi-instance multi-label learning, shown in Figure 2.9, we first group all examples into a bag B according to the mentioned entity pairs, e.g., Jonathan Lethem and Brooklyn. Similar to the supervised case, we use a neural network to map each example $\mathbf{X}_1, \ldots, \mathbf{X}_M$ in a bag to its representation $\mathbf{s}_i$. Under the assumption that at least one of the examples in the bag truly expresses the relation, the selection step selects the final bag-level representation $\mathbf{s}_{bag}$,

given the representations of each individual example in the bag $\mathbf{s}_1, \ldots, \mathbf{s}_M$. This representation is then input to the classification layer to obtain the probability distribution over the set of available relations.

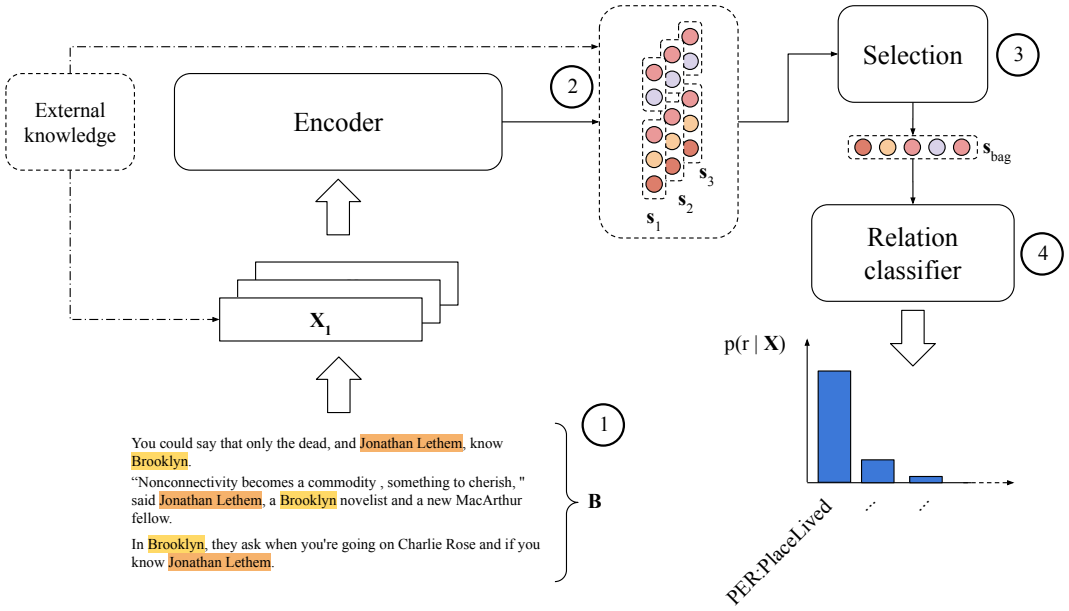


Figure 2.9 Architecture for multi-instance multi-label learning in neural distantly supervised relation extraction. First, we group all examples that mention the same entity pair, e.g., Jonathan Lethem and Brooklyn, into a bag **B**. Each example $\mathbf{X}_i$ in the bag is mapped to its representation $\mathbf{s}_i$ by an encoder (2), followed by combining the individual representations into a bag representation $\mathbf{s}_{bag}$ (3). Similar to the supervised setting, we use $\mathbf{s}_{bag}$ to predict the relation (4).

Initial multi-instance learning methods (Zeng et al., 2015) select only the most likely example in the bag. Lin et al. (2016) instead propose to use selective attention over examples, an approach that produces an attention distribution by comparing the sentence representation of each example to a learned representation of each relation. The attention distribution is then used to compute the group representation as a weighted combination of all sentence representations. The previous method treats relation extraction as multi-class classification, which does not consider that entity pairs can participate in multiple relations. Jiang et al. (2016) extend the method to multi-label classification to allow multi-instance multi-label learning.

Other methods include adversarial training (Qin et al., 2018; Wu et al., 2017), noise models (Luo et al., 2017), and soft labeling (Liu et al., 2017a; Wang et al., 2018b). Recent work showed graph convolutions (Vashishth et al., 2018) and capsule networks (Zhang et al., 2018a), previously applied to the supervised setting, to be applicable in a distantly supervised setting, too. In addition, linguistic and semantic background knowledge is helpful for the task, but the proposed

systems typically rely on explicit features, such as dependency trees, named entity types, and relation aliases (Vashishth et al., 2018).

2.3.4 Datasets and Evaluation

In this section, I first provide an overview of metrics and evaluation procedures typically used in relation extraction (Section 2.3.4.1). I then introduce the most popular benchmark datasets to evaluate supervised and distantly supervised relation extraction methods (Section 2.3.4.2).

2.3.4.1 Evaluation

Typically, we evaluate relation extraction methods in terms of *precision*, *recall*, and F_β *measure* against a test dataset, also commonly referred to as ground truth or gold standard. The dataset contains examples with all relation instances annotated that a relation extraction method is expected to identify. During evaluation, we compare the relations extracted by a model to the ground truth so we can check whether they are correct or incorrect. *True positives* (TP) are all relation instances that are retrieved and also in the set of ground truth instances. All other retrieved instances are considered *false positives* (FP).

Precision is the proportion of true positives among all retrieved instances, i.e., the overall correctness of the model. This is calculated as:

$$precision = \frac{\#TP}{\#TP + \#FP} \tag{2.12}$$

Recall is the proportion of true positives among all ground truth instances, i.e., the overall completeness of the model. This is calculated as:

$$recall = \frac{\#TP}{\#TP + \#FN} \tag{2.13}$$

There typically exists a trade-off between both metrics. For example, an application for knowledge base construction may demand that we only add relations that are extracted with high confidence. We therefore optimize our model for precision. This, however, goes at the expense of recall, as overall less relation instances are retrieved. The F_β score provides a single

metric that combines precision and recall:

$$F_\beta \; measure = \frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{2.14}$$

In relation extraction, it is common to use the harmonic mean of precision and recall ($\beta = 1$), the so-called F1 score (van Rijsbergen, 1979), which is defined as follows:

$$F1 \; score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.15}$$

The F1 score can be extended to the multi-class setting, e.g., by ignoring the class distribution (micro-averaged) or by calculating the F1-scores for each class individually and taking the mean (macro-averaged).

2.3.4.2 Datasets

Table 2.2 gives an overview over commonly used datasets in supervised and distantly supervised relation extraction. The SemEval 2010 Task 8 dataset (Hendrickx et al., 2010) is a standard

Dataset	Examples	Neg. examples (%)	Relations	Supervision
SemEval 2010 Task 8	10,717	17.4%	19	traditional
TACRED	106,264	79.5%	42	traditional
NYT-10	522,611	-	53	distant

Table 2.2 Statistics for relation extraction datasets.

benchmark for binary relation classification, and contains 8,000 sentences for training, and 2,717 for testing. Sentences are annotated with a pair of untyped nominals and one of 9 directed semantic relation types, such as Cause-Effect, Entity-Origin, as well as the undirected Other type to indicate no relation, resulting in 19 (2 x 9 + 1 other) distinct relation types in total. The official convention is to report macro-averaged F1 scores with the directionality of relations taken into account.

The *TAC R*elation *E*xtraction *D*ataset (TACRED) introduced by Zhang et al. (2017) contains 106k sentences with entity mention pairs collected from the TAC KBP evaluations[7] 2009–2014, with the years 2009 to 2012 used for training, 2013 for evaluation, and 2014 for testing.

[7] https://tac.nist.gov/2017/KBP/index.html

Sentence	Head	Tail	Relation
Mr. Scheider played the police chief of a resort town menaced by a shark.	Scheider	police chief	per:title
The measure included Aerolineas's domestic subsidiary, Austral.	Aerolineas	Austral	org:subsidiaries
Yolanda King died last May of an apparent heart attack.	Yolanda King	heart attack	per:cause_of_death
The key was in a chest.	key	chest	Content-Container
The car left the plant.	car	plant	Entity-Origin
Branches overhang the roof of this house.	roof	house	Component-Whole

Table 2.3 Examples for relation extraction datasets.

Sentences are annotated with person- and organization-oriented relation types, e.g., per:title, org:founded, and no relation for examples that do not contain a defined relation. While TACRED is approximately ten times the size of SemEval 2010 Task 8, it contains a much higher fraction of negative training examples, which makes classification more challenging. In contrast to the SemEval dataset the entity mentions are typed, with subjects classified into person and organization, and objects categorized into 16 fine-grained classes (e.g., date, location, title). The convention is to report results as micro-averaged F1 scores.

The NYT10 dataset by Riedel et al. (2010) is a standard benchmark for distantly supervised relation extraction. It was created by aligning Freebase relations with the New York Times corpus, with the years 2005–2006 reserved for training and 2007 for testing. The training data contains 522,611 sentences, 281,270 entity pairs and 18,252 relational facts. The test data contains 172,448 sentences, 96,678 entity pairs and 1,950 relational facts. There are 53 relation types, including NA if no relation holds for a given sentence and entity pair. The convention is to report *Precision@N*, i.e., precision scores for the top N extracted relation instances ranked by model confidence, and precision-recall curve. Since the test data is also created via distant supervision and can only provide an approximate measure of the performance, it is customary to also report Precision@N based on a manual evaluation.

2.3.5 Summary and Discussion

Relation extraction is a crucial sub-task of information extraction, and the shift from early pattern-based to machine learning-based methods considerably improved the quality of its results. The traditional supervised machine learning setting, however, requires large quantities of supervised data, which is costly to create. This led to the development of methods with less strict requirements on the degree of supervision, for example, distant supervision, semi-supervised learning via bootstrapping, or unsupervised relation extraction. These methods also have drawbacks, such as label noise, semantic drift, or difficult to interpret results. None of the methods is always suited and can be applied universally. At lot of research has since focused on better methods to create high quality relation extraction models, as measured in precision and recall. In particular, neural network-based methods lead to significant performance improvements.

Neural network-based methods, however, often require even larger quantities of supervised data to learn and generalize well to unseen examples. To make training more efficient, current methods try to leverage prior knowledge in the form of additional and often hand-engineered features, as well as side information. As I argue in Chapter 1, current relation extraction methods are too data-intensive and often exhibit insufficient performance on complex relations, and the current form of using prior knowledge is limited and scales poorly. To improve this, we need to: efficiently transfer knowledge from other tasks or domains that are helpful to relation extraction; and develop a better understanding what models learn, and where they still fail.

To address these limitations, I propose to use transfer learning and methods to analyze neural language processing models, which I cover in the next two sections.

2.4 Transfer Learning in Neural Language Processing

This chapter provides an overview of transfer learning. It describes the setting in which machine learning models are transferred to data outside of their training distribution. First, I contrast traditional supervised machine learning to the transfer learning setting, provide a formal definition, and introduce the four prevalent settings in transfer learning (Section 2.4.1). I then focus on those most relevant to this thesis. In particular, I review multi-task learning (Section 2.4.2) and sequential transfer learning (Section 2.4.3).

2.4.1 Transfer Learning

In the conventional supervised learning setting, if we want to learn a model A, we assume access to sufficient labeled data for the task and domain A, as shown in Figure 2.10. We can now train the model on this data an expect it to perform well on unseen data from the same task and domain. If we want to train another model B on a different task and domain B, we again need access to labeled data for this particular domain and expect it to perform well on unseen data from this particular task. The assumption in this setting is that training and unseen test

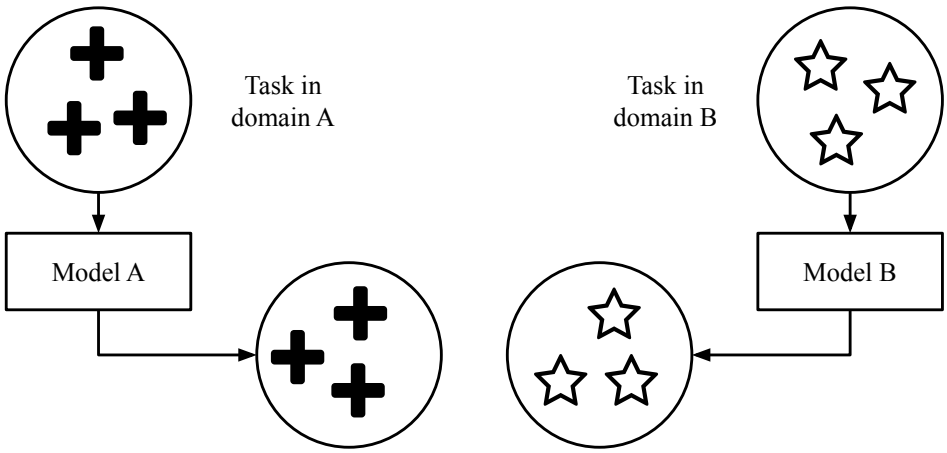


Figure 2.10 The classic supervised learning setting.

data is drawn from the same distribution and the same feature space, i.e., the data is assumed to be independent and identically distributed (i.i.d.).

This assumptions, however, fail in cases where we have insufficient labeled data for a task or domain C. Transfer learning aims to use data, or knowledge, from related tasks or domains to deal with this scenario. The related task and domain are known as *source task* and *source domain*. We first obtain the knowledge by training a model on the source task and its domain and then apply it to the *target task* and *target domain* as depicted in Figure 2.11. Depending on task and data, knowledge can be of various forms. In this thesis, knowledge is typically associated with the representations learned by neural network models (Section 2.1.1).

Following the taxonomy of Pan and Yang (2010), which was subsequently extended by Ruder (2019), transfer learning can be broadly divided into two scenarios:

- **Transductive transfer learning**: In this scenario, the source and target task are the same but supervised data is only available in the source domain. This includes *domain*

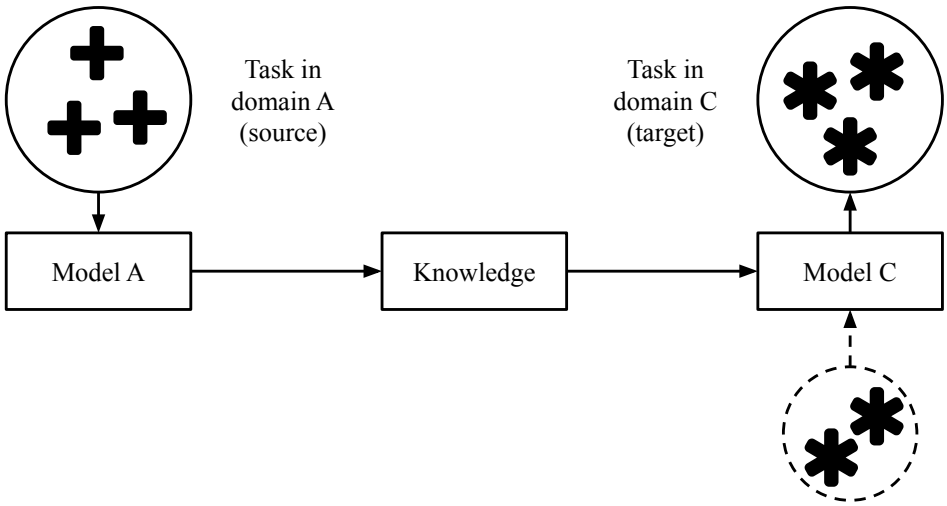


Figure 2.11 The transfer learning setting.

adaptation, where the marginal probability distributions $P(\mathbf{X}_S)$ and $P(\mathbf{X}_T)$ differ, e.g., NER for similar entity types in the legal domain (source) and in the medical domain (target) – both in the same language. The scenario also includes *cross-lingual learning*, which is characterized by differing feature spaces between source and target domain, i.e., the vocabulary differs between domains.

- **Inductive transfer learning**: In this scenario, the source and target task differ and supervised data is (only) available in the target domain. In this setting the label spaces differ between source and target tasks. This setting includes *multi-task learning* and *sequential transfer learning*, with the tasks either trained simultaneously or sequentially.

This thesis is concerned with the inductive transfer learning scenario, namely multi-task learning and sequential transfer learning, which I discuss in detail in the following sections.

2.4.2 Multi-Task Learning

The most common setting in machine learning is single-task learning, where we train a model on a single task. By focusing on just one task, however, we ignore useful knowledge from related tasks that may help us to improve the metric of interest. In multi-task learning, we address this by leveraging the knowledge contained in the training signals of related tasks (Caruana, 1997). For example, by sharing representations between tasks we can help our model to generalize better to our task of interest. Multi-task learning is also known as learning to learn,

joint learning, and learning with auxiliary tasks. Motivation for multi-task learning can be drawn from a biological or pedagogical perspective, e.g., humans first learn the necessary skills before mastering more complex tasks (Brown and Principles, 2001; Ruder, 2017). From the perspective of machine learning, multi-task learning introduces an inductive bias provided by the related tasks that forces a model to prefer some hypotheses over other hypotheses (Caruana, 1993). Specifically, it prefers those that explain more than a single task in isolation. This may result in better generalization with a fixed training set, or increased data efficiency, i.e., equal or better task performance with less supervised data. In the context of deep learning two methods are commonly used for multi-task learning with deep neural networks: *Hard* and *soft parameter sharing*.

Hard parameter sharing Hard parameter sharing is the most commonly used approach for multi-task learning in deep neural networks (Caruana, 1997). In this setting the hidden layers are typically shared between all tasks, except for some task specific output layers (Figure 2.12). Hard parameter sharing greatly reduces overfitting by forcing the model to find a representation that captures all tasks well, instead of just the task of interest, which reduces the chance of overfitting (Ruder, 2019).

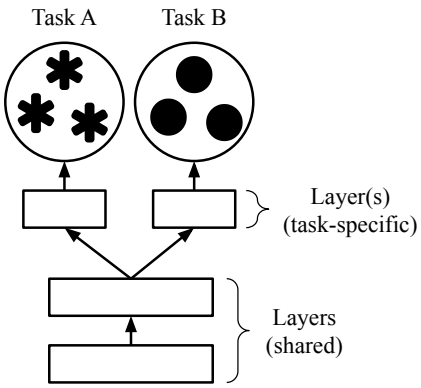


Figure 2.12 Multi-task learning with hard parameter sharing.

Soft parameter sharing In soft parameter sharing, each task has its own model with its own parameters. Instead, the difference between parameters is constrained, in order to regularize and thus encourage the parameters to be similar, as depicted in Figure 2.13. Approaches include the use of L2 norm for regularization (Duong et al., 2015), or the trace norm (Yang and Hospedales, 2017).

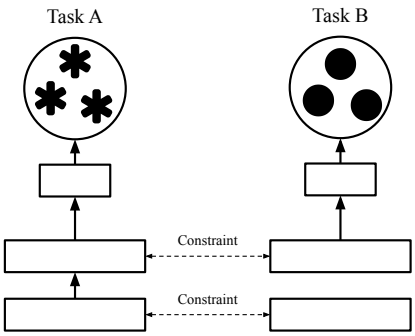


Figure 2.13 Multi-task learning with soft parameter sharing.

2.4.2.1 Auxiliary Tasks

In most multi-task learning settings, we are only interested in the performance of one particular task: the main task. For example, we may train a text classification model simultaneously with part-of-speech tagging or named entity recognition. In this case, we are mainly concerned with the text classification performance. Auxiliary tasks include: statistical tasks that predict underlying statistics of the data, e.g., predicting the log frequency of a word as an auxiliary task for sequence tagging (Plank et al., 2016); selective unsupervised auxiliary tasks that provide the model with hints about certain aspects of the task, for example, predicting whether a sentence contains a name as an auxiliary task for name error detection (Cheng et al., 2015); and supervised auxiliary tasks, e.g., tasks that are related to the main task.

2.4.3 Sequential Transfer Learning

Sequential transfer learning is the prevalent setting in transfer learning, which typically involves one source task and a target, or *downstream*, task. In this scenario the source and target task differ and training is performed in sequence (Ruder, 2019). The goal is to improve the target model by transferring knowledge from the model trained on the source task. It is mainly used in the following three settings: data for both tasks is not available simultaneously, data for the source task is available in much larger quantities than data for the target task, or repeated transfer to many target tasks is necessary.

Sequential transfer learning stages Sequential transfer learning with one source and one target task typically consists of two stages: *pre-training* and *adaptation*. In the pre-training

phase the initial model is trained on the source task, in the adaptation phase the knowledge contained in the trained model is transferred to the target task, i.e., the pre-trained model is further trained on the target task. Often the pre-training phase is expensive. However, it must be performed only once, while the fine-tuning stage is much more efficient. The source task is typically chosen to maximize the usefulness of the approach by selecting a source task that allows to learn more general representation that is useful for a wide variety of target tasks.

2.4.3.1 Pre-training

Pre-training, similar to the choice of auxiliary tasks in multi-task learning, should force a model to capture knowledge that is useful for the downstream tasks, and ideally the pre-training task provides large quantities of training data. We can discern pre-training tasks by the source of supervision: traditional supervision, distant supervision, and no supervision.

Supervised pre-training typically leverages existing supervised tasks and datasets. A common approach is to pre-train on a task related to the target task. For example, Yang et al. (2017) pre-train a model on part-of-speech tagging and adapt it to word segmentation. Another strategy is to pre-train on a task which is similar to the downstream task but has more available training data. Min et al. (2017) and Wiese et al. (2017), for instance, pre-train a question answering model on the large Stanford Question Answering Dataset (SQUAD, Rajpurkar et al., 2016) and adapt it to more specialized domains, such as biomedical QA. The focus of recent research is to learn general purpose language representations. In this case, pre-training tasks are selected which are thought of to learn something about the general structure of language, for example, by pre-training on paraphrasing (Wieting et al., 2015) or natural language inference (Conneau et al., 2017).

Distantly supervised pre-training, on the other hand, aims to provide hints, or helpful signals, to the model during training, which those solely trained on unsupervised data may be unable to exploit. The hints are similar to auxiliary tasks in multi-task learning but can be created by heuristics from raw data. For instance, Severyn and Moschitti (2015) and Felbo et al. (2017) predict emojis in tweets to pre-train a model for sentiment analysis. Nie et al. (2019) show that predicting discourse markers in sentences (e.g., “but”, “also”) is an effective pre-training signal to improve performance on downstream tasks that rely on sentence meaning, such as discourse relation prediction.

Unsupervised pre-training, also known as unsupervised transfer learning or self-taught learning, is of particular interest to this thesis. It uses raw language data to learn representations and thus is more scalable and more general than the supervised setting. The reason is that a model must encode more general information about language, its structure and meaning. Most approaches focus on learning representations of words from unsupervised data. Traditional approaches include: Latent Semantic Analysis (LSA, Deerwester et al., 1990), Latent Dirichlet Allocation (LDA, Blei et al., 2003), and Brown clustering (Brown et al., 1992) but are out of the scope of this thesis. Instead, we focus on methods that learn a dense representation of words with neural networks, in particular for pre-training word embeddings, and pre-training contextual embeddings via language modeling.

Word embeddings Representing words as dense vectors has a long history, for example, Bengio et al. (2003) use a neural network-based n-gram language model (NNLM) to learn word representations, i.e., they predict the next word in a corpus, given a context of $n-1$ words. The training procedure then minimizes the loss $\mathcal{L}_{NNLM}$, specifically, the log-likelihood under the training corpus $\mathcal{C}$:

$$\mathcal{L}_{NNLM} = -\frac{1}{|\mathcal{C}|} \sum_{i=n}^{|\mathcal{C}|-n+1} \log f(w_i|w_{i-1},\ldots,w_{i-n+1};\theta) + R(\theta)\,, \tag{2.16}$$

where $f(\cdot)$ is a neural network, e.g., a feed-forward network, θ denotes the set of model parameters, and $R(\theta)$ is a regularization term. Collobert et al. (2011) learn word representations by training a neural network to rank correct word sequences higher than incorrect ones using a max-margin loss:

$$\mathcal{L}_{MML} = \sum_{i=C}^{|\mathcal{C}|-C} \sum_{w' \in \mathcal{V}} max(0, 1 - f(w_{i-C},\ldots,w_i,\ldots,w_{i+C}) + f(w_{i-C},\ldots,w'_i,\ldots,w_{i+C})) \tag{2.17}$$

The outer sum iterates over all words in the corpus $\mathcal{C}$ and the inner sum iterates over all words in the vocabulary. Each word has a context window of c words to its left and right. Skip-gram with negative sampling (SGNS, Mikolov et al., 2013a,b) is one of the most popular methods to learn word embeddings, with the goal of training efficiency and robustness (Levy et al., 2015). Word2Vec is one of the most popular implementations of this approach. SGNS learns word representations that are good at predicting the surrounding context, given a target word w_i. The

objective is as follows:

$$\mathcal{L}_{SGNS} = \frac{1}{|C|} \sum_{i=1}^{|\mathcal{C}|} \sum_{-C<j<C, j\neq 0} \log P(w_{i+j}|w_i) \tag{2.18}$$

The probability $P(w_{i+j}|w_i)$ is computed using the softmax function:

$$P(w_{i+j}|w_i) = \frac{exp(\tilde{\mathbf{x}}_{i+j}{}^T \mathbf{x}_i)}{\sum_{t=1}^{|\mathcal{V}|} exp(\tilde{\mathbf{x}}_t{}^T \mathbf{x}_i)}, \tag{2.19}$$

where $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ are the word embedding and the context word embedding of word w_i, respectively. Global vectors (GloVe, Pennington et al., 2014) learn word representations via matrix factorization. GloVe minimizes the difference between the dot product of the embeddings $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_j$ of word w_i and its context word $\widetilde{w}_j$:

$$\mathcal{L}_{GloVe} = \sum_{i,j=1}^{|\mathcal{V}|} g(C_{ij})(\mathbf{x}_i{}^T \tilde{\mathbf{x}}_j + b_i + \tilde{b}_j - log\ C_{ij})^2, \tag{2.20}$$

where b_i and $\tilde{b}_j$ is a bias corresponding to $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_j$, and $g(\cdot)$ is a weighting function that assigns lower weight to rare and frequent word co-occurrences, as measured by the co-occurrence matrix $\mathbf{C}$.

Pre-trained language models All previous neural network-based methods are shallow: they trade off expressiveness for efficiency. This, however, limits the relations they can capture. For example, a model initialized with word embeddings must still learn from scratch to disambiguate words in context, and to infer meaning from a sequence of words. These abilities are at the heart of language understanding, which requires the ability to model complex language phenomena, such as long-term dependencies, compositionality, negation, and agreement. Methods therefore require large quantities of training data to achieve good performance.

Recent advances are based on the principle to not just initialize the first layer, i.e., the word embeddings, but to pre-train an entire model with hierarchical representations. All models discussed in the following use deep neural networks to learn representations. For instance, Dai and Le (2015) propose to pre-train an LSTM autoencoder that reconstructs the words in a sentence. Kiros et al. (2015), on the other hand, train an RNN to reconstruct the words in the preceding and succeeding sentences.

In recent years, increasing computational capabilities made it feasible to train deep neural language models (LM). The goal of neural language modeling is to estimate the joint probability $p(\mathbf{w})$ of a text, given its sequence of tokens, e.g., words or subword units, $\mathbf{w} = w_1, \ldots, w_T$. The joint probability can be decomposed as:

$$P(\mathbf{w}) = \sum_{w=1}^{T} \log p(w_i | w_{i-1}, \ldots, w_1) \,. \tag{2.21}$$

The conditional probability $P(w_i|w_{i-1}, \ldots, w_1)$ is modeled by a probability distribution over the vocabulary of tokens, given the context:

$$P(w_i|w_{i-1}, \ldots, w_1) = f(w_1, \ldots, w_{i-1}) \,, \tag{2.22}$$

where $f(\cdot)$ is a neural network model.

Although language modeling is a general concept, in practice it often refers to autoregressive language models, or unidirectional LM's. For example, Peters et al. (2017) apply a unidirectional language model to sequence labeling tasks and Ramachandran et al. (2017) utilize one for summarization and machine translation. Peters et al. (2018) first show that language model pre-training learns representations that are useful for many downstream tasks. Radford et al. (2018) show similar results but train a deeper model. Howard and Ruder (2018) found language representations learned by unsupervised language modeling to significantly improve text classification, prevent overfitting, and increase data efficiency. Devlin et al. (2019) introduce a masked language model (MLM) objective (Section 2.2), which randomly masks tokens in a text and then predicts only the masked tokens:

$$\mathcal{L}_{MLM} = \sum_{w' \in m(\mathbf{w})} \log P(w' | \mathbf{w}_{\setminus m(\mathbf{w})}) \,, \tag{2.23}$$

where $m(\mathbf{w})$ denotes the masked words, and $\mathbf{w}_{\setminus m(\mathbf{w})}$ the unmasked words. In comparison to other pre-training tasks, language modeling is more data-efficient (Zhang and Bowman, 2018).

2.4.3.2 Adaptation

The previous section focused on the first stage of the sequential transfer learning process. In the second stage, the adaptation phase, two ways are mainly used to adapt pre-trained representations to the target task: *feature extraction* and *fine-tuning*. Feature extraction provides

pre-trained representation as input features to another model. Fine-tuning, on the other hand, directly trains the pre-trained model on data of the target task.

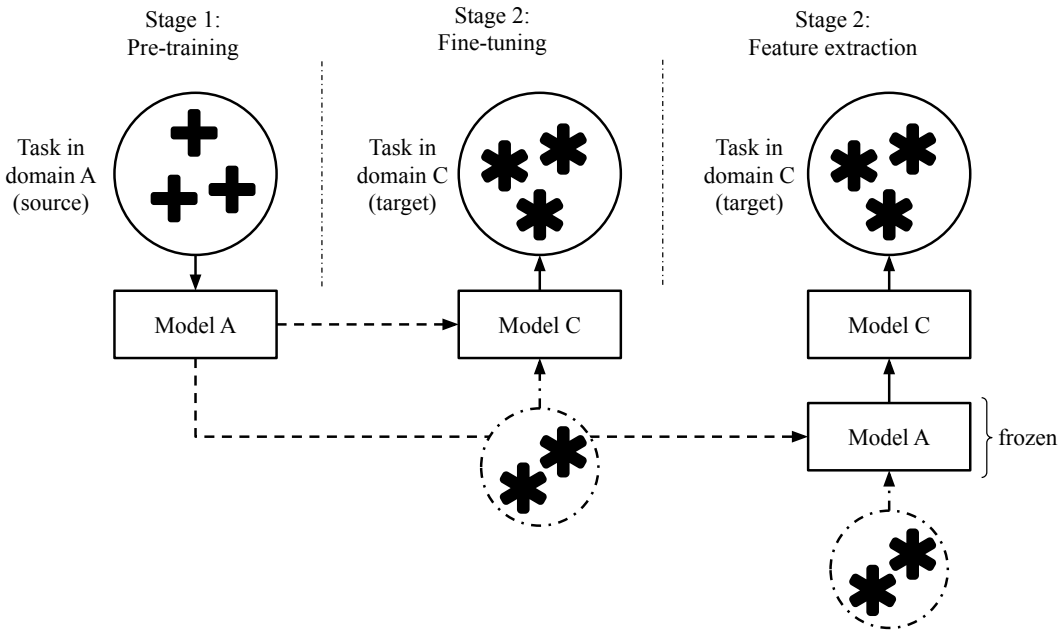


Figure 2.14 Adaptation phase in sequential transfer learning. A model is first pre-trained and then either fine-tuned (middle) or used as a feature extractor (right). The parameters of the feature extraction model are frozen, i.e., they are kept fixed throughout the adaptation phase.

Feature extraction In feature extraction, we provide the pre-trained representations as input features to a downstream model (Figure 2.14, right), similar to traditional feature-based approaches (Section 2.3.2.1). The pre-trained model parameters are frozen during training on the target task, i.e., they are left unchanged. A common use case is to provide pre-trained word representations or a combination of multiple representations as input to a model.

Fine-tuning In fine-tuning, on the other hand, we initialize the downstream model with the pre-trained model parameters (Figure 2.14, middle). In this case, the parameters are updated during training on the target task. Feature extraction is useful when we want to reuse pre-trained task specific models, or when we want to reuse the same data multiple times, either during training or for different tasks. Fine-tuning is convenient because no or minimal task-specific changes are necessary to adapt to new tasks.

2.5 Analysis Methods in Neural Language Processing

The field of natural language processing has seen impressive progress in recent years. In particular, this can be attributed to neural network-based methods replacing traditional rule- and feature-based systems; which led to improvements in various tasks, such as machine translation (Bahdanau et al., 2014), reading comprehension (Hermann et al., 2015), and syntactic parsing (Kiperwasser and Goldberg, 2016). The trend also sparked the development of various neural network architectures, which are typically more opaque than rule- and feature-based methods, as their highly non-linear structure makes them difficult to understand for humans (Samek et al., 2017). As a result, interest increased in methods that strive to better understand how NLP methods work on various tasks. This fits the larger question of interpretability, which aims to increase accountability, fairness, and trust. The goal is to better understand how models work and identify less covered aspects or scenarios where models fail; to provide insights, uncover new research directions, and ultimately improve their language abilities.

Following the taxonomy of Belinkov and Glass (2019), analysis methods can be broadly divided into the following categories:

- **Probing linguistic information:** Methods in this category identify linguistic information that is captured by neural network-based models (Section 2.5.1).
- **Adversarial examples:** Adversarial attacks create altered versions of examples that preserve their semantics with respect to the language task but change the prediction of a model (Section 2.5.2).
- **Challenge sets:** Challenge sets, or test suites, are exhaustive, systematically collected or created sets of examples that test specific aspects of a language task (Section 2.5.3).
- **Visualization:** Methods in this category visualize input feature saliency or model internals, e.g., attention distribution or hidden unit activation. These methods, however, are out of the scope of this thesis.

2.5.1 Probing Linguistic Information

Typically, we train neural network-based models in an end-to-end manner, without explicitly encoding linguistic features. To better understand a model it is helpful to identify particular

features it encodes. A common approach is to predict such properties from its hidden states, or activations, as shown in Figure 2.15. First, we train a model on a task, e.g., machine translation. To probe it for a property of interest, e.g., the tense of the input, we use a *diagnostic classifier* (Veldhoen et al., 2016), also known as *auxiliary prediction task* (Adi et al., 2017; Qian et al., 2016) or *probing task* (Conneau et al., 2018). For example, to probe the representations in layer k of our trained model, we train a second model on the probing task data, e.g., sentences and their tense, and use the model up to layer k as a feature extractor. The parameters of the model are frozen, i.e., we update only the parameters of the probing task model during backpropagation. For each probing task, a classifier is trained on a set of representations, and its

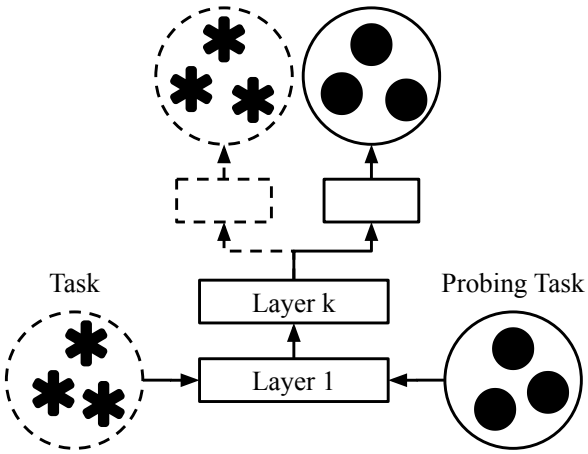


Figure 2.15 A typical probing task setup.

performance measures how well the information is encoded. Probing tasks are widely used to analyze the presence of linguistic information in a model's latent representations, for example, in machine-translation (Belinkov et al., 2017), language modeling (Giulianelli et al., 2018), and sentence encoding (Conneau et al., 2018). We typically select the probing tasks in accordance with the NLP task. For example, Conneau et al. (2018) first trained models on different tasks, such as, neural machine translation and natural language inference, and probed the sentence representations for tense, syntactic parse tree depth, subject number, etc.

Other methods try to find agreements between parts of the network and the properties of interest. Voita et al. (2018), for example, count the correspondences between attention weights and a property, e.g., for anaphora resolution. Others directly compute correlations between activations and the property of interest, for example, correlations between hidden state and syntactic parse tree depth (Qian et al., 2016).

2.5.2 Adversarial Examples

If we want to understand a model, we also need to understand its failures. Despite their success, machine learning-based models can be very brittle, i.e., they can be sensitive to malicious attacks and adversarial examples (Goodfellow et al., 2015; Szegedy et al., 2014). For example, in the vision domain small changes to an image can lead to misclassifications, even though the altered image is visually indistinguishable from the original. Adversarial examples can be generated either with access to model parameters or without. The former is commonly known as white-box attack, the latter as black-box attack (Liu et al., 2017b; Papernot et al., 2016a, 2017). White-box attacks are difficult to adapt to text, because they involve computing gradients with respect to the input, which is discrete. For example, computing the gradient with respect to the input word embedding and perturbing it may result in an embedding vector that does not correspond to an actual word, i.e., has zero likelihood in the discrete language space (Harbecke and Alt, 2020). One strategy is to search for the closest word embedding, given the dictionary (Papernot et al., 2016b) or use the gradient to compute a ranking of words to be modified in the input (Liang et al., 2018). Instead, most methods focus on black-box attacks to generate examples by changing the text input, e.g., introducing typos or misspellings (Belinkov and Bisk, 2018). The previously mentioned works focus on chracter or word level. Jia and Liang (2017) instead add sentences to the input to distract a question answering / reading comprehension model, Iyyer et al. (2018) generate paraphrases that preserve the syntactic structure. Ribeiro et al. (2018) generate semantic preserving rules that cause the model to change its prediction.

2.5.3 Challenge Sets

Typically, we train models on benchmark datasets drawn from text corpora, which exhibit a particular distribution of linguistic phenomena. One of the primary goals of challenge sets, or test suites, is to evaluate if a model is able to handle specific linguistic phenomena, or if it performs well on a wide range of phenomena. Cooper et al. (1996), for example, created a semantic test suite that covers phenomena like ellipsis, plurals, anaphora, quantifiers, etc. In recent years, in particular machine translation (MT) and natural language inference (NLI) received a lot of attention. Sennrich (2017) introduced a set of challenges to evaluate machine translation on phenomena such as noun-phrase agreement, subject-verb agreement, polarity, and so on. Burchardt et al. (2017) evaluate different machine translation engines with a test suite that covers 120 phenomena. Similarly, the GLUE NLI set (Wang et al., 2018a) covers more than

30 phenomena in four categories, such as logic, knowledge, lexical semantics, and predicate-argument structure. Challenge sets are created by handcrafting examples, either manually or programmatic. A common approach is to extract and modify examples from an existing dataset. Sanchez et al. (2018), for example, use examples from the SNLI dataset (Bowman et al., 2015) and replaced specific words, e.g., synonyms or hypernyms. Linzen et al. (2016) created a large dataset by extracting subject-verb agreement from raw text using a set of heuristics.

2.6 Conclusions

In this chapter, I provided background knowledge in machine learning and natural language processing that is necessary for the subsequent chapters. In particular, I highlighted the need for better performing and more data-efficient relation extraction methods and discussed how this can be addressed with transfer learning and analysis methods for neural language processing models. The next chapter will introduce my research on sequential transfer learning for supervised relation extraction.

Chapter 3

Sequential Transfer Learning for Supervised Relation Extraction

3.1 Introduction

As I argue in Chapter 1, current relation extraction methods are too data-intensive and exhibit insufficient performance when trained only on limited labeled data. A promising way to overcome this issue is transfer learning, i.e., to transfer knowledge previously acquired on other tasks or domains to the relation extraction task. In this chapter, I present the first part of my research on this objective: a sequential transfer learning method for supervised relation extraction.

In this section, I first give an overview of prior knowledge and supporting information used by current supervised relation extraction methods to improve training efficiency with limited labeled data, and highlight their limitations (Section 3.1.1). In the following, I discuss related work (Section 3.1.2) and propose a sequential transfer learning method based on unsupervised pre-training of language representations (Section 3.2). To study its effectiveness, I conduct an extensive evaluation in which I compare its performance and data efficiency to earlier baselines (Section 3.3). Finally, I conclude this chapter with a discussion of the proposed method and its alignment with the objective of this thesis (Section 3.4).

This chapter is mainly based on a previously published full paper (Alt et al., 2019b), but expands on the discussion of the proposed approach.

3.1.1 Problem Statement

The main challenge in relation extraction is to create a well performing system at low cost. In other words, we want to achieve the highest possible performance on a set of possibly complex relations with the least amount of supervised data. Currently, there is great interest in addressing this by incorporating prior knowledge into the relation extraction process (Lin et al., 2016; Zhang et al., 2017, 2018b). Contrary to methods that aim to increase the amount of labeled data, e.g., by heuristically labeling examples, these methods provide additional features from various sources to more effectively inform the relation extraction. Intuitively, this provides the model with a stronger signal, allowing it to model patterns that indicate a relation more effectively, despite the limited number of labeled examples. Given the stated objective of this thesis, such approaches may be highly relevant and accordingly are the starting point of my analysis.

Additional features combined with input text The most prominent approach to use prior knowledge in a model is to explicitly provide it with additional features related to the input text. Features include: prefix and morphological information about words; syntactic properties like part-of-speech tags; and semantic features, such as named entity tags, synonyms, or hypernyms. These features are typically obtained from additional, specialized systems, such as part-of-speech taggers, named entity taggers, or knowledge bases. Pre-trained word representations (Section 2.4.3.1) are another form of prior knowledge, encoding syntactic and semantic relationships among words. Representing a word's associated features, e.g., named entity types or part-of-speech tags, in vector space is also a common approach to model their semantics and compositionality.

Problems: error propagation, limited portability, and a priori feature selection Current methods that use prior knowledge do so by explicitly providing a set of features, e.g., lexical, syntactic, and semantic information, in addition to the input text. This, however, has several limitations: First, it typically requires multiple specialized systems, or feature extractors, to obtain these features, e.g., part-of-speech taggers, named entity taggers, or dependency parsers. Critically, a discrete decision is made by each extractor, which introduces a potential source of error that can propagate and accumulate if the result is used in subsequent steps, e.g., the relation extraction (Ji and Grishman, 2005). Relying on additional systems for feature extraction increases the overall complexity of the approach and restricts its portability. The reason is that each extractors must be created (Grishman, 2019), which typically requires supervised data in

the respective language or domain in addition to the task-specific data for relation extraction. Finally, the scope of features is limited and decided a priori, e.g., features that proved useful in earlier work or those commonly used in other NLP tasks. This may only cover a subset of important features and prevents the model from deciding which it considers most useful to solve the task.

Ideally, a method that addresses these limitations should consist of a single system to reduce the overall complexity; learn features without supervision that are as effective as those of earlier work to increase performance and training efficiency; be independent of discrete decisions to reduce error propagation; and instead use an implicit representation that covers a variety of (linguistic) features a model can select from during training.

3.1.2 Related Work

I first review earlier work that utilizes prior knowledge and supporting information to improve relation extraction, which I use to identify evaluation baselines (Section 3.1.2.1). I then provide an overview of approaches to transfer learning that are common in relation extraction and other NLP tasks (Section 3.1.2.2).

3.1.2.1 Use of Prior Knowledge

Discrete linguistic features Early methods in relation extraction use discrete, linguistically motivated features (Zelenko et al., 2003). For example, Rink and Harabagiu (2010) use a rich set of lexical features, including words in context, words between entity mentions, and lemmas. Hendrickx et al. (2010) utilize part-of-speech tags, dependency parse information, named entity types, morphological features, and hypernyms as additional features and demonstrated their importance to improve relation extraction performance. Although current research is mostly focused on deep learning-based methods, e.g., encoding the input text with a neural network, discrete features are still an important source of supporting information. For example, Socher et al. (2012); Zhang and Wang (2015) use discrete part-of-speech and named entity features in combination with recurrent neural networks. Kim (2014) use a similar feature set but combine it with convolutional neural networks.

Distributed representations of linguistic features Most recent methods use distributed representations of words and features to better model their semantics and compositionality. Zeng et al. (2014, 2015) combine discrete lexical features and pre-trained word embeddings with piecewise convolutional neural networks. Xu et al. (2015a,b) use shortest dependency path (SDP) information between entity mentions combined with an RNN-based model to encode only the sequence of words along the SDP instead of the whole input. Zhang et al. (2018b) propose a state-of-the-art relation extraction method that applies a combination of graph pruning and graph convolutional neural network to the dependency tree. This combination allows the model to effectively use information from words that are in close proximity based on the dependency tree, and pruning reduces the noise introduced by less important parts of the input.

3.1.2.2 Transfer Learning for NLP Tasks

Currently, the most common form of transfer learning in relation extraction is to use distributed word representations, or word embeddings (Mikolov et al., 2013b; Pennington et al., 2014). The syntactic and semantic features encoded in these representations provide useful information that often improves performance without additional supervised data (Kim, 2014; Xu et al., 2015b; Zhang et al., 2017, 2018b). Another approach is to reuse models trained on a related task. For example, Liu et al. (2018b) show that initializing a relation extraction model with weights of a trained named entity recognition model improved performance. This setup, however, requires supervised data to pre-train the NER model.

Most recently, pre-trained language representations (Dai and Le, 2015; Ramachandran et al., 2017), or contextual word embeddings, emerged as a very effective form of transfer learning. For example, Peters et al. (2018) demonstrate that substituting word embeddings with contextual ones improves performance on related NLP tasks, such as semantic similarity, coreference resolution, and semantic role labeling. Howard and Ruder (2018) show language representations learned by unsupervised language modeling to significantly improve text classification performance, to prevent overfitting, and to increase data efficiency. Radford et al. (2018) demonstrate that general-domain pre-training and task-specific fine-tuning achieves state-of-the-art results on several question answering, text classification, textual entailment, and semantic similarity tasks.

Contrary to previous approaches at the time of investigation, I use a sequential transfer learning method that utilizes a pre-trained language model, specifically its representations, as the source of prior knowledge by fine-tuning it to the relation extraction task. The method is less complex,

as it requires only a single model instead of multiple feature extractors. It also improves portability, because language model pre-training requires no supervision, i.e., it works on raw text. As no additional systems are involved, no discrete decisions are necessary, which mitigates error propagation. Ideally, pre-trained language representations encode a broad set of syntactic and semantic features that allow for feature selection during fine-tuning to the relation extraction task.

3.1.3 Contributions

I propose a sequential transfer learning method for supervised relation extraction and evaluate it against a set of baselines on standard benchmarks. Based on this, I create a state-of-the-art relation extraction model and further investigate the effect of pre-training on performance and data efficiency in depth. In more detail, the contributions are:

Sequential transfer learning method for supervised relation extraction I propose a novel method for supervised relation extraction that uses pre-trained language representations as the source of prior knowledge. The representations are obtained by language model pre-training on raw text without supervision, which considerably increases applicability and portability. A well performing language model must implicitly capture useful linguistic features that can be utilized during fine-tuning to the relation extraction task, instead of supplying features explicitly. The method's main advantage is that only a single model is necessary, which implicitly provides linguistic features, i.e., prior knowledge, and is also responsible for relation extraction. This mitigates error propagation and obviates additional feature extractors, such as part-of-speech taggers and dependency parsers. It also eliminates a priori feature selection, as the model can select the most useful features during fine-tuning. The proposed method demonstrates superior performance and data efficiency compared to earlier methods on two popular supervised relation extraction benchmarks.

Discussion of pre-training and its effect on performance and data efficiency I analyze and discuss the effect of pre-training on relation extraction performance, data efficiency, and the extent to which pre-trained language representations replace explicitly provided prior knowledge in the form of linguistic features. Based on the results of the qualitative evaluation, I conclude that a better understanding of the captured linguistic features is necessary to gain further insights into the strengths and limitations of transfer learning and neural network-based methods for relation extraction in general.

3.2 Transfer of Pre-Trained Language Representations

This section introduces a sequential transfer learning method for supervised relation extraction. It uses a pre-trained transformer-based language model, specifically its representations, as a source of prior linguistic knowledge, which is then transferred to the downstream relation extraction task by fine-tuning. The method is called *TRE*: the Transformer for Relation Extraction. First, I give a detailed overview of model architecture and input text representation (Section 3.2.1). In the following, I formally describe unsupervised language model pre-training (Section 3.2.2), and finally introduce the process of supervised fine-tuning to the relation extraction task (Section 3.2.3).

3.2.1 Model Architecture

TRE is a multi-layer transformer-decoder (Liu et al., 2018a), a decoder-only variant of the original transformer (Vaswani et al., 2017, Section 2.1.1.1). The model expects its input, e.g., a

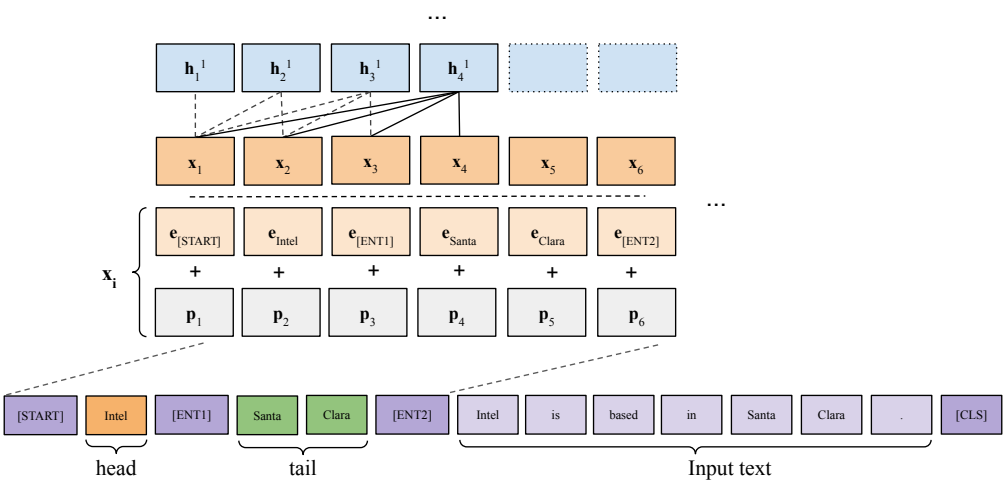


Figure 3.1 TRE input format and representation. Because the model expects a sequence of tokens, the input follows a specific structure suited for the relation extraction task: head and tail entity mention followed by the actual input text, all separated by special delimiters (*[START]*, *[ENT1]*, *[ENT2]*, *[CLS]*).

sentence or a document, as a sequence of tokens $t_1, \ldots, t_T$. The input text thus is first tokenized using byte pair encoding (BPE, Sennrich et al., 2016) to make use of sub-word information. The BPE algorithm creates a vocabulary of sub-word tokens, starting with single characters. It iteratively merges the most frequently co-occurring tokens into a new token until a predefined vocabulary size is reached. While language model pre-training is done on plain text, relation

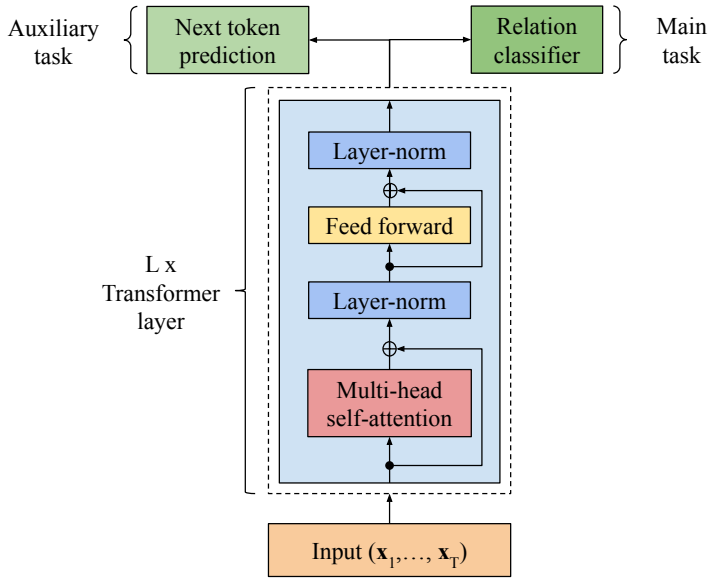


Figure 3.2 TRE architecture. The model applies L Transformer layers to the input token representations $\mathbf{x}_1, \ldots, \mathbf{x}_T$ followed by a classifier to predict the relation. Next token prediction (language modeling) is used as an auxiliary task to aid the training process.

extraction requires a more structured input, namely, a sentence[1] and a pair of entity mentions (in case of binary relation extraction). In order to avoid task-specific changes and extensions to the architecture, I adopt a traversal-style approach (Radford et al., 2018) that converts the structured input into an ordered sequence of tokens so it can be fed directly to the model. Figure 3.1 shows a visual illustration of the input format. It starts with the tokens belonging to both entity mentions, denoted *head* and *tail*, separated by delimiters (*[ENT1]*, *[ENT2]*), followed by the tokenized input text containing the mentions. The input token sequence is terminated by a special classification token (*[CLS]*) that signals the model to generate a sentence representation $\mathbf{s} \in \mathrm{R}^d$ at this position. Finally, the classifier predicts the relation based on $\mathbf{s}$. Since the model processes the input left-to-right, the entity mentions are added at the beginning in order to bias the attention mechanism towards their token representation while processing the sentence's token sequence. For each token t_i, its input representation $\mathbf{x}_i$ is obtained by first summing over its corresponding token embedding $\mathbf{e}_i$ and positional embedding $\mathbf{p}_i$. As depicted in Figure 3.2, the model then repeatedly encodes the sequence of input representations $\mathbf{x}_1, \ldots, \mathbf{x}_T$ by applying multiple transformer layers, each consisting of masked multi-head self-attention followed by a

[1] A "sentence" denotes an arbitrary span of contiguous text, rather than an actual linguistic sentence. During pre-training the input consists of multiple linguistic sentences, whereas relation extraction is typically applied to a single one. A "sequence" refers to the input token sequence.

position-wise feedforward operation:

$$\begin{aligned} &\{\mathbf{h}_i^0\}_{i=1,\ldots,T} = \mathbf{X} = \mathbf{t}\mathbf{W}_e + \mathbf{W}_p\,, \\ &\mathbf{h}_l = transformer\_layer(\{\mathbf{h}_i^{l-1}\}_{i=1,\ldots,T})\ \forall\ l\ \in\ [1,L]\,, \end{aligned} \tag{3.1}$$

where $\mathbf{t} = [t_1,\ldots,t_T]$ is a sequence of token indices in a sentence. $\mathbf{W}_e$ is the token embedding matrix, $\mathbf{W}_p$ is the positional embedding matrix, L is the number of transformer layers, and $\mathbf{h}_l^i$ is the i-th hidden state at layer l. Since the transformer has no implicit notion of token positions, the first layer adds a learned positional embedding $\mathbf{p}_i \in \mathbb{R}^d$ to each token embedding $\mathbf{e}_i \in \mathbb{R}^d$ at position i in the input sequence. The self-attentive architecture allows an output state $\mathbf{h}_i^l$ of a layer to be informed by all input states $\mathbf{h}_i^{l-1}$, which is key to efficiently model long-range dependencies. For language modeling, however, self-attention must be prohibited from attending to tokens ahead of the current position, i.e., the model must be prevented from looking into the future.

3.2.2 Unsupervised Pre-training of Language Representations

Relation extraction benefits from the ability to efficiently represent long-range dependencies (Zhang et al., 2018b) and hierarchical relation types (Han et al., 2018). Generative pre-training via language modeling can be seen as an ideal task for a model to learn representations that capture useful linguistic features, i.e., to acquire prior knowledge about language without supervision (Howard and Ruder, 2018; Linzen et al., 2016), before it is fine-tuned to the target task – in this case relation extraction.

Given a corpus $\mathcal{C} = [t_1,\ldots,t_M]$ of tokens t_i, the language modeling objective maximizes the likelihood:

$$L_{lang}(\mathcal{C}) = \sum_{i=k}^{M} \log P(t_i|t_{i-1},\ldots,t_{i-k+1};\theta)\,, \tag{3.2}$$

where k is the context window considered for predicting the next token t_i and θ denotes the model parameters. TRE models the conditional probability $P(t_i)$ by an output distribution over target tokens:

$$P(t_i) = softmax(\mathbf{h}_i^L \mathbf{W}_e^T)\,, \tag{3.3}$$

where $\mathbf{h}_i^L$ denotes the i-th hidden state after layer L and $\mathbf{W}_e$ is the embedding matrix.

3.2.3 Supervised Fine-tuning to Relation Extraction

After pre-training with the objective in Equation 3.2, the language model is fine-tuned to the relation extraction task. Assume a labeled dataset $\mathcal{D} = \{(t_i^1, \ldots, t_i^n, head_i, tail_i, r_i)\}_{i=1,\ldots,|\mathcal{D}|}$, where the i-th example consists of a token sequence $t_i^1, \ldots, t_i^n$ of length n, the spans $head_i$ and $tail_i$ of both entity mentions in the sequence of tokens, and a corresponding relation label r_i. The input sequence is fed to the pre-trained model to obtain the final hidden state representations $\{\mathbf{h}_i^L\}_{i=1,\ldots,m}$. To compute the output distribution $P(r_i)$ over relation labels, a linear layer followed by a softmax is applied to the representation $\mathbf{h}_m^L$, which represents a summary of the input sequence:

$$P(r_i|t_i^1, \ldots, t_i^n; head_i, tail_i) = softmax(\mathbf{h}_n^L \mathbf{W}_r) \tag{3.4}$$

During fine-tuning the following objective is optimized:

$$L_{rel}(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} \log P(r_i|t_i^1, \ldots, t_i^n, head_i, tail_i) \tag{3.5}$$

According to Radford et al. (2018), using language modeling as an auxiliary task (next token prediction in Figure 3.2) during fine-tuning leads to better generalization and faster convergence. I therefore adopt a similar objective:

$$L(\mathcal{D}) = \lambda * L_{lang}(\mathcal{D}) + L_{rel}(\mathcal{D}), \tag{3.6}$$

where λ is the language model weight, a scalar weighting the contribution of the language model objective during fine-tuning.

3.3 Evaluation

The goal of the evaluation is to determine whether pre-trained language representations capture prior knowledge that can effectively replace explicitly provided features. I evaluate the relation extraction performance of the proposed transfer learning method against baselines from previous work in a series of experiments. I then follow up on the results with an additional set of experiments to determine the effect of pre-training on overall performance and data efficiency. I begin this section with an outline of the evaluation setup, followed by a detailed discussion of the results.

3.3.1 Experimental Setup

I select two supervised relation extraction benchmarks for the experiments: SemEval 2010 Task 8 and the recently published TACRED (Section 2.3.4.2). Both datasets cover different aspects of the task, e.g., SemEval focuses on semantic relations between untyped nominals, whereas TACRED focuses on person- and organization-oriented relation types. It is also approximately 10x the size of SemEval and contains a much higher fraction of negative training examples (Table 3.1), which makes relation extraction more challenging.

Dataset	# Relations	# Examples	Negative examples (%)
SemEval 2010 Task 8	19	10,717	17.4%
TACRED	42	106,264	79.5%

Table 3.1 Statistics for evaluation datasets.

For each experiment, I initialize TRE with parameters from an existing pre-trained language model, then fine-tune it to the dataset and compare its performance to the selected baselines. As per convention, I report results on TACRED as micro-averaged F1 scores. Following the evaluation strategy of Zhang et al. (2017), the best performing model is selected based on the median validation F1 score over 5 independent training runs and its performance is reported on the test set. I follow the official convention of SemEval and report macro-averaged F1 scores with directionality taken into account, averaged over 5 independent training runs. In the first set of experiments, I evaluate the relation extraction performance of TRE on both benchmarks. For comparison, I select the following baselines:[2] Logistic regression (LR, Angeli et al., 2015b), support vector machine (SVM, Rink and Harabagiu, 2010), Tree-LSTM (Tai et al., 2015), BRCNN (Cai et al., 2016), DRNN (Xu et al., 2016b), PA-LSTM (Zhang et al., 2017), and C-GCN (Zhang et al., 2018b). For the second set of experiments, I select PCNN (Zeng et al., 2015) as the state-of-the-art baseline to study its data efficiency in comparison to *TRE*. I use the PCNN implementation of OpenNRE.[3]

I will now describe further details of the experimental setup.

[2] Section 2.3.3.1 describes the methods in more detail.
[3] https://github.com/thunlp/OpenNRE

3.3.1.1 Pre-trained Language Model

Since the main goal is to show the effectiveness of fine-tuning a pre-trained language model on the relation extraction task, I reuse the OpenAI GPT[4] published by Radford et al. (2018) for the experiments. The model was pre-trained on the BooksCorpus (Zhu et al., 2015), which contains around 7,000 unpublished books with a total of more than 800M words of different genres. It consists of $L = 12$ layers with 12 attention heads and 768 dimensional states, and a feed-forward layer of 3072 dimensional states. I also reuse the model's byte pair encoding (BPE) vocabulary, containing 40,000 tokens, but extend it with task-specific ones, e.g., entity mention delimiters. Also, I use the learned positional embeddings with supported sequence lengths of up to 512 tokens.

3.3.1.2 Entity Masking

For a subset of experiments I employ entity masking (Zhang et al., 2017). As shown in Figure 3.2, entity masking substitutes an entity mention with a placeholder according to the selected strategy. In the experiments, I use four different entity masking strategies. For the simplest masking strategy *UNK*, I replace all entity mentions with a special unknown token. For the *NE* strategy, I replace each mention with its named entity type. *GR* substitutes a mention with its grammatical role (subject or object). *NE + GR* combines both strategies. Entity masking has

Masking	Input
None	The measure included Aerolineas 's domestic subsidiary , Austral .
UNK	The measure included <UNK> 's domestic subsidiary , <UNK> .
GR	The measure included <SUBJ> 's domestic subsidiary , <OBJ> .
NE	The measure included <ORG> 's domestic subsidiary , <ORG> .
NE + GR	The measure included <SUBJ-ORG> 's domestic subsidiary , <OBJ-ORG> .

Table 3.2 Examples for entity masking strategies

been shown to provide a significant gain in relation extraction performance on TACRED, as it limits the information about a mention that is available to the model. This prevents overfitting to specific mentions and forces the model to focus more on the context. I also use it to simulate different scenarios, such as the presence of unseen entities, and to analyze the impact of entity type and grammatical role features on the model's performance.

[4] https://github.com/openai/finetune-transformer-lm

3.3.1.3 Hyperparameters and Optimization

During the experiments, I found the hyperparameters for fine-tuning, reported in Radford et al. (2018), to be very effective. Therefore, unless specified otherwise, the Adam optimization scheme (Kingma and Ba, 2015) is used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 8, and a linear learning rate decay schedule with warm-up over 0.2% of training updates. Residual, and classifier dropout is applied with a rate of 0.1. I also experimented with token dropout, i.e., randomly replacing a token embedding with the zero vector, but did not find it to improve performance. Table 3.3 shows the best performing hyperparameter configuration for each dataset. On SemEval 2010 Task 8, I first split 800 examples of the training set for hyperparameter selection and retrained on the entire training set with the best parameter configuration.

Dataset	Epochs	Learning rate	Warm-up learning rate	λ	Attention dropout
SemEval	3	6.25e-5	1e-3	0.7	0.15
TACRED	3	5.25e-5	2e-3	0.5	0.1

Table 3.3 Hyperparameter configurations for SemEval and TACRED

3.3.2 Results

In this section, I present the experimental results in comparing the proposed transfer learning method for relation extraction (TRE) to previous approaches on the two benchmark datasets, demonstrating that it achieves state-of-the-art performance even without explicitly provided linguistic features. I further provide results on model ablations to determine the effect of pre-training on relation extraction performance, and conduct a data efficiency comparison between TRE and the selected PCNN baseline from previous work.

3.3.2.1 Experiment 1: Relation Extraction Performance

As shown in Table 3.4, *TRE* achieves an F1 score of 67.4, outperforming all baselines on TACRED, including state-of-the-art methods. The results also show that methods with the ability to model complex syntactic and long-range dependencies, such as *PA-LSTM* and *C-GCN*,

perform better on TACRED. Outperforming these methods highlights the ability of pre-trained language representations to implicitly encode information similar to complex syntactic features and the ability to also capture long-range dependencies. It, however, is important to note that the result was obtained by using the same *NE + GR* entity masking strategy as in previous work (*PA-LSTM*, *C-GCN*). As described in Section 3.3.1.2, each entity mention is replaced by a special token, a combination of its named entity type and grammatical role. While the model achieves state-of-the-art results when provided only with named entity information, unmasked mentions decrease the test F1 score to 62.8. In Section 3.3.2.2, I analyze the effect of entity masking on task performance in more detail.

System	Precision	Recall	F1 score
LR† (Zhang et al., 2017)	72.0	47.8	57.5
CNN† (Zhang et al., 2017)	**72.1**	50.3	59.2
Tree-LSTM† (Zhang et al., 2018b)	66.0	59.2	62.4
PA-LSTM† (Zhang et al., 2018b)	65.7	64.5	65.1
C-GCN† (Zhang et al., 2018b)	69.9	63.3	66.4
TRE	70.1	**65.0**	**67.4**

Table 3.4 Test set performance on TACRED. The hyperparameters were selected on the validation set, and the test score is reported as the run with the median validation score among 5 randomly initialized runs. † marks results reported in the corresponding work.

Similar to TACRED, *TRE* outperforms the best previously reported methods on SemEval 2010 Task 8, establishing a new state-of-the-art score of 87.1 F1 (Table 3.5). The result indicates that pre-training with a language modeling objective forces the model to implicitly capture linguistic features that are useful for relation extraction, outperforming methods that rely on explicit lexical features (e.g., *SVM*). Further, *TRE* outperforms previous approaches that rely on explicit syntactic features, such as shortest dependency path information and distributed representations of part-of-speech tags and named entity types (e.g., *BRCNN*, *DRNN*, and *C-GCN*).

I also observe a high correlation between entity mentions and relation labels, analogous to Zhang et al. (2018b). According to the authors, simplifying SemEval sentences in the training and validation set to contain only subject and object, where subject and object refer to the entity mentions, already achieves an F1 score of 65.1. To better evaluate the model's (*TRE*) ability to generalize beyond entity mentions, I use the *UNK* entity masking strategy to substitute each mention in the training set with a special unknown token. This strategy prevents the model from observing any entity mentions during training and thus avoids overfitting to mentions that strongly correlate with specific relations. In this setting, *TRE* achieves an F1 score of

System	Precision	Recall	F1 score
SVM† (Rink and Harabagiu, 2010)	–	–	82.2
PA-LSTM† (Zhang et al., 2018b)	–	–	82.7
C-GCN† (Zhang et al., 2018b)	–	–	84.8
DRNN† (Xu et al., 2016b)	–	–	86.1
BRCNN† (Cai et al., 2016)	–	–	86.3
PCNN (Zeng et al., 2015)	86.7	86.7	86.6
TRE	88.0	86.2	**87.1** ($\pm$0.16)

Table 3.5 Test set performance on SemEval 2010 Task 8. † marks results reported in the corresponding papers. Mean and standard deviation are computed across 5 randomly initialized runs.

79.1 (Table 3.6), an improvement of 2.6 points F1 score over state-of-the-art baselines. The result suggests that language model pre-training also improves the model's ability to generalize beyond the mention level when predicting the relation between two previously unseen entities.

System	Precision	Recall	F1 score
PA-LSTM† (Zhang et al., 2018b)	–	–	75.3
C-GCN† (Zhang et al., 2018b)	–	–	76.5
TRE	80.3	78.0	**79.1** ($\pm$ 0.37)

Table 3.6 SemEval 2010 Task 8 test set performance with *UNK* entity masking, where each mention is replaced by a special unknown token. † marks results reported in the corresponding papers. Due to the small test set size, mean and standard deviation are reported across 5 randomly initialized runs.

3.3.2.2 Experiment 2: Effect of Pre-training

Although the previous experiments demonstrated strong empirical results, the contributions of individual parts of *TRE* are still unclear. I therefore first conduct an ablation study to determine the relative impact of each model component on overall performance, followed by experiments to validate if pre-trained language representations capture linguistic properties that are useful to relation extraction, and whether pre-training also improves data efficiency.

Effect of pre-training Pre-training affects the two main parts of the model: Language model and byte pair embeddings, i.e., the distributed representations of the input token sequence. In Table 3.7, I first compare the best model, initialized with pre-trained parameters, to one with randomly initialized parameters. The results on both datasets clearly show that fine-tuning

considerably benefits from a pre-trained language model. In case of the SemEval, the validation F1 score increases to 85.6 when fine-tuning with a pre-trained language model and no entity masking, compared to 75.6 without pre-training – a 13% increase in performance. The results show an even more pronounced performance gain for TACRED, where a pre-trained language model increases the validation F1 score by 20 to 63.3. With entity masking, performance gains are slightly lower, at +8 on SemEval and +9.4 (*UNK*) respectively +3.8 (*NE + GR*) on TACRED. The larger effect of pre-training when entity mentions are unmasked suggests that it has a regularizing effect that prevents overfitting to specific mentions. In addition, the contextualized features allow the model to better adapt to complex entities. These observations are consistent with the results of Howard and Ruder (2018), who observed that language model pre-training considerably improves text classification performance on small and medium-sized datasets, similar to those used in the experiments.

	SemEval		TACRED		
	None	UNK	None	UNK	NE + GR
Best model	**85.6**	76.9	63.3	51.0	**68.0**
– w/o pre-trained LM	**75.6**	68.2	43.3	41.6	**64.2**
– w/o pre-trained LM and BPE	55.3	**60.9**	38.5	38.4	**60.8**

Table 3.7 Model ablations on SemEval and TACRED validation set with different entity masking strategies. "w/o pre-trained LM" uses randomly initialized language model parameters, "w/o pre-trained LM and BPE" also randomly initializes the byte pair embeddings before fine-tuning. F1 scores are reported over 5 independent runs.

In addition, I train a model from scratch without pre-trained byte pair embeddings. I keep the vocabulary of sub-word tokens fixed and randomly initialize the embeddings. Again, the results show that both datasets clearly benefit from pre-trained byte-pair embeddings. Due to its small size SemEval benefits considerably more from pre-trained embeddings, because the model may be unable to learn reliable embeddings from a corpus of this size. This increases the risk of overfitting to entity mentions as suggested by the lower performance compared to *UNK* masking where entity mentions are hidden from the model during training. For TACRED, model performance drops by approximately $3\% - 5\%$ with and without entity masking when not using pre-trained byte pair embeddings.

Linguistic features captured by pre-trained language representations Undoubtedly, linguistic features, such as part-of-speech and entity type information, are crucial to relation

extraction performance. This is confirmed by the superior performance on TACRED (Table 3.7) when entity and grammatical role information is provided explicitly by *NE + GR* masking. The model achieves a validation F1 score of 68.0, compared to 63.3 without augmenting the entity mentions. Without a pre-trained language model, *NE + GR* masking still achieves a F1 score of 64.2, which suggests that pre-trained language representations capture features that are as informative as providing entity type and grammatical role information. This is also suggested by Peters et al. (2018), who show that a language model captures syntactic and semantic information useful for a variety of natural language processing tasks, such as part-of-speech tagging and word sense disambiguation.

Effect of entity masking Entity masking, as described in Section 3.3.1.2, is used to limit the information about entity mentions that is available to a model. It can be used to simulate different scenarios, such as the presence of unseen entities, to prevent overfitting to specific entity mentions, and to focus more on context. Table 3.8 shows F1 scores on the TACRED validation set for different entity masking strategies. As previously shown, masking with entity and grammatical role information yields the best overall performance, yielding a F1 score of 68.0. The results indicate that different entity masking strategies mostly impact recall, while precision tends to remain high – with the exception of the *UNK* masking strategy. Applying

Entity masking	Precision	Recall	F1 score
None	**69.5**	58.1	63.3
UNK	56.9	46.3	51.0
GR	63.8	50.1	56.1
NE	68.8	65.3	67.0
NE + GR	68.8	**67.2**	**68.0**

Table 3.8 TACRED validation F1 scores with different entity masking strategies.

UNK masking prevents the model from observing anything about the entity mention, which greatly decreases precision and consequently the F1 score drops to 51.0. Using grammatical role information (*GR*) considerably increases performance to an F1 score of 56.1. This result may suggest that the semantic role type is a very helpful feature, however, its importance may also be attributed to the fact that it provides robust information on where each argument entity is positioned in the input sentence. *NE* masking significant increases recall, which intuitively suggests a better generalization ability of the model. Combining *NE* masking with grammatical role information yields only a minor gain in recall, which increases from 65.3% to 67.2%, while precision stays at 68.8%.

3.3.2.3 Experiment 3: Data Efficiency

The main goal of transfer learning is to improve data efficiency, i.e., the amount of supervised data that is necessary to achieve a certain performance. I thus expect *TRE* to reach a defined performance, e.g., F1 score, more quickly than a baseline that is trained on the same fraction of the data. To assess their data efficiency, I train *TRE* and the *PCNN* baseline on stratified subsets of the TACRED training set with sampling ratios from 10% to 100%. Further, I train a variant with *NE + GR* entity masking and randomly initialized language model (w/o LM). I train each variant on all subsets and evaluate its performance on the validation set using micro-averaged F1 score averaged over 5 independent training runs.

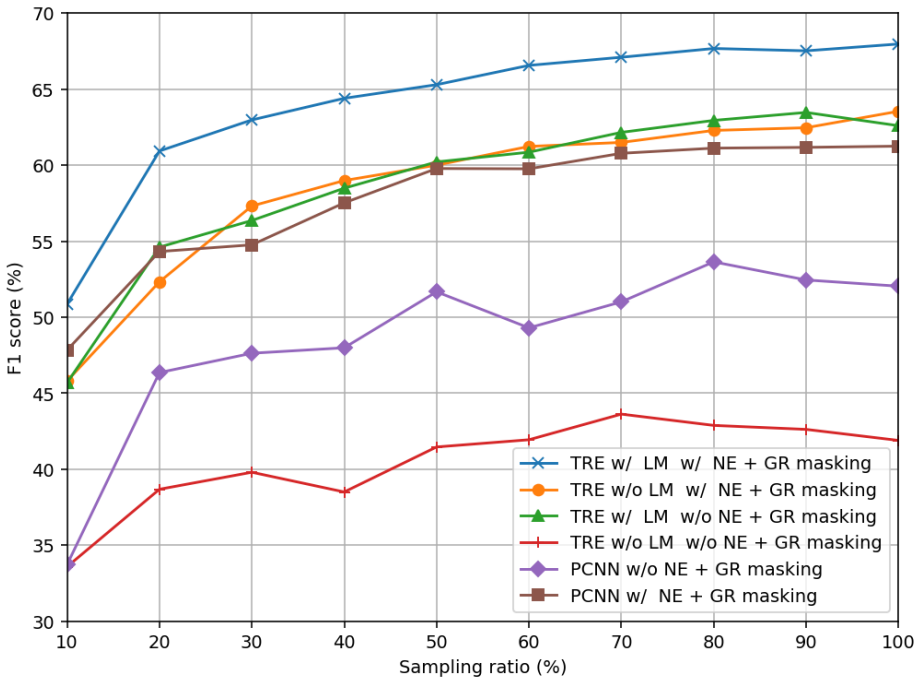


Figure 3.3 Validation F1 score over increasing sampling ratios of the training set, averaged over 5 runs.

The results in Figure 3.3 show that the best performing model uses a pre-trained language model (*TRE*) combined with NE+GR masking, which consistently performs better than the other variants of *TRE* and the baseline *PCNN*. There is a steep performance increase in the first part of the curve, when only a small subset of the training examples is used. The model reaches an F1 score of more than 60 with only 20% of the training data, and continues to

improve with more training data. The next best models are *TRE* without a pre-trained language model, followed by *TRE* without *NE+GR* masking. Both perform very similar, which aligns well with the previous observations. The *PCNN* baseline performs well with entity masking applied, but slightly drops in performance compared to *TRE* after 30% of training data, slowly approaching a performance plateau of around 61 F1 score. *PCNN* without masking performs worse, but improves steadily due to its low base score. *TRE* with randomly initialized language model seems to overfit early and diminishes in performance with more than 70% training data. Interestingly, the performance of several models drops or stagnates after about 80% of the training data, which might indicate that these examples do not increase the models' ability to generalize to unseen data.

3.4 Discussion and Summary

In this chapter, I investigated a transfer learning method for supervised relation extraction that uses a pre-trained language model and its latent representations as a source of prior knowledge to mitigate the need for explicitly provided linguistic features. Instead of multiple systems dedicated to feature extraction, this method is based on a single model and thus reduces error propagation. It also eliminates a priori feature selection, as this is implicitly done during fine-tuning to the relation extraction task. In a series of experiments on two standard relation extraction benchmarks, I demonstrated that the proposed method consistently outperforms the baselines, including state-of-the-art methods from previous work. In additional experiments, I showed that pre-training considerably increases data efficiency, and also suggested that useful linguistic features encoded in pre-trained language representations are one of the main reasons for the observed increase in relation extraction performance and data efficiency.

Despite the strong empirical results, I found two aspects of the proposed approach demand further investigation. First, it is still unclear to what extent linguistic information is encoded in language representations and how it is used by the model to predict a relation between entity mentions in a sentence. To gain further insights and identify areas for improvement it would be helpful to determine the linguistic features encoded by neural network-based relation extraction models. Their complexity, however, makes it difficult to understand this process. This is a problem inherent to all models with highly non-linear structure, such as neural networks, which makes it difficult to explain their inner workings. Secondly, while the experiments show that pre-training increases data efficiency, they also show that increasing the amount

of supervised data benefits overall performance. Creating labeled examples, however, is still costly as annotation is a time consuming process.

I conclude that the increased complexity demands a better understanding of the aspects about the input a model considers relevant for its decision. I investigate this in more detail in Chapter 5. Also, I observe that relation extraction performance still considerably benefits from labeled examples, despite the improvements in data efficiency. I therefore conclude that a combination of the proposed transfer learning method with an approach to automatically annotate more examples, e.g., distant supervision, could further improve performance. This is investigated in further detail in the next chapter.

Chapter 4

Combining Sequential Transfer Learning with Distant Supervision

4.1 Introduction

As I demonstrated in the previous chapter, sequential transfer learning increases relation extraction performance as well as data efficiency in the supervised setting. The results, however, also show that even with pre-training a considerable amount of supervised data is necessary, and that more labeled data further increases performance. Creating a well performing relation extraction model in this scenario would still be costly and time consuming, due to the effort of manual annotation. One way to further reduce the cost is to combine transfer learning with methods that automatically label data, for example, via heuristics. In this chapter, I therefore extend the transfer learning method proposed in the previous chapter to the distantly supervised setting.

I first discuss the challenge of creating models from distantly supervised data and how this is commonly addressed in relation extraction (Section 4.1.1). In the following, I look at previous works and highlight the limitations of current methods that combine multi-instance learning with additional linguistic features and side information to mitigate this problem (Section 4.1.2). I then introduce a method to extend the model developed in Chapter 3 to distantly supervised relation extraction (Section 4.2). To demonstrate its effectiveness, I conduct an extensive automated and manual evaluation in which I compare its performance to state-of-the-art

baselines (Section 4.3). Finally, I discuss the proposed approach and its alignment with the objectives of this thesis (Section 4.4).

This chapter is based on a previously published full paper (Alt et al., 2019a).

4.1.1 Problem Statement

The major challenge in distantly supervised relation extraction is the presence of label noise, which is a result of the heuristic labeling process (Section 2.3.2.2). Multi-instance learning (Riedel et al., 2010; Surdeanu et al., 2012) is the most common approach to create models in the presence of noisily supervised data. It groups all examples with the same entity pair, e.g., Jonathan Lethem and Brooklyn, into a "bag". The task is to identify a target relation between the entities on a bag-level, under the assumption that at least one of the examples in a bag properly expresses this relation. Intuitively, this approach forces a model to decide which patterns or features from a set of examples are on average most indicative of a relation – without knowledge of which example is actually labeled correctly. Recently, there has been increased interest in using additional knowledge in the form of linguistic features and side information to improve distantly relation extraction performance (Vashishth et al., 2018; Yaghoobzadeh et al., 2017).

Problem 1: Explicitly provided prior knowledge Similar to the supervised setting, methods in distantly supervised relation extraction leverage prior knowledge by explicitly providing it as features alongside the input text. These features include: prefix and morphological properties, lexical features of words, and syntactic features. Recent methods (e.g., Vashishth et al., 2018) utilize additional side information, e.g., paraphrases, relation aliases, and entity types to better inform the relation extraction. Similar to supervised relation extraction methods, these features are often represented in a vector space to better model their compositionality and semantics.

Consequently, this leads to limitations similar to the supervised setting. To obtain useful features, dedicated systems are necessary, e.g., part-of-speech taggers, dependency parsers, or knowledge bases. Each additional system introduces a potential source of error that propagates through subsequent steps, and further introduces additional complexity. Again, the set of features is selected a-priori; creating the feature extractors requires supervised data in addition

to the labeled task-specific data, which may not be readily available for the desired language or domain; and useful side knowledge may be unavailable in structured form.

Problem 2: Bias towards a subset of relations A key problem of multi-instance learning is that it creates a bias towards relations that are frequently expressed by similar, often simple, patterns, as confirmed by preliminary experiments. For example, a LocationContains relation may be frequently expressed as "<Location>, <Location>", e.g., "Berlin, Germany". This leads to a situation where a model recognizes a small set of relations with high precision, while performing poorly on the remaining ones in the long-tail (Kirschnick et al., 2014). This limits the applicability of the approach, because many relations of interest do not appear frequently in the training dataset and may be expressed in a more complex fashion. Background knowledge could help to mitigate this problem (Rocktäschel et al., 2015).

A method that addresses these limitations therefore should consist of a single system to reduce the overall complexity; learn linguistic features and entity related background knowledge without supervision; efficiently learn on noisily supervised data; and effectively recognize complex syntactic patterns from a small set of examples.

4.1.2 Related Work

I first discuss previous work that utilizes prior knowledge and side information to improve distantly supervised relation, which I use to determine evaluation baselines (Section 4.1.2.1). In the following, I review approaches that are commonly used to create relation extraction models with noisily supervised data (Section 4.1.2.2).

4.1.2.1 Use of Prior Knowledge

Linguistic features Similar to the supervised setting, previous works utilize various discrete linguistic features to support the relation extraction process. Early distantly supervised approaches (Mintz et al., 2009) use prefix and morphological features; lexical features of specific words between and surrounding the mentions; flags that indicate which mention appears first in the text; and a window of words to the left and right of the entity mentions. They also utilize syntactic features, e.g., the dependency path between mentions, and named entity types of both arguments. Other works used argument features, e.g., the head word of each mention and

their combination (Surdeanu et al., 2011); syntactic features, e.g., the sequence of dependency links between mention heads and the lemma of all words on the path (Hoffmann et al., 2011); and surface features, e.g., flags indicating the presence of a particular entity type between mentions (Surdeanu et al., 2012). Similar, He et al. (2018) use dependency parse features to obtain distinct sentence and mention level representations via a tree based recurrent neural network (Tree-RNN), which are subsequently combined for relation classification.

Side information and background knowledge Recent work shows that in addition to linguistic features, side information, or background knowledge, is helpful to improve relation extraction. Yaghoobzadeh et al. (2017) use entity type information to constrain the relations considered by the model. The authors observed that a relation typically holds only between a subset of entity types, which reduces the set of possible relations. RESIDE (Vashishth et al., 2018) combines dependency parse and entity type information with open information extraction (Angeli et al., 2015a) to extract relation instances without the need for a pre-defined ontology. The extracted instances, however, must be matched against the pre-defined set of relations, for which they use known relation aliases from existing knowledge bases, e.g., Freebase or Wikidata.

4.1.2.2 Learning with Noisily Supervised Data

Multi-instance learning (Riedel et al., 2010) is a popular approach to create a relation extraction model from noisily supervised data. It assumes that in a group of examples (bag) with the same pair of entities at least one example is labeled with the correct relation. Multi-instance multi-label learning (Hoffmann et al., 2011; Surdeanu et al., 2012) extended this setting to account for the fact that multiple relations can hold between a pair of entities. With the increasing popularity of neural networks, PCNN (Zeng et al., 2014) became the most widely used architecture for distantly supervised relation extraction, e.g., with an extensions for multi-instance learning (Zeng et al., 2015). Initial multi-instance learning methods select only the most likely example in a group. Lin et al. (2016) instead proposed to use selective attention (PCNN+ATT) to obtain an attention distribution over the sentence representations. This distribution is then used to compute a bag representation as a weighted combination of all sentence representations. Other methods include adversarial training Qin et al. (2018); Wu et al. (2017), noise models Luo et al. (2017), and soft labeling Liu et al. (2017a); Wang et al. (2018b).

In contrast to earlier work in distantly supervised relation extraction at the time of investigation, I use sequential transfer learning based on a pre-trained language model as the source of supporting linguistic features and entity related background knowledge. Again, the method is less complex than previous systems, reduces error propagation, and does not require a priori feature selection. To enable learning on distantly supervised data, I extend the fine-tuning step with a selective attention mechanism for multi-instance and multi-instance multi-label learning on groups of examples.

4.1.3 Contributions

I extend the transfer learning approach introduced in the previous chapter to distantly supervised relation extraction and evaluate it against a set of baselines on a standard benchmark. Based on this, I create a state-of-the-art relation extraction system and further investigate the performance on relations in the long tail. In more detail, the contributions are:

Sequential transfer learning method for distantly supervised relation extraction I introduce a novel approach that extends the sequential transfer learning method presented in Chapter 3 to distantly supervised relation extraction. Similarly, it uses pre-trained language representations to replace explicitly provided linguistic features, which reduces error propagation and overall complexity. To efficiently learn on distantly supervised data, it utilizes selective attention to aggregate sentence-level evidence that is then used to produce bag-level predictions for multi-instance (multi-label) learning. In an extensive evaluation, I show that pre-trained language models are better suited for distant supervision; more effectively guiding the relation extraction with the knowledge acquired during unsupervised pre-training. Although the proposed method achieves lower precision for the top ranked predictions, it shows a state-of-the-art area under curve (AUC) score compared to earlier approaches that rely on additional linguistic features and side information. Importantly, I demonstrate an overall more balanced performance, and show that it recognizes a more diverse set of relations and performs especially well at higher recall levels.

Discussion of pre-training and how it impacts performance on infrequent relations I analyze and discuss the effect of pre-training on distantly supervised relation extraction with multi-instance learning. In particular, I provide an extensive manual analysis of the bias towards a small subset of relations, including state-of-the-art methods of earlier work. I conclude that transfer learning is able to improve performance on long-tail relations

and that a better understanding of neural network-based models and the (linguistic) knowledge they encode is necessary to further improve performance.

4.2 Multi-Instance Learning with Pre-trained Language Representations

This section introduces *DISTRE*: the Distantly Supervised Transformer for Relation Extraction. It extends TRE, the sequential transfer learning method introduced in the previous chapter, by a mechanism that allows us to create a model from distantly supervised data. First, I briefly introduce the model architecture and extension compared to the supervised setting (Section 4.2.1), followed by a detailed description of my approach to distantly supervised fine-tuning on the target relation extraction task (Section 4.2.2).

4.2.1 Model Architecture

As an extension to TRE, DISTRE shares the same multi-layer transformer-decoder architecture. It, however, utilizes multi-instance learning to mitigate the issues of noisily supervised data. Instead of a single input, it expects a group (bag) of input texts mentioning the same pair of entities, e.g., (Jonathan Lethem, Brooklyn). Each text in a bag is first tokenized using byte pair encoding then converted into an ordered sequence of tokens that can be directly supplied to the model. Similar to TRE, each sequence starts with the tokens belonging to both entity mentions separated by delimiters, which is followed by the token sequence of the actual text containing the mentions. The input sequence is terminated by a special classification token that signals the model to generate its representation $\mathbf{s} \in \mathrm{R}^d$ at this position.

To determine the relations a pair of entities participates in, I first supply the model with each input $\{(\mathbf{x}_1^i, \ldots, \mathbf{x}_T^i)\}_{i=1,\ldots,p}$ contained in a bag to obtain its representation $\mathbf{s}_i \in \mathbb{R}^d$. Then, I use selective attention to compute a bag level representation $\mathbf{s}_{bag}$ as a weighted combination of the individual representations $\{\mathbf{s}_1, \ldots, \mathbf{s}_p\}$, as shown in Figure 4.1. The step can be thought of as a denoising step that filters, or down-weights, sentences that do not indicate a particular relation, while retaining those that do.

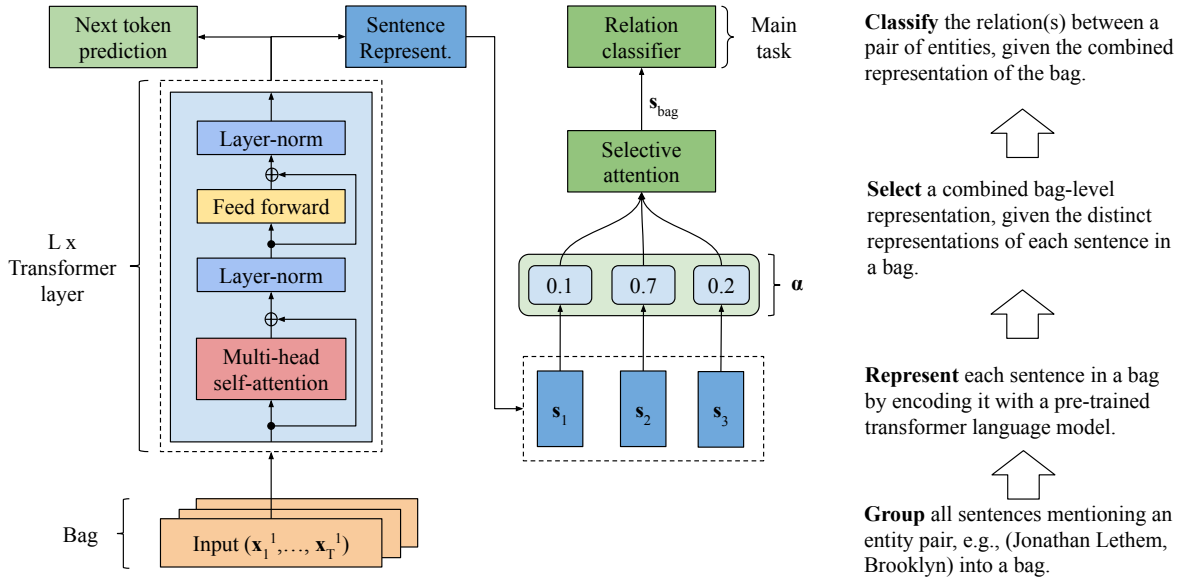


Figure 4.1 DISTRE architecture. In multi-instance learning, we group all input texts mentioning a particular entity pair, e.g., (Jonathan Lethem, Brooklyn), into a bag and apply the model to each input. The model computes a sentence representation $\mathbf{s}_j \in \mathbb{R}^d$ for each input in the bag $\{\mathbf{x}_1^j, \ldots, \mathbf{x}_T^j\}_{j=1,\ldots,p}$ by applying L transformer layers. After encoding all inputs, we obtain a bag level representation $\mathbf{s}_{bag}$ by selectively attending over the individual representations $[\mathbf{s}_1, \ldots, \mathbf{s}_p]$. Finally, we classify all relation(s) the two entities participate in based on $\mathbf{s}_{bag}$.

4.2.2 Distantly Supervised Fine-tuning on Relation Extraction

Similar to *TRE*, language model pre-training seeks to maximize the following likelihood, given a corpus $\mathcal{C} = [t_1, \ldots, t_M]$ of tokens t_i:

$$L_{lang}(\mathcal{C}) = \sum_{i=k}^{M} \log P(t_i | t_{i-1}, \ldots, t_{i-k+1}; \theta), \tag{4.1}$$

where k is the context window of tokens that is considered for predicting the next token t_i via the conditional probability $P(t_i)$, and θ are the model parameters optimized during pre-training. The distribution over target tokens is computed according to the following equation:

$$P(t_i) = softmax(\mathbf{h}_i^L \mathbf{W}_e^T), \tag{4.2}$$

where $\mathbf{h}_i^L$ denotes the hidden state after the final layer L, and $\mathbf{W}_e$ is the embedding matrix.

After pre-training the language model is fine-tuned to the relation extraction task. Similar to the supervised setting, we assume a labeled dataset $\mathcal{D} = \{(t_i^1, \ldots, t_i^n, head_i, tail_i, r_i)\}_{i=1,\ldots,|\mathcal{D}|}$, where

each example consists of an input sequence of tokens $t_i^1, \ldots, t_i^n$, the positions $head_i$ and $tail_i$ of the relation's head and tail entity mention in the sequence of tokens, and the corresponding relation label r_i, which is assigned by a distant supervision. The label r_i, however, is an unreliable target on its own, because it may be incorrectly assigned. Relation(s) therefore are classified based on a bag, which contains all inputs $\{(\mathbf{x}_1^i, \ldots, \mathbf{x}_T^i)\}_{i=1,\ldots,p}$ that mention a particular entity pair, e.g., (Jonathan Lethem, Brooklyn). The model is applied to each of the inputs to compute its representation $\mathbf{s}_i = \mathbf{h}_j^L$, where j denotes the position of the special classification token (*[CLS]*, Figure 3.1) in the token sequence. A set representation $\mathbf{s}_{bag}$ is then derived as a weighted sum over the individual representations:

$$\mathbf{s}_{bag} = \sum_{i=1}^{p} \alpha_i \mathbf{s}_i \,, \tag{4.3}$$

where α_i is the weight assigned to the corresponding representation $\mathbf{s}_i$. The bag representation $\mathbf{s}_{bag}$ is then used to inform the relation classification.

I use selective attention (Lin et al., 2016) to aggregate a bag-level representation $\mathbf{s}_{bag}$ given the individual representations $\mathbf{s}_i$, as shown in Figure 4.1 (middle). In contrast to average selection, where each representation contributes equally to the bag level representation, selective attention learns to identify the representations with features most indicative of a relation, while de-emphasizing those that contain noise. The weight α_i is obtained for each input by comparing its representation against a learned relation representation $\mathbf{r} \in \mathbb{R}^d$:

$$\alpha_i = \frac{exp(\mathbf{s}_i \mathbf{r})}{\sum_{j=1}^{p} exp(\mathbf{s}_j \mathbf{r})} \tag{4.4}$$

To compute the output distribution $P(r)$ over relation labels, a linear layer followed by a softmax is applied to $\mathbf{s}_{bag}$:

$$P(r|\{\mathbf{x}_1^i, \ldots, \mathbf{x}_T^i\}_{i=1,\ldots,p}; \theta) = softmax(\mathbf{W}_r \mathbf{s}_{bag} + \mathbf{b}), \tag{4.5}$$

where $\mathbf{W}_r$ is the representation matrix of relations $\mathbf{r}$ and $\mathbf{b} \in \mathbb{R}^d$ is a bias vector. During fine-tuning the following objective is optimized:

$$L_{bag}(\mathcal{D}) = \sum_{i=1}^{|\mathcal{B}|} \log P(r_i|\{(\mathbf{x}_1^j, \ldots, \mathbf{x}_T^j)\}_{j=1,\ldots,|\mathcal{B}_i|}; \theta) \,, \tag{4.6}$$

where $|\mathcal{B}|$ denotes the number of unique bags. Similar to fine-tuning in the supervised setting, Equation 4.1 and Equation 4.6 are combined by a scalar weight λ:

$$L(\mathcal{D}) = \lambda * L_{lang}(\mathcal{D}) + L_{bag}(\mathcal{D}) \tag{4.7}$$

4.3 Evaluation

The main evaluation goal is to determine if language model pre-training and fine-tuning combined with multi-instance learning improves relation extraction performance in the distantly supervised setting. In particular, I study whether the method can replace explicitly provided side information and background knowledge, and if it improves performance on less frequently observed relations. I therefore evaluate the proposed method in a series of experiments on a standard benchmark, followed by a manual evaluation to compare predicted relations to those of earlier methods. In this section, I first outline the evaluation setup (Section 4.3.1), then discuss the results in detail (Section 4.3.2).

4.3.1 Experimental Setup

For the experiments, I use the distantly supervised NYT-10 (Section 2.3.4.2), or Riedel dataset, as the benchmark. Similar to the experiments on TRE, I initialize DISTRE from an existing pre-trained language model, then fine-tune it to the dataset and compare its performance to the selected baselines. As per convention, I report precision at the top N most confident predictions, as well as the area under curve (AUC), and provide a precision-recall curve. For comparison, I select the method proposed by Mintz et al. (2009) (MINTZ) as a baseline, which is a multi-class logistic regression trained directly on the distantly supervised data. Further, I select two state-of-the-art methods: PCNN+ATTN (Lin et al., 2016), the piecewise convolutional neural network (PCNN) that segments each input text into three parts, to the left, middle, and right of the entity mentions, followed by selective attention to compute a bag level representation for relation classification; and RESIDE (Vashishth et al., 2018), which first encodes the input text via a bidirectional recurrent neural network, followed by a graph convolutional neural network (GCN) to encode the explicitly provided dependency parse tree. The computed representation

is combined with named entity type information to obtain the final input representation that can be used in the selective attention step.

In the following, I describe additional details of the experimental setup.

4.3.1.1 Pre-Trained Language Model

Since the main goal is to show the effectiveness of fine-tuning a pre-trained language model on the relation extraction task, I reuse the OpenAI GPT published by Radford et al. (2018) for the experiments. The model was pre-trained on the BooksCorpus (Zhu et al., 2015), which contains around 7,000 unpublished books with a total of more than 800M words of different genres. It consists of $L = 12$ layers (blocks) with 12 attention heads and 768 dimensional states, and a feed-forward layer of 3072 dimensional states. I also reuse the model's byte pair vocabulary, containing 40,000 tokens, but extend it with task-specific ones, e.g., entity mention delimiters. Also, I use the learned positional embeddings with supported sequence lengths of up to 512 tokens.

4.3.1.2 Hyperparameters and Optimization

For the experiments, I use the Adam optimization scheme (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 8, and a linear learning rate decay schedule with warm-up over 0.2% of training updates. I also apply residual, and attention dropout with a rate of 0.1, and classifier dropout with a rate of 0.2.

4.3.2 Results

This section presents the experimental results in comparing DISTRE to earlier work on the NYT-10 dataset. In the first experiment, I conduct an automated evaluation to study distantly supervised relation extraction performance in comparison to the baselines. The automated evaluation can only provide an approximate result, because the test set is distantly supervised, too. I therefore conduct an additional manual evaluation and show that the proposed method recognizes a more diverse set of relations, while still achieving a state-of-the-art AUC – even without explicitly provided side information and linguistic features.

4.3.2.1 Experiment 1: Relation Extraction Performance

Table 4.1 shows the results of *DISTRE* compared to the baselines on the held-out (test) dataset. *DISTRE* with selective attention achieves a state-of-the-art AUC value of 0.422. The precision-recall curve in Figure 4.2 shows that it outperforms *RESIDE* and *PCNN+ATT* at higher recall levels, while precision is lower for top predicted relation instances. The results of the *PCNN+ATT* model indicate that its performance is only better in the very beginning of the curve, but its precision drops early and only achieves an AUC value of 0.341. Similar, *RESIDE* performs better in the beginning but drops in precision after a recall-level of approximately 0.25. This demonstrates that *DISTRE* yields a more balanced overall performance, which benefits applications that rely on extracting long-tail relations with high precision.

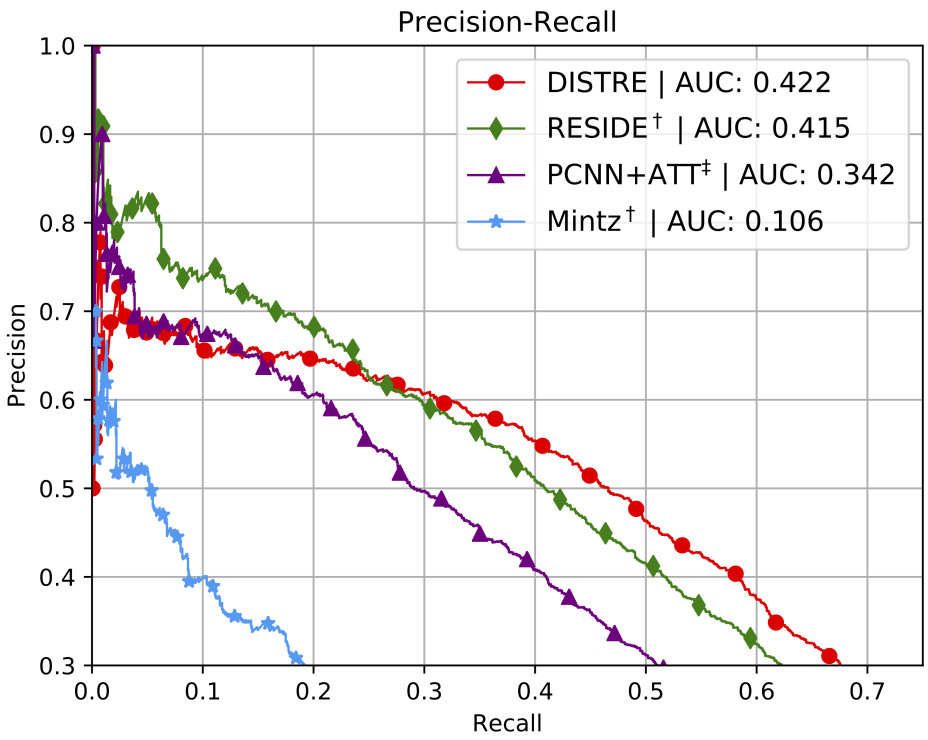


Figure 4.2 Precision-recall curve on NYT-10. *DISTRE* shows a more balanced performance across relations, especially in the long tail. † marks results reported by Vashishth et al. (2018). ‡ indicates results obtained with the OpenNRE implementation.

Table 4.1 also shows detailed precision values measured at different points along the PR curve. Again, we can observe that while *DISTRE* has lower precision for the top 500 most confidently

predicted relation instances, it shows a state-of-the-art precision of 60.2% for the top 1000 and continues to perform higher for the remaining, much larger part of the predictions.

System	AUC	P@100	P@200	P@300	P@500	P@1000	P@2000
Mintz†	0.107	52.3	50.2	45.0	39.7	33.6	23.4
PCNN+ATT‡	0.341	73.0	68.0	67.3	63.6	53.3	40.0
RESIDE†	0.415	**81.8**	**75.4**	**74.3**	**69.7**	59.3	45.0
DISTRE	**0.422**	68.0	67.0	65.3	65.0	**60.2**	**47.9**

Table 4.1 Precision evaluated automatically for the top rated relation instances. † marks results reported in the original paper. ‡ marks results obtained by the OpenNRE implementation.

Since automated evaluation on a distantly supervised, held-out dataset does not reflect the actual performance of the models given false positive labels and incomplete knowledge base information, I also evaluate all models manually. This also allows to gain a better understanding of the difference of the models in terms of their predictions. To this end, three human annotators manually rated the top 300 predicted relation instances for each model. Annotators were asked to label a predicted relation as correct only if it expressed a true fact at some point in time (e.g., for a */business/person/company* relationship, a person may have worked for a company in the past, but not currently), and if at least one sentence clearly expressed this relation, either via a syntactic pattern or via an indicator phrase.

Table 4.2 shows the P@100, P@200, P@300 and average precision scores, averaged over all annotators. *PCNN+ATT* has the highest average precision at 94.3%, 3% higher than the 91.2% of *RESIDE* and 5% higher than *DISTRE*. However, it can be seen that this is mainly due to *PCNN+ATT's* very high P@100 and P@200 scores. For P@300, all models have very similar precision scores. *PCNN+ATT's* scores decrease considerably, reflecting the overall trend of its PR curve, whereas *RESIDE's* and *DISTRE's* manual precision scores remain at approximately the same level. The model's precision scores for the top rated predictions are around 2% lower than those of *RESIDE*, confirming the results of the held-out evaluation. Manual inspection of *DISTRE's* output shows that most errors among the top predictions arise from wrongly labeled */location/country/capital* instances, which the other models do not predict among the top 300 relations.

System	P@100	P@200	P@300	Avg.
PCNN+ATT	97.3	94.7	90.8	94.3
RESIDE	91.3	91.2	91.0	91.2
DISTRE	88.0	89.8	89.2	89.0

Table 4.2 Precision evaluated manually for the top 300 relation instances, averaged across 3 human annotators.

4.3.2.2 Experiment 2: Performance on Infrequent Relations

Table 4.3 shows the distribution over relation types for the top 300 predictions of the different models. We see that *DISTRE's* top predictions encompass 10 distinct relation types, more than the other two models, with */location/location/contains* and */people/person/nationality* contributing 67% of the predictions. Compared to *PCNN+ATT* and *RESIDE*, *DISTRE* predicts additional relation types, such as */people/person/place_lived*, e.g., "Sen. <PER>, Republican/Democrat of <LOC>", and */location/neighborhood/neighborhood_of*, e.g., "the <LOC> neighborhood/area of <LOC>", with high confidence.

Relation	DISTRE	RESIDE	PCNN+ATT
location/contains	168	182	214
person/nationality	32	65	59
person/company	31	26	19
person/place_lived	22	–	–
country/capital	17	–	–
admin_div/country	13	12	6
neighborhood/neighborhood_of	10	3	2
location/team	3	–	–
company/founders	2	6	–
team/location	2	–	–
person/children	–	6	–

Table 4.3 Distribution over the top 300 predicted relations for each method. *DISTRE* achieves performance comparable to *RESIDE*, while predicting a more diverse set of relations with high confidence. *PCNN+ATT* shows a strong focus on two relations: */location/location/contains* and */people/person/nationality*.

RESIDE's top 300 predictions cover a smaller range of 7 distinct relation types, but also focus on */location/location/contains* and */people/person/nationality* (82% of the model's predictions). *RESIDE's* top predictions include, e.g., the additional relation types */business/company/founders*,

e.g., "<PER>, the founder of <ORG>", and */people/person/children*, e.g., "<PER>, the daughter/son of <PER>".

PCNN+ATT's high-confidence predictions are strongly biased towards a very small set of only four relation types. Of these, */location/location/contains* and */people/person/nationality* together make up 91% of the top 300 predictions. Manual inspection shows that for these relations, the *PCNN+ATT* model picks up on entity type signals and basic syntactic patterns, such as "<LOC>, <LOC>", e.g., "Berlin, Germany", and "<LOC> in <LOC>", e.g., "Green Mountain College in Vermont", for */location/location/contains*, and "<PER> of <LOC>", e.g., "Stephen Harper of Canada", for */people/person/nationality*. This suggests that the *PCNN* model ranks short and simple patterns higher than more complex patterns where the distance between the arguments is larger. The two other models, *RESIDE* and *DISTRE*, also identify and utilize these syntactic patterns.

Table 4.4 also lists some of the more challenging sentence level predictions that *DISTRE* correctly classified. These example require a model to correctly identify the entity type of each mention, resolve long range dependencies (e.g., in the third example), and precise syntactic patterns (e.g., in the first and fourth example).

Sentence	Relation
Mr. Snow asked, referring to Ayatollah Ali Khamenei, Iran's supreme leader, and Mahmoud Ahmadinejad, Iran's president.	/people/person/nationality
In Oklahoma, the Democratic governor, Brad Henry, vetoed legislation Wednesday that would ban state facilities and workers from performing abortions except to save the life of the pregnant woman.	/people/person/place_lived
Jakarta also boasts of having one of the oldest golf courses in Asia, Rawamangun , also known as the Jakarta Golf Club.	/location/location/contains
Cities like New York grow in their unbuilding: demolition tends to precede development, most urgently and particularly in Lower Manhattan, where New York City began.	/location/location/contains

Table 4.4 Examples of challenging relation mentions. These examples benefit from the ability to capture more complex features. Relation arguments are underlined.

4.4 Discussion and Summary

This chapter investigated an extension of the previously developed sequential transfer learning method (TRE) to distantly supervised relation extraction. Similarly, it uses a pre-trained language model as the source of prior knowledge, which is transferred to the relation extraction task during fine-tuning. In contrast to previous state-of-the-art methods, e.g., RESIDE, which heavily utilize explicitly provided side information and linguistic features, the proposed approach only uses features implicitly captured in pre-trained language representations. This mitigates error propagation as pre-processing and other feature extraction systems can be omitted, and also allows for feature selection during fine-tuning instead of selecting them a priori. In experiments on a widely used distantly supervised relation extraction benchmark, I show that the proposed method exhibits lower precision for the 300 top ranked predictions but achieves a state-of-the-art AUC score and an overall more balanced performance, especially for higher recall values. Most importantly, the approach predicts a larger set of distinct relation types with high confidence among the top predictions. This is also confirmed in the manual evaluation, which uncovered that transfer learning helped in particular for infrequent observed relations, while previous methods only recognize a limited set of relations with simple patterns.

Similar to the supervised setting, the developed transfer learning method demonstrated strong empirical results. It, however, is still unclear what linguistic features neural relation extraction models encode. To gain further insights and ultimately improve performance, it is crucial to better understand this aspect. Further, it is pivotal to explore and characterize circumstances in which models fail in order to better understand a method's capabilities and to identify possible systematic issues.

I conclude that two options exist that could further improve supervised and distantly supervised relation extraction performance. The first option is to improve pre-training, i.e., to learn representations that capture more syntactic and semantic knowledge, which allows us to fine-tune models more efficiently to less supervised data. The second option is to gain further insights and develop a better understanding of encoded linguistic features, model errors, datasets, and annotations – which is what I investigate in the next two chapters.

Chapter 5

Analyzing Captured Linguistic Knowledge

5.1 Introduction

As I demonstrated in the previous two chapters, sequential transfer learning increases relation extraction performance and data efficiency in the supervised setting, and similarly improves performance in the distantly supervised setting. The good results suggest that pre-trained language representations capture linguistic properties useful for the task. With features provided implicitly through transfer learning, it, however, is unclear what these properties are; uncovering them could gain insights into model decisions, help with debugging model errors, and ultimately improve relation extraction performance even further.

In this section, I first discuss the challenges of analyzing neural network-based methods, particularly in neural relation extraction (Section 5.1.1), followed by a discussion of related work (Section 5.1.2). I then propose my approach based on probing tasks, or diagnostic classifiers, to uncover the linguistic knowledge encoded by neural relation extraction models (Section 5.2). In the following, I conduct an extensive evaluation of the linguistic knowledge encoded in a various relation extraction models trained on two benchmark datasets (Section 5.3). Finally, I conclude this chapter with a discussion of the proposed method and how it aligns with the thesis' objective (Section 5.4).

This chapter is mainly based on a previously published full paper (Alt et al., 2020a).

5.1.1 Problem Statement

Recently, neural networks have considerably improved performance on many NLP tasks, including relation extraction. Understanding how they work is desirable for multiple reasons: It allows us to identify areas for improvement, and understanding their decision process is crucial to ensure accountability, trust, and fairness, which is important in critical domains such as healthcare. Their nested non-linear structure, however, makes them highly non-transparent, i.e., no information is provided about how they arrive at their decision (Samek et al., 2017) – essentially making them black boxes that are difficult to understand by humans.

Problem: Limited understanding of neural relation extraction methods Earlier feature-based, or feature-rich, relation extraction methods are easier to understand in this regard, as they use morphological properties, lexical classes, syntactic categories, semantic relations, etc.[1] In contrast, it is harder to understand end-to-end trained neural relation extraction models that take word embeddings as input, which in itself are quite opaque, and processes them with multiple non-linear neural network layers of arbitrary architecture. Also, due to the recent success of un- or self-supervised pre-training, models became even more complex. It therefore would benefit further research to be able to analyze what (linguistic) properties of the input models encode, and consequently use for their decisions; how this is influenced by transfer learning, neural network architecture, and additional linguistic knowledge; and how this affects performance on the task.

Ideally, a method that addresses this problem should be applicable independently of the chosen relation extraction method, domain, and dataset. As these vary greatly depending on the use case and application.

5.1.2 Related Work

Prior work in neural relation extraction is heavily focused on improving its performance. While there exist ablation studies that aim to quantifying how explicitly provided input features impact overall performance (Zeng et al., 2014; Zhang and Wang, 2015), no prior work specifically focused on analyzing neural relation extraction models. Relation extraction has very little coverage when it comes to the understanding of methods compared to other areas of NLP, such as question answering, machine-translation, and natural language inference. I therefore

[1] Although, one may question how practical such an analysis is; considering, for example, explaining support vectors in high dimensional support vector machines.

focus on related work on analysis methods for other NLP tasks, in particular I focus on probing linguistic information (Section 2.5.1).

5.1.2.1 Probing Linguistic Information for Other NLP Tasks

Probing tasks (Section 2.5.1), or diagnostic classifiers (Adi et al., 2017; Shi et al., 2016), are a well established method to analyze the presence of specific information in latent representations, e.g., to analyze linguistic information captured by neural network-based models in machine-translation (Belinkov et al., 2017), language modeling (Giulianelli et al., 2018), and general sentence encoding (Conneau et al., 2018). Shi et al. (2016) use probing tasks to probe syntactic properties captured in encoders trained on neural machine translation. Adi et al. (2017) extend this concept of "auxiliary prediction tasks", proposing sentence length, word count, and word order tasks to probe general sentence encoders. Conneau et al. (2018) use probing tasks for ten linguistic properties, e.g., sentence length and dependency parse tree depth, and analyze a set of encoders pre-trained on neural machine translation and natural language inference that were fine-tuned to text classification. Their setup, however, is not applicable to relation extraction for two reasons: First, the probing tasks target linguistic properties specific to the respective task, which are either too generic or unsuited for relation extraction; second, relation extraction requires both an input sentence and corresponding entity mentions.

5.1.3 Contributions

To uncover the linguistic properties captured by neural relation extraction models, I propose a set of probing tasks that specifically target properties that are considered useful for the task. I then analyze the features encoded by a broad set of models of various neural architectures on two benchmark datasets. In more detail, the contributions are:

Probing tasks for neural relation extraction To reveal the linguistic knowledge that is captured and used for relation extraction, I develop a workflow based on probing tasks. Each task targets a linguistic property of the input, e.g., the entity type of the head relation argument, and estimates how well it is encoded in the final input representation on which a model's decision is based. In particular, I introduce 14 probing tasks that specifically target linguistic properties relevant to relation extraction. The tasks cover syntactic, semantic, as well as surface features of the input texts. An extensive evaluation

on two benchmark datasets and more than 40 different models finds that the bias induced by the neural network architecture and the inclusion of additional linguistic knowledge are clearly expressed in the probing task performance.

Software libraries for relation extraction and probing tasks To facilitate future research and development of probing tasks, I also introduce two modular and extensible software libraries: *RelEx*,[2] a comprehensive suite of state-of-the-art neural relation extraction methods; and *REval*,[3] a framework to develop and evaluate probing tasks that extends the widely used SentEval toolkit (Conneau and Kiela, 2018).

Discussion of captured linguistic information In addition to the overall probing task performance, I discuss how the bias induced by different neural network architectures, e.g., convolution, recurrence, and self-attention, affects the encoded linguistic information. Also, I examine how additionally provided linguistic knowledge influences the captured information. This includes explicit semantic and syntactic knowledge, e.g., entity type or grammatical role, and implicit knowledge contained in contextualized word representations, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019).

5.2 Linguistic Probing Tasks for Neural Relation Extraction

This section introduces the tasks I use to probe relation extraction models for linguistic features. For each probing task, I train a classifier to predict a specific linguistic property of the input texts, e.g., the entity type of a relation argument, given the final encoder representations of a trained neural network-based relation extraction model (Section 5.3.1.1). The classifier performance then indicates how well the property of interest is encoded. Although I focus on the supervised relation extraction, this setup is applicable to the distantly supervised setting as well. Relation extraction literature is rich with information about useful linguistic features (Mintz et al., 2009; Surdeanu et al., 2011; Zhou et al., 2005), which I use as a starting point to develop suitable tasks. The features range from simple surface phenomena, e.g., relation argument distance; to syntactic information, e.g., parse tree depth and argument ordering; and semantic information, e.g., the entity types of relation arguments. I create the probing task data based on SemEval 2010 Task8 and TACRED (Section 2.3.4.2); with the standard training, validation, and test splits for TACRED, and the standard SemEval test split. As SemEval lacks a validation split, I randomly select 10% of the training split for this purpose. I use the named

[2] https://github.com/dfki-nlp/RelEx
[3] https://github.com/dfki-nlp/REval

entity, part-of-speech, and dependency parsing information provided in TACRED and parse SemEval with the Stanford Parser (Manning et al., 2014) in version 2018-10-05.

5.2.1 Surface Properties

These tasks test whether a model captures simple surface properties of input sentences it encodes. The sentence length (*SentLen*) task, proposed by Adi et al. (2017), predicts the number of tokens in a sentence. I group sentences into $n = 10$ bins (TACRED, 7 bins for SemEval) by length, selecting bin widths so that training sentences are distributed approximately uniformly across bins, and treat *SentLen* as an n-way classification task. The next probing task, argument distance (*ArgDist*), predicts the number of tokens between the two relation arguments. Similar to *SentLen*, I group sentences into 10 bins (5 for SemEval) by relative distance. Inspired by a common feature in classical relation extraction (Surdeanu et al., 2011), I also test if any named entity exists between the two relation arguments (*EntExist*), treating it as a binary classification problem. Addressing this task requires the encoder to produce a sentence embedding that (at least partially) represents the inner context of the relation arguments.

5.2.2 Syntactic Properties

Syntactic information is highly relevant for relation extraction. Many methods utilize dependency path information (Bunescu and Mooney, 2005a; Krause et al., 2012; Mintz et al., 2009), or part-of-speech tags (Surdeanu et al., 2011; Zhou et al., 2005). I therefore include the tree depth task (*TreeDepth*) described by Conneau et al. (2018). This task tests whether an encoder can group sentences by the depth of the longest path from root to any leaf. I group tree depth values into 10 (TACRED, SemEval 7) approximately uniformly distributed classes, ranging from depth 1 to depth 15. To account for shortest dependency path (SDP) information, I include an SDP tree depth task (*SDPTreeDepth*), which tests if a model encodes information about the syntactical link between the relation arguments. Again, I group SDP tree depth values into bins, in this case only 6 (TACRED, SemEval 4) classes, since the SDP trees are generally more shallow than the original sentence dependency parse tree. The argument ordering task (*ArgOrd*) tests if the head argument of a relation occurs before the tail argument in the token sequence. A model that successfully addresses this challenge captures some information about syntactic structures where the order of a relation's arguments is inverted, e.g., in constructs such as "The acquisition of Monsanto by Bayer", as compared to default constructions like "Bayer

acquired Monsanto". This is useful to determine whether the representation can for example be used to distinguish between active and passive constructions, where relation arguments may be reversed. I also include 4 tasks that test for the part-of-speech tag of the token directly to the left or right of the relation's arguments: *PosHeadL*, *PosHeadR*, *PosTailL*, *PosTailR*. These tasks test whether the encoder is sensitive to the immediate context of an argument. Some relation types, e.g., *per:nationality* or *org:top_member*, can often be identified based on the immediate argument context, e.g., "US *president-NN* Donald Trump", or "Google *'s-POSS CEO-NN* Larry Page". Capturing this type of information should be beneficial to the relation classification.

5.2.3 Semantic Properties

Finally, I include tasks that target the understanding of what each argument denotes. The argument entity type tasks (*TypeHead*, *TypeTail*) ask for the entity tag of the head and tail argument, respectively. Entity type information is highly relevant for relation extraction, since it strongly constrains the set of possible relation labels for a given argument pair. I treat these tasks as multi-class classification problems over the set of possible argument entity tags (Section 2.3.4.2).

The final tasks concern the grammatical function of relation arguments. The grammatical role tasks (*GRHead*, *GRTail*) ask for the role of each argument, as given by the dependency label connecting the argument and its syntactic head token. The motivation is that the subject and object of verbal constructions often correspond to relation arguments for some relation types, e.g., "Bayer acquired Monsanto". In this scenario I test for four roles, namely *nsubj*, *nsubjpass*, *dobj* and *iobj*, and group all other dependency labels into an *other* class. Note that there are other grammatical relations that may be of interest for relation extraction, for example possessive modifiers ("Google's Larry Page"), compounds ("Google CEO Larry Page"), and appositions ("Larry Page, CEO of Google").[4]

[4] I exclude all sentences with multi-token entities where the dependency parser failed to create a single directly connected sub-tree for the entity tokens, with the entity's head as the root of the sub-tree.

5.3 Evaluation

The goal of the evaluation is to reveal what linguistic properties of the input text neural relation extraction models capture. First, I create a set of relation extraction models based on two benchmark datasets. I then use the proposed probing tasks to determine how well the properties of interest are encoded, and follow up on the results with a detailed discussion of factors that might affect the encoded features, such as neural network architecture and supporting linguistic knowledge. Also, I discuss how probing task performance relates to relation extraction performance. This section first outlines the evaluation setup (Section 5.3.1), followed by a detailed discussion of the results (Section 5.3.2).

5.3.1 Experimental Setup

Figure 5.1 gives an overview of the probing task setup I use to determine the captured linguistic properties. First, I train a neural relation extraction model, optionally with supporting linguistic knowledge, on supervised data. Then, I freeze the sentence encoder of this model and obtain the representations $\mathbf{s}_1, \ldots, \mathbf{s}_N$ for each example $\{(t_1^i, \ldots, t_n^i, head_i, tail_i)\}_{i=1,\ldots,N}$, where $t_1^i, \ldots, t_n^i$ denotes the input token sequence and $\{head, tail\}_i$ are the spans of head and tail entity mention, respectively. Subsequently, I fit a logistic regression classifier to the probing task data $\{(\mathbf{s}_i, y_i)\}_{i=1,\ldots,N}$, where y_i is the probing target, e.g., the type of the head relation argument. The probing classifier performance indicates how well the sentence representations encode the probed information. I use SemEval 2010 Task 8 and TACRED (Section 2.3.4.2) as datasets for the experiments and report macro-averaged F1 scores for SemEval and micro-averaged F1 for TACRED.

5.3.1.1 Sentence Encoders

Typically, binary neural relation extraction methods follow a sequence to vector approach, as described in Section 2.3.3. The input text is first transformed into a fixed-size representation by a neural-network based (sentence) encoder, before applying a classification layer to predict the relation. An input is represented as a sequence of T tokens $t_1, \ldots, t_T$, and the corresponding spans of *head* and *tail* entity mention. In the experiments, I focus on four widely used neural network-based architectures and signal the position of *head* and *tail* with relative offsets to each token t_i as positional embeddings $\mathbf{p}_i^h \in \mathbb{R}^d$ and $\mathbf{p}_i^t \in \mathbb{R}^d$. The positional embeddings are

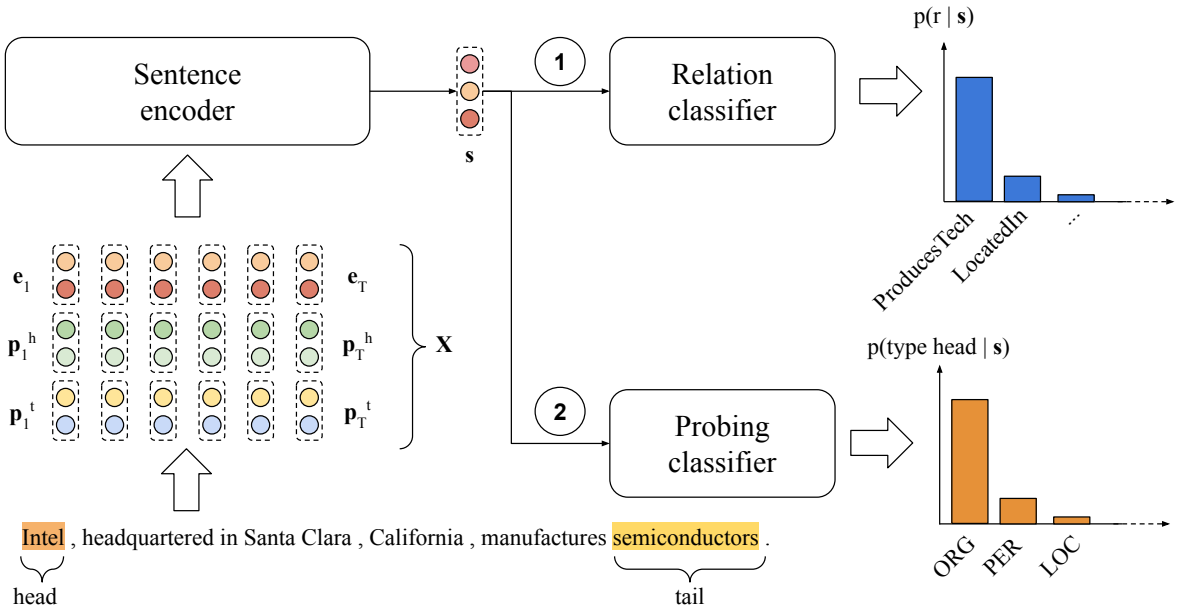


Figure 5.1 Probing task setup. In the first step, I train a neural relation extraction model (sentence encoder + relation classifier) from a dataset. Subsequently, I freeze the encoder and use the sentence representations $\mathbf{s}$ to train a classifier for a particular probing task. The probing task classifier performance indicates how well a property of interest, e.g., type of the head entity mention, is encoded in $\mathbf{s}$, which is also used to predict the relation.

concatenated to the token embedding $\mathbf{e}_i \in \mathbb{R}^c$ to form the input representation $\mathbf{x}_i = [\mathbf{e}_i, \mathbf{p}_i^h, \mathbf{p}_i^t]$ for each token t_i. In the following, I now describe the different sentence encoders.

CNN I follow the work of Zeng et al. (2014) and Nguyen and Grishman (2015), who both use a convolutional neural network for relation extraction. Their methods encode the input token sequence $t_1, \ldots, t_T$ by applying a series of 1-dimensional convolutions of different filter sizes, yielding a set of output feature maps $\mathbf{M}_f$, followed by a max-pooling operation that selects the maximum values along the temporal dimension of $\mathbf{M}_f$ to form a fixed-size representation.

Bi-LSTM Similar to Zhang and Wang (2015) and Zhang et al. (2017), I use a Bi-LSTM to encode the input sequence. A Bi-LSTM computes a sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_T$, where $\mathbf{h}_i$ is a concatenation $[\overrightarrow{\mathbf{h}_i}, \overleftarrow{\mathbf{h}_i}]$ of the hidden states of a forward LSTM $\overrightarrow{\mathbf{h}_i}$ and a backward LSTM $\overleftarrow{\mathbf{h}_i}$. Similar to the CNN, I use max pooling across the temporal dimension to obtain a fixed-size representation[5].

[5] I considered taking the final hidden state of both directions but found max pooling to perform superior.

GCN Graph convolutional networks (GCN, Kipf and Welling, 2017) adapt convolutional neural networks to graphs. Following the approach of Zhang et al. (2018b), I consider the input token sequence $t_1, \ldots, t_T$ as a graph with T nodes, with an edge between t_i and t_j if there exists a dependency link between the two tokens. I convert the dependency parse into a $T \times T$ adjacency matrix after pruning the graph to the shortest dependency path between *head* and *tail* entity mention. An L-layer GCN applied to the input tokens yields a sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_T$ contextualized on neighboring tokens with a graph distance of at most L. Forming a fixed size representation is done by max pooling over the temporal dimension and local max pooling over the tokens $\{t_i\}$, for $i \in (head_{start}, \ldots, head_{end})$ and similar for $i \in (tail_{start}, \ldots, tail_{end})$.

Self-attention Similar to the transformer (Vaswani et al., 2017), I compute a sequence of contextualized representations $\mathbf{h}_1, \ldots, \mathbf{h}_T$ by applying L layers of multi-head self-attention to the input tokens $t_1, \ldots, t_T$. The representation h_t of t_i is computed as a weighted sum of a projection V of the input tokens, with respect to the scaled, normalized dot product of Q and K, which are also both linear projections of the input with the procedure repeated for each attention head. A fixed-size representation is obtained by taking the final state h_T^L at the L-th layer.

5.3.1.2 Supporting Linguistic Knowledge

Zeng et al. (2014); Zhang et al. (2017, 2018b) show that explicitly adding additional lexical, syntactic, and semantic input features to neural relation extraction models considerably improves performance. Features include, for example, casing, named entity, part-of-speech, and dependency information. Pre-trained language representations, as previously discussed, are claimed to implicitly capture syntactic and semantic information useful to a wide range of downstream tasks (Devlin et al., 2019; Peters et al., 2018; Radford et al., 2018). I therefore evaluate how adding explicit named entity and grammatical role information through entity masking affects the linguistic features captured by the models, and compare it to adding contextualized word representations computed by ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) as additional input features.

Entity Masking Zhang et al. (2017) show that entity masking provides a significant performance gain on the TACRED. It replaces each mention with a combination of its entity type and

grammatical role (subject and object). It limits the information about mentions available to a model, possibly preventing overfitting to specific entity mentions and also forcing the model to focus more on the context.

ELMo Peters et al. (2018) introduced **E**mbeddings from **L**anguage **Mo**dels, an approach to compute contextualized word representations by applying a pre-trained, two-layer Bi-LSTM to an input token sequence $t_1, \ldots, t_T$. ELMo operates on a character level and is pre-trained with the forward and backward direction as a separate unidirectional language model. It computes a representation $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}, \overleftarrow{\mathbf{h}_i}]$ for each token t_i, with $\overrightarrow{\mathbf{h}_i}$ conditioned on the preceding context $t_1, \ldots, t_{i-1}$ and independently $\overleftarrow{\mathbf{h}_i}$, conditioned on the succeeding context $t_{i+1}, \ldots, t_T$.

BERT **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (Devlin et al., 2019) improves upon methods such as ELMo and the OpenAI Generative Pre-trained Transformer (GPT) (Radford et al., 2018) by using a masked language model (Section 2.4.3.1) that allows for jointly training forward and backward directions. Compared to ELMo, BERT operates on word-piece input and is based on the transformer architecture. It computes a representation for a token t_i jointly conditioned on the preceding context $t_1, \ldots, t_{i-1}$ and succeeding context $t_{i+1}, \ldots, t_T$.

5.3.2 Results

Table 5.1 and Table 5.2 report the accuracy scores of the probing task experiments for models trained on the TACRED and SemEval benchmarks. I exclude the *ArgOrd* and *EntExists* task in the SemEval evaluation, since relation arguments are always ordered in the sentence as indicated by the relation type, and entity types recognizable by standard tools such as Stanford CoreNLP that might occur between head and tail are irrelevant to the dataset's entity types and relations.

Baseline performances are reported in the top section of Table 5.1 and Table 5.2. *Length* and *ArgDist* are both linear classifiers which use sentence length as the only feature, and distance between head and tail argument, respectively. Bag-of-embeddings (*BoE*) computes a representation of the input by taking the sum over its token embeddings. Generally, there is a large gap between top baseline performance and that of a trained encoder. While *SentLength* and *ArgDist* are trivially solved by the respective linear classifier, *BoE* shows surprisingly good

	Type Head	Type Tail	Sent Len	Arg Dist	Arg Ord	Ent Exist	PosL Head	PosR Head	PosL Tail	PosR Tail	Tree Dep	SDP Dep	GR Head	GR Tail	F1 score
Majority	66.4	33.5	14.5	14.8	54.7	51.0	22.8	23.0	26.9	20.0	23.7	28.4	58.4	75.2	-
Length	66.4	33.5	**100**	13.8	54.8	59.4	18.6	24.7	26.9	20.1	**30.5**	29.6	58.4	75.2	-
ArgDist	66.4	33.5	16.5	**100**	54.7	77.5	14.9	23.0	26.9	19.8	23.8	35.3	58.4	75.2	-
BoE	77.7	47.6	61.1	22.6	97.3	66.5	33.7	41.5	32.5	36.3	29.8	31.0	66.3	77.4	39.4
CNN	94.0	85.8	47.6	88.1	98.8	**84.5**	70.7	76.1	84.0	86.5	28.5	44.0	78.0	88.6	55.9
ELMo	**97.0**	**90.2**	48.7	91.7	99.1	84.3	**76.1**	**81.2**	**86.6**	**90.1**	28.3	45.0	**82.8**	**91.9**	58.8
BERT ↓	95.9	88.8	44.7	46.0	93.8	79.9	64.7	74.4	80.8	88.4	29.4	41.0	77.7	90.0	59.7
+ BERT ↑	96.1	88.8	48.0	43.7	91.9	80.0	56.9	70.3	80.1	87.5	28.0	41.3	75.0	89.6	61.0
CNN ⊗	84.2	60.9	46.4	58.3	94.3	81.5	44.3	50.9	54.4	63.9	27.7	40.0	68.5	82.0	59.5
+ ELMo	82.8	69.8	47.4	75.6	98.1	82.9	54.2	60.2	65.4	77.3	28.7	42.4	71.9	85.0	61.7
+ BERT ↓	87.6	80.3	50.9	29.3	83.2	72.4	39.3	46.1	67.7	80.7	30.1	36.9	67.1	87.4	65.3
+ BERT ↑	87.2	79.3	50.6	25.3	78.3	69.8	39.6	42.9	59.9	77.5	30.3	35.1	65.6	86.9	66.1
Bi-LSTM	93.4	81.2	42.0	47.9	**99.4**	79.2	41.2	50.8	50.6	68.4	28.7	41.7	69.3	85.2	55.3
+ ELMo	96.4	89.6	27.9	47.0	97.9	80.9	47.8	52.5	67.2	72.6	25.2	42.8	72.1	90.0	61.8
+ BERT ↓	96.0	87.3	31.0	45.5	99.1	78.8	46.1	55.6	61.7	71.3	26.6	42.7	72.2	87.7	62.5
+ BERT ↑	96.0	87.7	28.6	45.3	97.7	80.4	48.0	50.9	61.4	67.4	25.1	42.3	70.8	87.0	63.1
Bi-LSTM ⊗	81.9	71.4	27.6	35.6	90.6	73.2	36.1	40.5	59.3	66.4	25.7	38.4	64.6	85.3	62.9
+ ELMo	82.8	50.7	30.6	19.7	73.4	65.0	32.0	35.9	37.9	41.8	28.0	32.2	63.0	79.5	64.1
+ BERT ↓	82.3	77.9	34.1	25.6	87.6	68.4	32.5	36.7	61.5	64.7	27.6	35.1	66.6	86.0	65.4
+ BERT ↑	81.7	79.6	30.2	21.3	81.1	67.0	30.6	33.8	55.9	55.1	27.3	34.2	64.1	84.9	66.1
GCN	93.0	81.9	18.8	35.5	86.0	74.4	48.6	48.8	51.2	52.3	24.0	**49.9**	74.2	85.9	57.4
+ ELMo	96.3	86.2	18.7	29.3	77.5	74.0	50.4	52.0	48.9	51.7	23.2	47.4	77.1	86.9	62.1
+ BERT ↓	96.0	85.2	20.7	31.2	83.6	74.2	48.6	52.4	47.4	50.4	23.9	48.7	74.4	85.3	62.9
+ BERT ↑	96.3	85.7	21.4	32.9	84.3	75.3	50.1	54.6	48.6	52.5	24.5	49.2	76.3	85.8	61.5
GCN ⊗	87.6	67.4	18.1	33.1	81.6	72.8	36.8	51.1	44.8	48.8	24.1	47.3	73.2	83.0	63.7
+ ELMo	92.7	68.6	18.6	26.4	76.8	71.4	41.9	50.4	43.6	45.1	23.8	47.1	76.3	83.9	65.4
+ BERT ↓	93.5	71.5	22.0	33.3	88.5	73.8	44.9	50.6	44.7	47.7	24.4	49.1	72.6	82.3	66.3
+ BERT ↑	93.4	72.0	23.7	33.2	90.4	73.9	42.8	50.1	44.0	48.3	24.9	48.0	72.9	83.0	65.9
S-Att.	89.9	81.8	22.7	32.8	75.7	78.1	34.1	38.9	40.8	44.8	26.1	38.2	60.7	81.1	57.6
+ ELMo	96.6	87.8	24.9	30.6	74.1	79.1	36.0	41.4	39.2	44.1	26.4	37.9	64.1	83.4	64.7
+ BERT ↓	96.2	87.0	25.9	31.4	75.6	76.5	35.3	40.8	39.8	44.4	25.4	39.1	61.8	81.3	63.9
+ BERT ↑	96.5	87.3	26.1	32.6	76.8	78.0	34.7	40.9	40.0	44.0	25.7	38.1	62.2	81.7	63.8
S-Att. ⊗	79.5	56.5	29.0	44.3	91.2	79.5	29.6	43.0	36.1	60.3	26.1	39.6	64.7	79.5	65.9
+ ELMo	78.2	44.4	25.1	31.5	72.3	77.1	31.6	37.5	34.4	34.8	26.2	36.7	62.1	75.9	66.6
+ BERT ↓	82.4	66.9	36.2	33.2	74.9	76.8	32.0	37.6	38.0	41.3	27.4	37.6	63.0	79.8	66.7
+ BERT ↑	80.0	69.0	31.9	32.8	78.6	76.6	30.3	34.2	37.5	39.2	27.0	38.2	60.4	79.9	**66.9**

Table 5.1 TACRED probing task accuracies and model F1 scores on the test set. ↑ and ↓ indicate the cased and uncased version of BERT, ⊗ denotes models with entity masking.

performance on *SentLen* and *ArgOrd*, and a clear improvement over the other baselines for named entity- and part-of-speech-related probing tasks.

	Type Head	Type Tail	Sent Len	Arg Dist	PosL Head	PosR Head	PosL Tail	PosR Tail	Tree Dep	SDP Dep	GR Head	GR Tail	F1 score
Majority	22.0	21.3	25.7	42.1	62.1	39.3	38.3	34.0	25.4	67.2	37.3	80.9	-
Length	25.8	24.7	**100**	42.1	62.1	39.1	38.3	46.3	**44.3**	67.2	40.6	80.9	-
ArgDist	23.6	22.3	25.7	**100**	62.1	43.7	37.9	35.3	26.2	67.8	45.4	80.9	-
BoE	58.5	58.0	82.4	84.8	65.1	66.1	49.2	72.5	44.1	69.8	65.4	83.6	55.7
CNN	76.1	76.2	34.9	87.5	66.0	85.8	74.2	73.1	34.1	72.1	70.3	89.1	80.2
+ ELMo	81.3	81.8	38.1	88.5	70.0	89.0	79.5	76.4	35.5	71.8	75.1	90.9	84.4
+ BERT ↓	**83.9**	**84.1**	55.9	90.2	74.0	89.3	81.2	**84.6**	41.3	73.1	76.8	90.6	**86.3**
+ BERT ↑	83.4	83.7	54.3	90.4	**74.4**	**89.4**	**82.0**	82.8	42.0	73.0	78.3	90.8	86.0
Bi-LSTM	77.1	77.0	50.5	74.9	63.8	75.9	61.8	68.5	41.3	70.3	69.2	87.7	80.1
+ ELMo	81.5	81.8	41.1	66.6	62.8	71.8	59.3	64.5	37.5	70.1	70.0	87.6	83.7
+ BERT ↓	83.6	83.7	41.8	61.5	62.7	68.9	57.9	63.0	37.1	70.8	67.4	86.7	85.6
+ BERT ↑	82.5	82.8	41.8	66.0	63.1	70.8	58.6	64.3	37.7	71.0	68.9	87.5	85.1
GCN	75.4	75.5	35.0	81.5	68.5	87.5	71.2	55.5	35.5	**80.3**	76.3	91.7	79.6
+ ELMo	80.7	80.8	32.2	68.1	68.3	83.4	65.8	53.2	34.4	75.8	80.0	91.1	84.2
+ BERT ↓	82.5	83.0	42.5	66.5	73.6	84.7	69.2	66.3	38.9	77.2	82.1	91.0	85.7
+ BERT ↑	81.5	81.9	42.7	67.3	73.8	85.1	69.6	67.8	39.6	77.6	**84.2**	**91.9**	84.3
S-Att.	77.4	77.6	34.2	50.0	62.1	56.2	49.8	47.1	35.9	67.9	54.2	84.1	80.2
+ ELMo	80.7	81.3	33.1	46.2	62.0	53.9	49.1	45.7	34.7	68.1	54.9	84.4	83.6
+ BERT ↓	83.4	83.3	31.0	45.3	62.1	51.8	48.4	44.7	33.0	67.8	53.3	83.6	85.6
+ BERT ↑	82.8	82.8	30.6	46.1	62.1	52.7	48.2	44.4	33.6	67.9	54.6	84.1	84.9

Table 5.2 SemEval probing task accuracies and model F1 scores on the test set. ↑ and ↓ indicate the cased and uncased version of BERT.

Relation extraction performance The relation extraction performance on TACRED ranges between 55.3 (Bi-LSTM) and 57.6 F1 (S-Att.), with performance improving to around 58.8 - 64.7 F1 when adding pre-learned, contextualized word representations. As observed in previous work (Zhang et al., 2017), masking helps the encoders to generalize better, with gains of around 4 - 8 F1 when compared to the vanilla models. This is mainly due to better recall, which indicates that without masking, models may overfit, for example, by memorizing specific entity names. The best performing model achieves a score of 66.9 F1 (S-Att. + BERT cased and masking). On SemEval performance of the vanilla models is around 80.0 F1. Adding contextualized word representations significantly improves the performance of all models by 3.5 - 6 F1. The best-performing model on this dataset is a CNN with uncased BERT embeddings with an F1-score of 86.3.

Encoder architecture For most probing tasks, except *SentLen* and *ArgOrd*, a proper encoder clearly outperforms *BoE*, which is coherent with the findings of Adi et al. (2017) and Conneau et al. (2018). Similarly, the results indicate that the prior imposed by the encoder architecture

preconditions the information encoded in the learned embeddings. Models with a local or recency bias (CNN, Bi-LSTM) perform better on probing tasks with local focus, such as *PosHead{L,R}* and *PosTail{L,R}*, and partially on distance related tasks (*ArgDist*, *ArgOrd*). Similar, models with access to dependency information (GCN) perform well on tree related tasks (*SDPTreeDepth*). Due to the graph pruning step the GCN is left with a limited view of the dependency tree, which explains the low performance on *TreeDepth*. Surprisingly, while self-attention exhibits superior relation extraction performance, it consistently performs lower on the probing tasks compared to the other encoding architectures. This could indicate that self-attention encodes deeper, more abstract, linguistic features into the sentence representation, which are not covered by the current set of probing tasks.

Probing task performance Compared to the baselines, all proper encoders exhibit consistently high performance on *TypeHead* and *TypeTail*, clearly highlighting the importance of entity type information to relation extraction. In contrast, these encoders perform worse on *SentLen*, which intuitively makes sense, since sentence length is mostly irrelevant for relation extraction. This is consistent with Conneau et al. (2018), who found *SentLen* performance to decrease for models trained on more complex downstream tasks, e.g., neural machine translation, strengthen the assumption that, as a model captures deeper linguistic properties it will tend to forget about this superficial feature. With the exception of the CNN, all encoders consistently show low performance on the argument distance (*ArgDist*) task. The same can be observed for *ArgOrd*, where models that are biased towards locality (CNN and Bi-LSTM) perform better, while models that are able to efficiently model long range dependencies, such as GCN and self-attention, show lower performance. The superior relation extraction performance of the latter indicates that their bias may allow them to learn more complex linguistic features.

The balanced performance of CNN, Bi-LSTM and GCN encoders across part-of-speech related tasks (*PosHeadL*, *PosHeadR*, *PosTailL*, *PosTailR*) highlights the importance of part-of-speech-related features to relation extraction, again with the exception of self-attention, which performs just slightly above baselines. On *TreeDepth* and *SDPTreeDepth* (with GCN as the exception), average performance in many cases ranges just slightly above baseline performance, suggesting that *TreeDepth* requires more nuanced syntactic information, which the models fail to acquire. The good performance on grammatical role tasks (*GRHead*, *GRTail*) once more emphasizes the relevance of this feature to relation extraction, with the GCN exhibiting the best performance on average. This is unsurprising, because the GCN focuses on token-level information along the dependency path connecting the arguments, and hence seems to be able to capture grammatical

relations among tokens more readily than the other encoders – even though the GCN also does not have access to the dependency labels themselves.

Entity masking Perhaps most interestingly, masking entity mentions with their respective named entity and grammatical role information considerably lowers the performance on entity type related tasks (*TypeHead* and *TypeTail*). This indicates that masking forces the encoder's focus away from the entity mentions, which is confirmed by the performance decrease in probing tasks with a focus on argument position and distance, e.g., *ArgDist*, *ArgOrd*, and *SentLen*. CNN and Bi-LSTM encoders exhibit the greatest decrease in performance, suggesting a severe overfitting to specific entity mentions when no masking is applied. In comparison, the GCN shows less tendency to overfit. Surprisingly, with entity masking the self-attentive encoder (S-Att.) increases its focus on entity mentions and their surroundings as suggested by the performance increase on the distance and argument related probing tasks.

Word representations Adding contextualized word representations computed by ELMo and BERT greatly increases performance on probing tasks with a focus on named entity and part-of-speech information. This indicates that contextualized word representations encode useful syntactic and semantic features relevant to relation extraction. The improved performance on syntactic and semantic abilities is also reflected in an overall increase in relation extraction performance. Compared to ELMo, encoders with BERT generally exhibit an overall better and more balanced performance on the probing tasks. This is also reflected in a superior relation extraction performance, suggesting that a bidirectional language model encodes linguistic properties of the input more effectively. Somewhat surprisingly, BERT without casing performs equally or better on the probing tasks focused on entity and part-of-speech information, compared to the cased version. While this intuitively makes sense for SemEval, as the dataset focuses on semantic relations between concepts, it is surprising for TACRED, which contains relations between proper entities, e.g., person and company names, with casing information more important to identify the entity type.

Probing task vs. relation extraction performance One interesting observation is that better performance on probing tasks not necessarily implies better relation performance. For example, CNN + ELMo scores highest for most of the probing tasks, but has an 8.1 lower F1 score than the best model on this dataset, S-Att. + BERT cased with entity masking. Similarly, all variants of the self-attentive encoder (S-Att.) show superior relation extraction performance

but consistently come up last on the probing tasks, occasionally performing just above the baselines.

5.4 Discussion and Summary

In this chapter, I proposed a set of probing tasks to uncover the linguistic features captured by neural relation extraction models. The tasks cover a wide range of surface, syntactic, and semantic properties of input texts. I conducted an extensive evaluation of more than 40 neural relation extraction models, and studied the effect of explicitly and implicitly provided linguistic knowledge, uncovering interesting properties about the architecture and input features. For example, I found self-attentive encoders to be well suited for relation extraction on sentences of different complexity, though they consistently perform lower on probing tasks; hinting that these architectures capture more complex linguistic features. I also showed that the bias induced by different architectures clearly affects the learned properties, as suggested by probing task performance, e.g., for distance and dependency related probing tasks. To facilitate further research, I released two software libraries that implement state-of-the-art relation extraction methods and simplify future development of probing tasks.

Although the evaluation uncovered that architecture and supporting linguistic knowledge affect captured features, I found two aspects merit further investigation. First, I observed that probing task performance does not correlate with relation extraction performance. For instance, the self-attentive models do not capture any of the probed features well but perform superior on relation extraction, which suggests that this architecture allows the model to capture different and possibly more complex linguistic features that are not covered by the current set of tasks. Second, probing tasks only target model internals, i.e., representations, but do not consider their predictions, specifically their errors, and the dataset. This makes it difficult to identify systematic errors and their sources.

I conclude that probing tasks should cover more properties and specific linguistic patterns such as appositions, and it would be of interest to also investigate a model's ability of generalizing to certain entity types, e.g., company and person names. Further, I conclude that a proper model analysis should be complemented by an evaluation focused on model predictions and dataset; with the goal to uncover systematic model and dataset errors. I investigate this in more detail in the next chapter.

Chapter 6

Fine-Grained Analysis of Model Errors and Datasets

6.1 Introduction

The research presented in the previous chapter found that model analysis via probing tasks uncovers linguistic features encoded by neural relation extraction methods, and also showed how these features are affected by neural network architecture and supporting linguistic information. The evaluation via probing tasks, however, neglects circumstances where models err. Understanding these circumstances is crucial to improve their performance, but can be difficult because errors may be caused for a variety of reasons, e.g., biases in the dataset, annotation errors, or insufficient modeling capabilities. A thorough evaluation, therefore, should be complemented by an analysis targeting dataset, annotations, and model predictions – in particular incorrect ones.

This section first discusses the challenges of model error analysis and dataset evaluation (Section 6.1.1). In the following, I look at previous works in neural language processing that address these challenges via data grouping and counterfactual analysis (Section 6.1.2). I then propose a principled approach for fine-grained analysis of relation extraction methods that consists of four manual and automated steps (Section 6.2). To demonstrate its effectiveness, I conduct an extensive evaluation of TACRED (Zhang et al., 2017), one of the largest and most widely used crowdsourced relation extraction benchmarks (Section 6.3). Finally, I conclude

this chapter with a discussion of the proposed analysis approach and its alignment with the objective of this thesis (Section 6.4).

This chapter is based on a previously published full paper (Alt et al., 2020b), but expands on the description and discussion of the proposed approach.*

6.1.1 Problem Statement

As previously shown, neural network-based methods considerably improved relation extraction performance. Despite the recent improvements, their performance is still far below human level, as suggested by the state-of-the-art F1 score of 71.5 on TACRED, one of the largest and most widely used benchmarks. This is a common situation, and naturally we ask if it is possible to identify the underlying factors that contribute to the error rate of approximately 30%. In such cases, understanding where models fail is crucial to revise them, uncover deficiencies, and ultimately improve performance.

Problem: Focus on a single metric and model errors in isolation Typically, we use a single metric to quantify the performance of a relation extraction method, e.g., precision, recall or F1 score; either per relation or over the whole dataset. A single number, however, provides us with no insights into what is causing the errors. For example, the dataset may contain biases or annotation errors, a problem frequently observed with datasets created by crowdsourcing (Geva et al., 2019). Instead, we could inspect incorrect model predictions directly to build hypotheses of what is causing the errors. This may not reveal the underlying cause either, because we are unable to verify our hypotheses. In the worst case, we focus on cases that are actually well handled on average when looking at all predictions, not just the errors (Rondeau and Hazen, 2018). It would benefit the development of future relation extraction methods, if we were able to conduct a more systematic and fine-grained analysis of methods with a focus on detailed aspects of models, datasets, and annotations, instead of a single metric.

A method that addresses these limitations should be broadly applicable, independently of relation extraction method, domain, and dataset.

*As stated in Section 1.3, this chapter contains results contributed by my co-author Aleksandra Gabryszak, specifically Section 6.3.2 and Section 6.3.3.1. They, however, are necessary to describe and demonstrate the overall approach, and subsequent analyses build on those findings.

6.1.2 Related Work

I first review earlier works that analyze neural language processing model errors, e.g., by data grouping and counterfactual analysis (Section 6.1.2.1). In the following, I provide an overview of prior work on dataset evaluation (Section 6.1.2.2). However, no prior work specifically focused on analyzing model errors in neural relation extraction. I therefore focus on related work from other NLP tasks.

6.1.2.1 Analysis of neural language processing models

Analysis methods for neural language processing models include occlusion- and gradient-based approaches. Occlusion measures the relevance of an input feature, e.g., a token, by replacing or removing it and observing the change in model prediction (Zintgraf et al., 2017). Gradient-based methods use the gradient to quantify the relevance of input features to the output (Harbecke et al., 2018). As previously discussed, probing tasks (Conneau et al., 2018; Kim et al., 2019) can be used to probe the presence of specific properties, e.g., in neural network hidden states. None of these, however, are suited to directly analyze model errors. A typical approach aimed at model error analysis is data grouping, i.e., to group examples based on some property and aggregate metrics for a particular slice, e.g., accuracy over question types in machine comprehension (He et al., 2017) or per label performance in semantic role labeling (Liu et al., 2018c).

Adversarial examples are another approach to analyse models. For example, Jia and Liang (2017) add distracting sentences to text passages used for machine comprehension. Ribeiro et al. (2018) use rewrite rules to alter the input while preserving its semantics to cause the model to change its prediction. While the previous two works use adversarial examples to evaluate model robustness, Wu et al. (2019) instead combine them with data grouping (Section 2.5.2) to explicitly formalize and verify model error hypotheses in extractive question answering. They first group examples according to a property that represents an error hypothesis, for example, an entity in the passage with the same type as the answer tricks the model into making an incorrect prediction. In a second step, counterfactual analysis is used to verify if the hypothesis holds, i.e., creating adversarial examples (Section 2.5.2) by removing the distracting entities from the passages and observing whether the model's predictions change.

6.1.2.2 Dataset evaluation

Typically, dataset evaluation aims to uncover open challenges and artefacts, or biases, which can be exploited by models to predict the correct solution despite lacking the necessary capabilities to solve the task. Biases frequently occur in datasets created via heuristics or crowdsourcing. For example, Chen et al. (2016) evaluate the heuristically supervised CNN/Daily Mail reading comprehension task and show that a carefully designed logistic regression classifier can outperform more complex deep neural network-based state-of-the-art methods. They further inspect 100 randomly selected examples and group them based on heuristics a model could use to answer a question. The authors, for instance, find that many of the examples could be easily answered by just considering the word overlap between question and passage. Barnes et al. (2019) identify remaining challenges in sentiment analysis by collecting model misclassifications on six different datasets and annotating them for 18 linguistic phenomena. The authors find "mixed polarity" to be the most challenging, i.e., sentences where two differing polarities are expressed, either towards two separate entities, or towards the same entity.

Other works also explore bias in datasets and the adoption of shallow heuristics on biased datasets. Niven and Kao (2019) show that near human level performance of models in argument reasoning comprehension is due to spurious statistical cues in the dataset. For example, the authors find that choosing the answer containing the word "not" leads to the correct solution in 61% of the cases. McCoy et al. (2019) hypothesize that three heuristics may be exploited by models for natural language inference. To verify their hypotheses, they create a controlled evaluation set similar to the challenge sets described in Section 2.5.3.

6.1.3 Contributions

Fine-grained model analysis should focus on incorrect predictions and also consider the dataset and its annotations. I therefore propose a principled approach to automatically select the most challenging examples in a dataset for manual evaluation, followed by an automated analysis of developed error hypotheses. I demonstrate its effectiveness by analyzing the widely used TACRED relation extraction benchmark, its annotations, and errors exhibited by state-of-the-art models. In more detail, the contributions are:

Fine-grained evaluation of relation extraction methods and datasets I propose a method that allows for a fine-grained and effective evaluation of relation extraction datasets and

model errors. First, the most challenging examples in a dataset are automatically selected for manual evaluation based on incorrect predictions of a larger model set. The goal of the manual evaluation is to identify possible error hypotheses, including labeling errors. In the following, the hypotheses are automatically verified by using fine-grained data grouping and counterfactual analysis. An extensive evaluation of the widely used TACRED benchmark finds that label errors account for 8% absolute F1 test error, and that more than 50% of the 5k selected examples need to be re-labeled, because the annotations generated by crowd-workers are incorrect.

Revised TACRED benchmark To improve the accuracy and reliability of future relation extraction method evaluations, I release the revised TACRED development and test set with the 5k most challenging examples re-annotated, and labeling errors corrected.[2]

Analysis of common model errors and state-of-the-art methods I provide a detailed analysis and discussion of the TACRED benchmark, its annotations, and state-of-the-art model errors. The first analysis evaluates the most challenging, incorrectly predicted examples of the revised test set, and develops a set of 9 categories for common relation extraction errors, that will also aid evaluation on other datasets. I then formalize, verify, and discuss the previously developed error hypotheses on three state-of-the-art relation extraction methods and show that two groups of ambiguous relations are responsible for most of the remaining errors, and that models exploit cues in the dataset when entities are unmasked.

6.2 Fine-Grained Analysis

As previously discussed, the goal of the fine-grained analysis is to provide insights into models by analyzing their errors, or incorrect predictions. A model, however, can fail for various reasons: examples that are difficult to solve; dataset biases that could be exploited to solve the task more easily; and annotation errors, which are often observed in crowdsourced datasets. An effective analysis therefore should focus on incorrect predictions, but also consider the dataset and its annotations as a possible source of errors. Many datasets, however, are too large to thoroughly inspect all examples, and inspecting them without sufficient evidence can be time consuming. For instance, errors of a single model may be insufficient to highlight examples of interest that are caused by a systematic issue, such as annotation errors. It therefore is best to first collect evidence from multiple models to highlight and rank examples for human

[2] https://github.com/dfki-nlp/tacrev

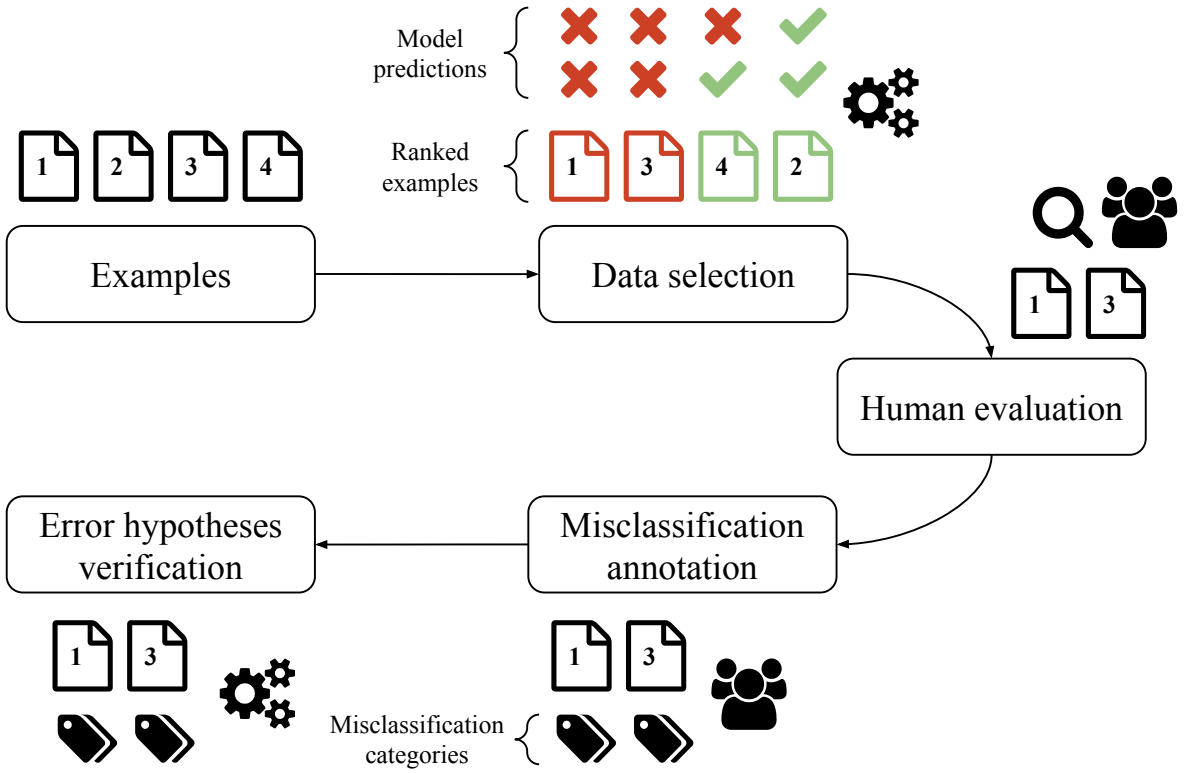


Figure 6.1 The first step of the fine-grained analysis (data selection) gathers evidence for possible systematic issues, e.g., annotation errors, by aggregating incorrect predictions of multiple models for each example and ranking them accordingly. The highest ranked examples are then selected for human evaluation with the goal to re-annotate them, and to subsequently determine possible error hypotheses (misclassification annotation). In the last step, the misclassification categories are extended to testable hypotheses and automatically verified on the whole dataset.

evaluation, which has the goal to re-annotate the selected examples and develop possible error hypotheses. The hypotheses are subsequently formalized and verified on the whole dataset to ensure their validity. To allow for a more effective and fine-grained analysis of models, I propose the approach depicted in Figure 6.1. It assumes a set of labeled examples on which the models are evaluated, i.e., a dataset, and consists of four steps that subsequently analyze the dataset, its annotations, and model errors:

Data selection To identify systematic issues of models and datasets, and to build error hypotheses it is necessary to manually inspect incorrectly predicted examples. However, the number of examples is often quite large. The goal of data selection therefore is to collect evidence that best guides the subsequent manual evaluation efforts. It implements a selection strategy which is based upon ordering examples by the difficulty of predicting them correctly, i.e., to rank each example according to the number of models

predicting a different relation label than the ground truth. Intuitively, examples with large disagreement, between all models or between models and the ground truth, are either difficult, or incorrectly labeled. The highest ranking examples are then selected for manual evaluation in the next step.

Human evaluation The human evaluation involves manual inspection and potentially re-annotation of the selected examples, according to the task or annotation guidelines. This is the most time consuming step. Therefore, it is crucial to ensure that the manual evaluation is focused on examples with high probability of being incorrectly labeled, or being the artefact of a systematic model or dataset issue. In the former case annotators will revise an example, in the latter it is subject to detailed inspection in the next step.

Misclassification annotation The goal of the misclassification annotation is to identify possible linguistic aspects that cause incorrect model predictions. Starting from single observations, a system of categories is iteratively developed based on the existence, or absence, of contextual and entity-specific features that might mislead the models, e.g., entity type errors or distracting phrases. Following the exploration, the final set of categories is defined, guidelines are developed for each, and annotators are instructed to assign one of the error categories to each misclassified example, if applicable.

Error hypotheses verification Error hypotheses verification extends the misclassification categories developed in the previous step to testable hypotheses, or groups, that are verifiable on all examples, i.e., the whole dataset. For instance, if we suspect a model to be distracted by an entity in context that has the same type as one of the relation arguments, we formulate a group *has_distractor*. The group contains all examples, both correct and incorrectly predicted, that satisfy a certain condition, e.g., there exists at least one entity in the sentential context that has the same type as one of the arguments. The grouping ensures that we do not mistakenly prioritize groups that are actually well-handled on average. I follow the approach proposed by Wu et al. (2019), and extend their *Errudite* framework[3] to the relation extraction task. After formulating a hypothesis, the error prevalence can be assessed over the entire dataset split to validate whether the hypothesis holds, i.e., the group of instances shows an above average error rate. In a last step, the error hypothesis is explicitly tested by counterfactual evaluation (Section 2.5.2) of a group's examples, e.g., by replacing the distracting entities and observing the models' predictions on the altered examples. In our example, if the *has_distractor* hypothesis is correct, removing the entities in context should change the prediction of previously incorrect examples.

[3] https://github.com/uwdata/errudite

The overall approach allows for an effective and fine-grained analysis of model errors by first highlighting challenging examples for human evaluation based on the evidence provided by multiple model predictions. The manual evaluation identifies and potentially revises annotation errors, and subsequently develops model misclassification categories, which are then formalized into testable hypotheses that can be automatically verified for each model.

6.3 Evaluation

The goal of the evaluation is to demonstrate the effectiveness of the proposed analysis approach on the basis of an existing dataset. I select TACRED for this purpose, because it is one of the largest and most widely used relation extraction benchmarks. It contains more than 106k examples annotated by crowd workers, and the currently best performing methods (Baldini Soares et al., 2019; Peters et al., 2019) achieve an F1 score of 71.5 based on transfer learning, e.g., fine-tuning pre-trained language representations. Although this performance is impressive, the error rate of almost 30% is still high. Naturally, one might ask the following questions: Is there still room for improvement, and is it possible to identify the underlying factors that contribute to this error rate?

I use the proposed approach to analyse this question from two separate viewpoints: First, to what extent does the quality of crowd based annotations contribute to the error rate; and secondly, what can be attributed to dataset and models? Answers to these questions can provide insights for improving crowdsourced annotation in relation extraction, and suggest directions for future research. To answer the first question, I first select examples in the development and test set according to the misclassifications of 49 relation extraction models and select the top 5k instances for manual evaluation (Section 6.3.2). To answer the second question, two analyses are carried out. The first analysis conducts a manual explorative analysis of model misclassifications on the most challenging test instances and categorizes them into several linguistically motivated error categories (Section 6.3.3.1). In the second analysis, I formulate these categories into testable hypotheses, which can be automatically validated on the full test set by counterfactual analysis (Section 6.3.3.2).

I begin this section with an outline of the evaluation setup, followed by a detailed discussion of the analysis results.

6.3.1 Experimental Setup

As I selected the TACRED benchmark for analysis, I first provide a detailed description of the dataset and its creation process (Section 6.3.1.1). Although the previous section introduced the overall four step analysis approach, the specifics of each step depend on the dataset under investigation. I therefore describe each individual step in detail, namely, data selection, human validation, misclassification annotation, and error hypotheses verification (Section 6.3.1.2).

6.3.1.1 Dataset

The *TAC Relation Extraction Dataset* (TACRED), introduced by Zhang et al. (2017), is a fully supervised dataset of sentence-level binary relation mentions. It consists of 106k sentences with entity mention pairs collected from the TAC KBP evaluations 2009–2014, with the years 2009 to 2012 used for training, 2013 for development, and 2014 for testing. Each sentence is labeled with one of 41 person- and organization-oriented relation types, e.g. *per:title*, *org:founded*, or the label *no_relation* for negative instances. Table 6.1 summarizes key statistics of the dataset.

Split	# Examples	# Neg. examples
Train	68,124	55,112
Dev	22,631	17,195
Test	15,509	12,184

Table 6.1 TACRED statistics per split. About 79.5% of the examples are labeled as *no_relation*.

All relation labels were obtained by crowdsourcing, using Amazon Mechanical Turk. Crowd workers were shown the example text, with head (subject) and tail (object) mentions highlighted, and asked to select among a set of relation label suggestions, or to assign *no_relation*. Label suggestions were limited to relations compatible with the head and tail types.[4]

The data quality is estimated as relatively high by Zhang et al. (2017), based on a manual verification of 300 randomly sampled examples (93.3% validated as correct). The inter-annotator kappa label agreement of crowd workers was moderate at $\kappa = 0.54$ for 761 randomly selected mention pairs.

[4] See the supplemental material provided by Zhang et al. (2017) for details of the dataset creation and annotation process.

6.3.1.2 Details of Fine-Grained Analysis Steps

This section describes each step of the analysis in more detail, as the specifics of each depend on the scenario at hand, e.g., the dataset, the models used for data selection and error hypotheses verification, and annotation guidelines for manual evaluation.

Data selection I use a set of 49 different relation extraction models to obtain predictions on the development and test sets, and rank each example according to the number of models predicting a different relation label than the ground truth.[5] I select the following examples for validation: (a) *Challenging* – all examples that were misclassified by at least half of the models, and (b) *Control* – a control group of (up to) 20 random examples per relation type, including *no_relation*, from the set of examples classified correctly by at least 39 models. The two groups cover both presumably hard and easy examples, and allow to contrast validation results based on example difficulty. In total 2,350 (15.2%) *Test* examples and 3,655 (16.2%) *Dev* examples are selected for validation. Of these, 1,740 (*Test*) and 2,534 (*Dev*) were assigned a positive label, i.e., a relation, by crowd workers.

Human validation All selected examples are validated on the basis of the TAC KBP guidelines.[6] Validation follows the approach of Zhang et al. (2017), and presents each example by showing the example's text with highlighted head and tail spans, and a set of relation label suggestions. The setup differs from the original setup by showing more label suggestions to make the label choice less restrictive: (a) the original, crowd-generated ground truth label, (b) the set of labels predicted by the models, (c) any other relation labels matching the head and tail entity types, and (d) *no_relation*. The suggested positive labels are presented in an alphabetical order and are followed by *no_relation*, with no indication of a label's origin. Two annotators are asked to assign *no_relation* or up to two positive labels from this set. A second label was allowed only if the sentence expressed two relations, according to the guidelines, e.g., *per:city_of_birth* and *per:city_of_residence*. Any disagreements are subsequently resolved by a third annotator, who is also allowed to consider the original ground truth label. All annotators are educated in general linguistics, have extensive prior experience in annotating data for information extraction tasks, and are trained in applying the task guidelines in a trial annotation of 500 sentences selected from the development set.

[5] See Appendix A for details on the models, training procedure, hyperparameters, and task performance.
[6] https://tac.nist.gov/2014/KBP/ColdStart/guidelines/TAC_KBP_2014_Slot_Descriptions_V1.4.pdf

Misclassification annotation The first manual exploratory analysis focuses on the revised *Control* and *Challenging* test instances that are misclassified by the majority of the 49 models. Based on the exploration, a final set of categories is defined, guidelines are developed for each, and two annotators are instructed to assign an error category to each misclassified instance in the revised test subset. In cases where multiple categories are applicable the annotator selected the most relevant one. As in the validation step, any disagreements between the two annotators are resolved by a third expert.

Error hypotheses verification I evaluate the error hypotheses on a baseline and three of the most recent state-of-the-art relation extraction methods. Importantly, none of those were part of the model set used for data selection, so as not to bias the automatic evaluation. As the baseline, I select a single layer CNN (Nguyen and Grishman, 2015; Zeng et al., 2014) with max-pooling and 300-dimensional GloVe (Pennington et al., 2014) word embeddings as input. The state-of-the-art models use pre-trained language representations that were fine-tuned to the relation extraction task and include:

- **TRE**: The Transformer for Relation Extraction (Alt et al., 2019b) introduced in Section 3.2. It uses a pre-trained unidirectional language model based on the OpenAI Generative Pre-Trained Transformer (GPT) (Radford et al., 2018).
- **SpanBERT**: SpanBERT (Joshi et al., 2019) uses a pre-trained bidirectional language model which is based on BERT (Devlin et al., 2019). It, however, differs from BERT in that it is pre-trained on a span level instead of word level.
- **KnowBERT**: KnowBERT (Peters et al., 2019) extends BERT to encode additional external knowledge, e.g., more fine-grained entity type information. In particular, I use KnowBERT-W+W, which is pre-trained by joint entity linking and language modelling on Wikipedia and WordNet.

6.3.2 Analysis of Label Errors

In order to identify the impact of potentially noisy, crowd-generated labels on the observed model performance, I start with the analysis of TACRED's label quality. The hypothesis is that while comparatively untrained crowd workers may on average produce relatively good labels for easy relation mentions, e.g., those with obvious syntactic and/or lexical triggers, or unambiguous entity type signatures such as *per:title*, they may frequently err on challenging examples, e.g., highly ambiguous ones or relation types whose scope is not clearly defined.

	Dev		Test	
	Challenging	*Control*	*Challenging*	*Control*
# Examples (# positive)	3,088 (1,987)	567 (547)	1,923 (1,333)	427 (407)
# Revised (# positive)	1,610 (976)	46 (46)	960 (630)	38 (38)
# Revised (% positive)	**52.1** (**49.1**)	**8.1** (**8.4**)	**49.9** (**47.3**)	**8.9** (**9.3**)

Table 6.2 Re-annotation statistics for TACRED *Dev* and *Test* splits.

Table 6.2 shows the results of the validation process. In total, the annotators revised 960 (49.9%) of the *Challenging Test* examples, and 1,610 (52.1%) of the *Challenging Dev* examples, a very large fraction of label changes for both dataset splits. Revision rates for originally positive examples are lower at 47.3% (*Test*) and 49.1% (*Dev*). Approximately 57% of the negative examples were re-labeled with a positive relation label (not shown). Two labels were assigned to only 3.1% of the *Test*, and 2.4% of the *Dev* examples. The multi-labeling mostly occurs with location relations, e.g., the phrase *"[Gross]$_{head:per}$, a 60-year-old native of [Potomac]$_{tail:city}$"* is labeled with *per:city_of_birth* and *per:city_of_residence*, which is justified by the meaning of the word *native*.

As expected, the revision rate in the *Control* groups is much lower, at 8.9% for *Test* and 8.1% for *Dev*. We can also see that the fraction of negative examples is approximately one-third in the *Challenging* group, much lower than the dataset average of 79.5%. This suggests that models have more difficulty predicting positive examples correctly.

	Dev		Test	
IAA	H1,H2	H,C	H1,H2	H,C
Challenging	0.78	0.43	0.85	0.44
Control	0.87	0.95	0.94	0.96
All	0.80	0.53	0.87	0.55

Table 6.3 Inter-Annotator Kappa-agreement for the relation validation task on TACRED *Dev* and *Test* splits (H1,H2 = agreement between human re-annotators, H,C = average agreement between human re-annotators and original TACRED crowd-generated labels).

The validation inter-annotator agreement is shown in Table 6.3. It is very high at $\kappa_{Test} = 0.87$ and $\kappa_{Dev} = 0.80$, indicating a high annotation quality. For both *Test* and *Dev*, it is higher for the easier *Control* groups than for the *Challenging* groups. In contrast, the average agreement between human re-annotators and the crowdsourced labels is much lower at $\kappa_{Test} = 0.55$, $\kappa_{Dev} = 0.53$, and lowest for *Challenging* examples (e.g., $\kappa_{Test} = 0.44$).

Frequently erroneous crowd labels are *per:cities_of_residence*, *org:alternate_names*, and *per:other_family*. Typical errors include mislabeling an example as positive which does not express the relation, e.g., labeling *"[Alan Gross]$_{head:per}$ was arrested at the [Havana]$_{tail:loc}$ airport."* as *per:cities_of_residence*, or not assigning a positive relation label, e.g., *per:other_family* in *"[Benjamin Chertoff]$_{head:per}$ is the Editor in Chief of Popular Mechanics magazine, as well as the cousin of the Director of Homeland Security, [Michael Chertoff]$_{tail:per}$"*. Approximately 49% of the time an example's label was changed to *no_relation* during validation, 36% of the time from *no_relation* to a positive label, and the remaining 15% it was changed to or extended with a different relation type.

To measure the impact of dataset quality on the performance of models, I evaluated all 49 models on the revised test split. The average model F1 score rises to 70.1%, a major improvement of 8% over the 62.1% average F1 on the original test split, corresponding to a 21.1% error reduction.

Discussion The large number of label corrections and the improved average model performance show that the quality of crowdsourced annotations is a major factor contributing to the overall error rate of models on TACRED. Even though the selection strategy was biased towards examples challenging for models, the large proportion of changed labels suggests that these examples were difficult to label for crowd workers as well. To put this number into perspective – Riedel et al. (2010) showed that, for a distantly supervised dataset, about 31% of the sentence-level labels were wrong, which is less than what we observe here for human-supervised data.[7]

The low quality of crowd-generated labels in the *Challenging* group may be due to their complexity, or due to other reasons, such as lack of detailed annotation guidelines, lack of training, etc. It suggests that, at least for *Dev* and *Test* splits, crowdsourcing, even with crowd worker quality checks as used by Zhang et al. (2017), may not be sufficient to produce high quality evaluation data. While models may be able to adequately utilize noisily labeled data for training, measuring model performance and comparing progress in the field may require an investment in carefully labeled evaluation datasets. This may mean, for example, that we need to employ well-trained annotators for labeling evaluation splits, or that we need to design better task definitions and task presentations setups as well as develop new quality control methods when using crowd-sourced annotations for complex NLP tasks like relation extraction.

[7] Riedel et al.'s estimate is an average over three relations with 100 randomly sampled examples each, for similar news text. Two of the relations they evaluated, *nationality* and *place_of_birth*, are also contained in TACRED, the third is *contains* (location).

6.3.3 Analysis of Model Errors and Dataset

In this section, I present the main analysis results, providing an answer to the question: which of the remaining errors can be attributed to the models, and what are the potential reasons for these errors? I first discuss the findings of the manual misclassification analysis (Section 6.3.3.1), followed by the results of the automatic analysis (Section 6.3.3.2).

6.3.3.1 Model Error Categories

Table 6.4 summarizes the developed linguistic misclassification categories. It distinguishes between errors resulting from (1) relation argument errors, and (2) context misinterpretation.[8] The category *relation argument errors* refers to misclassifications resulting from incorrectly assigned *entity spans* or *entity types* of arguments. Type annotation errors are always labeled, but minor span annotation errors were tolerated if they did not change the interpretation of the relation or the entity.

The category *context misinterpretation* refers to cases where the sentential context of the arguments is misinterpreted by the model. The following context problems were identified: (1) *Inverted arguments*: The prediction is inverse to the correct relation, i.e., the model's prediction would be correct if head and tail were swapped. (2) *Wrong arguments*: The model incorrectly predicts a relation that holds between head or tail and an un-annotated entity mention in the context, therefore misinterpreting one annotated argument. (3) *Linguistic distractor*: The example contains words or phrases related to the predicted relation, however they do not connect to any of the arguments in a way justifying the prediction. (4) *Factuality*: The model ignores negation, speculation, future tense markers, etc. (5) *Context ignored*: The example does not contain sufficient linguistic evidence for the predicted relation except for the matching entity types. (6) *Relation definition*: The predicted relation could be inferred from the context using common sense or world knowledge, however the inference is prohibited by the guidelines (e.g., the spokesperson of an organization is not a top member/employee, or a work location is not a pointer to the employee's residence). (7) *No Relation*: The model incorrectly predicts *no_relation* even though there is sufficient linguistic evidence for the relation in the sentential context.

[8]The manual analysis focused on the sentence semantics, and left aspects such as sentence length, distance between entities, etc. for the automatic analysis, which can handle the analysis of surface features more effectively.

Error Type	Examples	Prediction	Freq.
Arguments			
Span	This is a tragic day for the Australian [Defence Force]$_{head:org}$ ([ADF]$_{tail:org}$)	*org:alt._nam*	12
Entity Type	[Christopher Bollyn]$_{head:per}$ is an [independent]$_{tail:religion}$ journalist	*per:religion*	31
	The company, which [Baldino]$_{head:org}$ founded in [1987]$_{tail:date}$ sells a variety of drugs	*org:founded*	
Context			
Inverted Args	[Ruben van Assouw]$_{head:per}$, who had been on safari with his 40-year-old father [Patrick]$_{tail:per}$, mother Trudy , 41 , and brother Enzo , 11 .	*per:children*	25
Wrong Args	Authorities said they ordered the detention of Bruno 's wife , [Dayana Rodrigues]$_{tail:per}$, who was found with [Samudio]$_{head:per}$'s baby .	*per:spouse*	109
Ling. Distractor	In May , [he]$_{head:per}$ secured $ 96,972 in working capital from [GE Healthcare Financial Services]$_{tail:org}$.	*per:employ._of*	35
Factuality	[Ramon]$_{head:per}$ said he hoped to one day become an [astronaut]$_{head:title}$	*per:title*	11
	Neither he nor [Aquash]$_{head:per}$ were [American]$_{tail:nationality}$ citizens .	*per:origin*	
Relation Def.	[Zhang Yinjun]$_{tail:per}$, spokesperson with one of China 's largest charity organization , the [China Charity Federation]$_{head:org}$	*org:top_mem.*	96
Context Ignored	[Bibi]$_{head:per}$, a mother of [five]$_{tail:number}$, was sentenced this month to death .	*per:age*	52
No Relation	[He]$_{head:per}$ turned a gun on himself committing [suicide]$_{tail:causeofdeath}$.	*no_relation*	646
		Total	**1017**

Table 6.4 Misclassification types along with sentence examples, relevant false predictions, and error frequency. The problematic sentence parts are underlined (examples may be abbreviated due to space constraints).

Discussion The relation label predicted most frequently across the 49 models disagreed with the ground truth label of the re-annotated *Challenging* and *Control Test* groups in 1017 (43.3%) of the cases. The inter-annotator agreement of error categories assigned to these examples is high at $\kappa_{Test} = 0.83$ ($\kappa_{Test} = 0.67$ if the category *No Relation* is excluded).

Argument errors accounted for only 43 (4.2%) misclassifications, since the entities seem to be mostly correctly assigned in the dataset. In all entity type misclassification cases except one, the errors originate from false annotations in the dataset itself.

Context misinterpretation caused 974 (95.8%) false predictions. *No relation* is incorrectly assigned in 646 (63.6%) of misclassified instances, even though the correct relation is often explicitly and unambiguously stated. In 134 (13.2%) of the erroneous instances the misclassification resulted from inverted or wrong argument assignment, i.e., the predicted relation is stated, however, the arguments are inverted or the predicted relation involves an entity other than the annotated one. In 96 (9.4%) instances the error results from TAC KBP guidelines prohibiting specific inferences, affecting most often the classification of the relations *per:cities_of_residence* and *org:top_member/employee*. Furthermore, in 52 (5.1%) of the false predictions models seem to ignore the sentential context of the arguments, i.e., the predictions are inferred mainly from the entity types. Sentences containing linguistic distractors accounted for 35 (3.4%) incorrect predictions. Factuality recognition causes only 11 errors (1.1%). However, the assumption is that this latter low error rate is due to TACRED data containing an insufficient number of sentences suitable for extensively testing a model's ability to consider the missing factuality of relations.

6.3.3.2 Automated Model Error Analysis

For the automated analysis, I defined the following categories and error groups:

- **Surface structure**: Error groups in this category are based on surface properties, e.g., argument distance and sentence length. In particular, I test whether examples with a short distance between relation arguments (*argdist=1*), long distance between arguments (*argdist>10*), or long sentences (*sentlen>30*) result in an increased model error rate.
- **Arguments**: The argument semantics category includes tests for entity type signature and argument ordering. Specifically, I test head and tail mention named entity types (*same_nertag*, *per:**, *org:**, *per:loc*), pronominal head/tail (*has_coref*), and relations with an existing inverse (*has_inverse*), e.g., *org:parents* vs. *org:subsidiary*.
- **Context**: For contextual information, I define groups based on, e.g., whether there exists a distracting entity of same type between the arguments (*has_distractor*), or whether there is a large number of (distracting) entities in the sentential context (*count(entities)>5*).

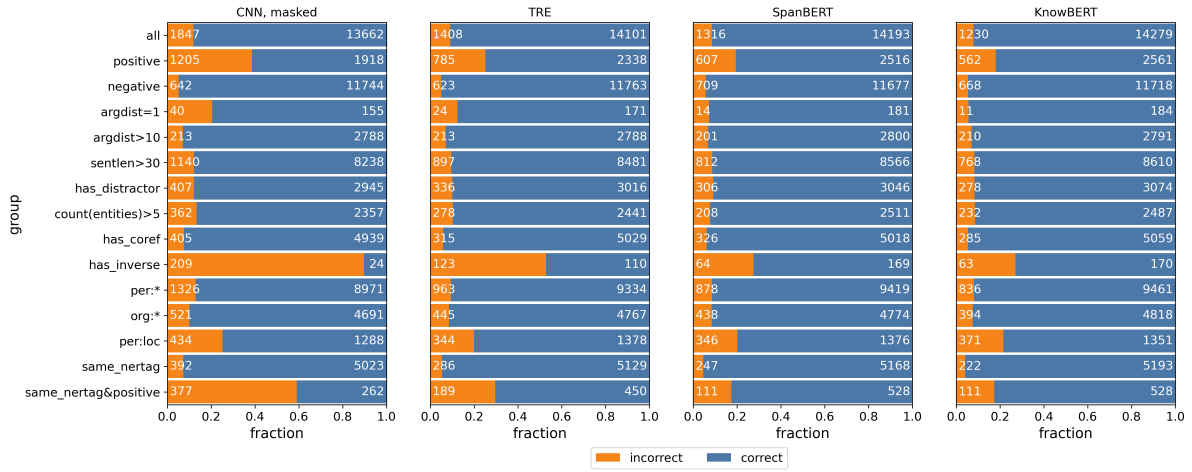


Figure 6.2 Error rates for different groups (example subsets) on the revised TACRED test set, for four different models. The bars show the number and fraction of correctly (blue) and incorrectly (orange) predicted examples in the given group. KnowBERT, as the best-performing model, has the lowest error rates across most groups. Error rates for *has_inverse*, *per:loc*, *same_nertag&positive* are higher for all models than the model error rate on the complete test set (*all*), highlighting examples for further error analysis and potential model improvements.

- **Ground Truth**: Finally, some groups are also conditioned on the ground truth label, e.g., whether the ground truth is "no relation" (*negative*) or a positive label (*positive*, *same_nertag&positive*).

Figure 6.2 shows the error rates of different groups on the revised TACRED test set. The plot shows error rates across four representative models. Each chart displays groups on the y-axis, and the fraction and number of correct (blue) vs. incorrect (orange) instances in the respective group on the x-axis. The average error rate of each model on the full test set is shown for reference in the top-most column titled *all*. Groups with higher than average error rate may indicate characteristics of examples that make classification difficult. On the other hand, groups with lower than average error rate comprise examples that the given model performs especially well on.

What is the error rate for different groups? In Figure 6.2, we can see that KnowBERT has the lowest error rate on the full test set (7.9%), and the masked CNN model the highest (11.9%). SpanBERT's and TRE's error rates are in between the two. Overall, all models exhibit a similar pattern of error rates across the groups, with KnowBERT performing best across the board, and the CNN model worst. We can see that model error rates, e.g., for the groups *has_distractor*, *argdist>10*, and *has_coref* do not diverge much from the corresponding overall

model error rate. The presence of distracting entities in the context therefore does not seem to be detrimental to model performance. Similarly, examples with a large distance between the relation arguments, or examples where co-referential information is required, are generally predicted correctly.

On the other hand, we can see that all models have above-average error rates for the group *positive*, its subgroup *same_nertag&positive*, and the groups *per:loc* and *has_inverse*. The above-average error rate for *positive* may be explained by the fact that the dataset contains much fewer positive than negative training instances, and is hence biased towards predicting *no_relation*. For *has_inverse*, the automatic analysis confirms that the observation made during manual error analysis also holds true for the entire test dataset, including correctly classified instances. Especially the CNN model almost never correctly classifies test instances in this group, while the TRE model still has an error rate of approximately 55%, almost twice as much as SpanBERT and KnowBERT. However, a detailed analysis of erroneous examples in this group shows that in most cases, instances are misclassified as negative or as another relation with the same entity type signature, but not actually as the inverse relation type. Thus, the existence of an inverse relation is not truly the cause of misclassifications. A subsequent analysis shows that the groups *per:loc* and *same_nertag&positive* are the most ambiguous. *per:loc* contains relations such as *per:cities_of_residence*, *per:countries_of_residence* and *per:origin*, that may be expressed in a similar context but differ only in the fine-grained type of the tail argument (e.g., per:city vs. per:country). In contrast, *same_nertag* contains all person-person relations such as *per:parents*, *per:children* and *per:other_family*, as well as, e.g., *org:parent* and *org:subsidiaries* that involve the same argument types (per:per vs. org:org) and may be only distinguishable from context.

How important is context? KnowBERT and SpanBERT show about the same error rate on the groups *per:loc* and *same_nertag&positive*. They, however, differ in which examples they predict correctly: For *per:loc*, 78.6% are predicted by both models, and 21.4% are predicted by only one of the models. For *same_nertag&positive*, 12.8% of the examples are predicted by only one of the models. The two models thus seem to identify complementary information. One difference between the models is that KnowBERT has access to entity information, while SpanBERT masks entity spans.

To test how much the two models balance context and argument information, I apply rewriting to alter the instances belonging to a group and observe the impact on performance. I use two strategies: (1) I remove all tokens outside the span between head and tail argument

(*outside*), and (2) I remove all tokens between the two arguments (*between*). I find that SpanBERT's performance on *per:loc* drops from 62.1 F1 to 57.7 (outside) and 43.3 (between), whereas KnowBERT's score decreases from 63.7 F1 to 60.9 and 50.1, respectively. On *same_nertag&positive*, I observe a drop from 89.2 F1 to 58.2 (outside) and 47.7 (between) for SpanBERT. KnowBERT achieves a score of 89.4, which drops to 83.8 and 49.0. The larger drop in performance on *same_nertag&positive* suggests that SpanBERT, which uses entity masking, focuses more on the context, whereas KnowBERT focuses on the entity content because the model has access to the arguments. Surprisingly, both models show similar performance on the full test set (Table 6.5). This suggests that combining both approaches may further improve relation extraction performance.

Should instance difficulty be considered? Another question is whether the dataset contains instances that can be solved more easily than others, e.g., those with simple patterns or patterns frequently observed during training. I assume that these examples are also more likely to be correctly classified by the baseline set of 49 relation extraction models.

Model	Original	Revised	Weighted
CNN, masked	59.5	66.5	34.8
TRE	67.4	75.3	48.8
SpanBERT	70.8	78.0	61.9
KnowBERT	71.5	79.3	58.7

Table 6.5 Test set F1 score on TACRED, the revised version, and weighted by difficulty (on the revised version). The weight per instance is determined by the number of incorrect predictions in the set of 49 RE models. The result suggests that SpanBERT better generalizes to more challenging examples.

To test this hypothesis, I change the evaluation setup and assign a weight to each instance based on the number of correct predictions. An example that is correctly classified by all 49 baseline models would receive a weight of zero – and thus effectively be ignored – whereas an instance misclassified by all models receives a weight of one. In Table 6.5, we can see that SpanBERT has the highest score on the weighted test set (61.9 F1), a 16% decrease compared to the unweighted revised test set. KnowBERT has the second highest score of 58.7, 3% less than SpanBERT. The performance of TRE and CNN is much worse at 48.8 and 34.8 F1, respectively. The result suggests that SpanBERT's span-level pre-training and entity masking are beneficial to relation extraction and allow the model to generalize better to challenging examples. Given this observation, I propose to consider an instance's difficulty during evaluation.

6.4 Discussion and Summary

In this chapter, I proposed a fine-grained analysis approach for relation extraction methods that targets model errors, dataset, and annotations. To demonstrate its effectiveness, I conducted an extensive analysis of the TACRED benchmark. The first analysis validated the 5k most challenging examples in development and test set and found that labeling is a major error source, accounting for 8% absolute F1 error on the test set. This clearly highlights the need for careful evaluation of development and test splits when creating datasets via crowdsourcing. To improve the evaluation accuracy and reliability of future relation extraction methods, I released a revised, extensively re-labeled dataset. An additional analysis categorized the model misclassifications into 9 common relation extraction error categories and showed that models are often unable to predict a relation, even if it is expressed explicitly. Models also frequently do not recognize argument roles correctly, or ignore the sentential context. In an extensive automated evaluation, I verified the error hypotheses on the whole test split and showed that two groups of ambiguous relations are responsible for most of the remaining errors. I further showed that models adopt heuristics when entities are unmasked and proposed that evaluation metrics should consider an instance's difficulty.

Although the evaluation clearly shows the effectiveness of the fine-grained analysis, I found two aspects merit further investigation. First, considerable effort must be spent on manual evaluation and re-annotation. While this certainly is acceptable for standard benchmarks like TACRED, it may be infeasible for other domains or applications. Also, creating a large set of models for data selection can be costly or is not always possible. Second, it is still challenging to accurately define error hypotheses, i.e., to create error groups that exhibit an above average error rate.

I conclude that special care must be taken to ensure high dataset quality, in particular for standard benchmarks, which allows us to accurately track improvements in modeling abilities for the task. Also, it is crucial to conduct additional fine-grained analyses on other datasets to ensure high evaluation quality and to identify aspects of the relation extraction task that are not yet well handled by current methods. In the latter case, it is important to understand how models and dataset contribute to the errors. Furthermore, it would benefit future benchmarks if they accurately reflect important aspects of the relation extraction task, such as factuality, negation, and temporal aspects.

Chapter 7

Conclusion

7.1 Summary

Relation extraction in practical scenarios is often characterized by limited availability of supervised data, due to the cost of creation and scarcity of domain-specific resources. In this thesis, I researched methods for such scenarios that reuse knowledge acquired from related tasks for relation extraction. Based on an investigation of transfer learning approaches, I proposed a novel sequential transfer learning method for supervised relation extraction that is based on unsupervised language model pre-training. First, a model is trained on a large collection of text to acquire knowledge about the language. In a second step, the knowledge is transferred to the task of relation extraction by further training, or fine-tuning, the model on task-specific supervised data. I showed that this approach increases overall performance and data efficiency in settings with limited labeled data compared to state-of-the-art methods. To also utilize noisily supervised data, which is often available in larger quantities, I subsequently extended the proposed method to distantly supervised relation extraction by combining it with a mechanism for multi-instance learning.

The improved modeling abilities, however, made it even more evident that better understanding these methods is crucial to further performance improvements. I first presented an approach based on probing tasks to uncover linguistic features of the input that neural models encode and use for relation classification. I used this approach to show how network architecture and inclusion of supporting information affects the linguistic knowledge encoded by more than 40 different neural relation extraction models. Furthermore, I complemented this with a fine-

grained semi-automated analysis approach focused on model errors, dataset, and annotations. To demonstrate its effectiveness, I conducted a thorough analysis of the widely used TACRED benchmark, including state-of-the-art model errors, and found that label errors account for 8% absolute F1 test error, and that more than 50% of the 5k inspected examples need to be re-labeled. Further, I showed two groups of ambiguous relations to be responsible for most of the remaining errors, and that models exploit cues in the dataset when entities are unmasked.

Together, the researched approaches allow us to build better performing, more data efficient relation extraction models, and advance our understanding despite their complexity. Further, it facilitates fine-grained analysis of model errors and datasets in the future.

7.2 Outlook

This thesis focused exclusively on binary relation extraction on a sentence level, i.e., identifying relations between two entities mentioned in a single sentence. It therefore would be interesting to extend the proposed methods to n-ary relation extraction and also to the document level. Also, to get a more complete picture it would be of interest to analyze the linguistic knowledge encoded by relation extraction models trained on other datasets, and how this affects the ability of models to extract relations of varying complexity. Another very important research direction would be to analyze further standard relation extraction benchmarks to ensure that the measured progress can indeed be attributed to improved modeling abilities instead of biases in the dataset or annotation errors.

In my opinion, future work on advancing the state of the art in relation extraction should focus on two aspects: Improving reuse and acquisition of relevant knowledge, and better defining the necessary abilities for the task.

Improving reuse and acquisition of relevant knowledge It is unrealistic to assume that task-specific supervised data will grow in size. Even benchmark datasets are often magnitudes larger than those used in practical scenarios. While distant supervision can help in such circumstances, it falls short if no pre-existing knowledge base is available. Furthermore, unsupervised relation extraction is typically unable to extract the desired relational structure without supervision, defeating the purpose of extracting clean structured information. I believe that only the ability to effectively use additional knowledge, either through pre-training or

dynamically at run time, can further improve performance and data efficiency of relation extraction methods.

Current pre-trained language representations are good at capturing the syntactic meaning of a word in context, however, less so for semantic meaning, e.g., as demonstrated on TACRED, where models are unable to sufficiently determine the type of an entity or do not generalize well to infrequent or unseen entities. It is therefore crucial to develop pre-training methods that bias models towards encoding more commonsense knowledge, i.e., information about entities and the ways they interact. Knowledge enhanced pre-training, e.g., by joint entity linking and language modeling, already started to address this limitation. The approach, however, still requires supervision, e.g., entity linked text. Of particular interest would be to explore how self-supervised language modeling could be used as a signal instead of supervised entity linking. Also, no studies exist how supervised pre-training or multi-task learning on other (possibly related) tasks, such as, semantic role labeling, question answering, or coreference resolution, affects relation extraction performance and data efficiency.

Defining abilities necessary for the task Another important step in advancing the state of the art in relation extraction is to define and test the abilities of the task a model is expected to have. In practical scenarios, for example, a model should be able to estimate the factuality of an extracted relation, for example, "It is rumoured that Google is about to buy Deepmind." vs. "Google announced its acquisition of Deepmind.", where the former example is speculative, or uncertain, and the latter a confirmed fact. Current works, however, measure and report progress based on benchmark datasets that neglect these important aspects. I believe that future work should determine and discuss abilities and also create datasets to test them accordingly. As discussed in the previous chapter, important aspects include linguistic phenomena that must be handled in practical scenarios, such as factuality, negation, and temporal aspects (e.g., a relation may be true only for some period of time).

Bibliography

Heike Adel, Benjamin Roth, and Hinrich Schütze. Comparing convolutional neural networks to traditional models for slot filling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838, San Diego, California, June 2016. Association for Computational Linguistics.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR Conference Track*, Toulon, France, 2017.

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, 2000.

Alan Akbik. *Exploratory relation extraction in large multilingual data*. Doctoral thesis, Technische Universität Berlin, Berlin, 2016.

Alan Akbik and Alexander Löser. KrakeN: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 52–56, Montréal, Canada, June 2012. Association for Computational Linguistics.

Alan Akbik, Larysa Visengeriyeva, Priska Herger, Holmer Hemsen, and Alexander Löser. Unsupervised discovery of relations and discriminative extraction patterns. In *Proceedings of COLING 2012*, pages 17–32, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

Alan Akbik, Larysa Visengeriyeva, Johannes Kirschnick, and Alexander Löser. Effective selectional restrictions for unsupervised relation extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1312–1320, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

Christoph Alt, Marc Hübner, and Leonhard Hennig. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1388–1398, Florence, Italy, July 2019a. Association for Computational Linguistics.

Christoph Alt, Marc Hübner, and Leonhard Hennig. Improving relation extraction by pre-trained language representations. In *Proceedings of the 2019 Conference on Automated Knowledge Base Construction*, Amherst, Massachusetts, 2019b.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. Probing linguistic features of sentence-level representations in relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1534–1545, Online, July 2020a. Association for Computational Linguistics.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online, July 2020b. Association for Computational Linguistics.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D. Manning. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1556–1567, Doha, Qatar, October 2014. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July 2015a. Association for Computational Linguistics.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Tejasvi Chaganty, Jason Bolton, Melvin Jose Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. Bootstrapped self training for knowledge base population. In *TAC*, 2015b.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy, July 2019. Association for Computational Linguistics.

Jeremy Barnes, Lilja Øvrelid, and Erik Velldal. Sentiment analysis is not solved! assessing and probing sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 12–23, Florence, Italy, August 2019. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *International Conference on Learning Representations*, 2018.

Yonatan Belinkov and James Glass. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics (TACL)*, 7:49–72, 2019.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

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.

H Douglas Brown and Teaching By Principles. An interactive approach to language pedagogy. *NY: Longman*, 430, 2001.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

Razvan Bunescu and Raymond Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada, October 2005a. Association for Computational Linguistics.

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*, HLT '05, pages 724–731. Association for Computational Linguistics, 2005b.

Aljoscha Burchardt, Vivien Macketanz, Jon Dehdari, Georg Heigold, Jan-Thorsten Peter, and Philip Williams. A linguistic evaluation of rule-based, phrase-based, and neural mt engines. *The Prague Bulletin of Mathematical Linguistics*, 108(1):159–170, 2017.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 756–765, Berlin, Germany, August 2016. Association for Computational Linguistics.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany, August 2016. Association for Computational Linguistics.

Hao Cheng, Hao Fang, and Mari Ostendorf. Open-domain name error detection using a multi-task RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *AAAI/IAAI*, 2002.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

Alexis Conneau and Douwe Kiela. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium, 1996.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 696–703, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China, July 2015. Association for Computational Linguistics.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012.

Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China, November 2019. Association for Computational Linguistics.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

Ralph Grishman. Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692, 2019.

Ralph Grishman and John Sterling. New york university: Description of the proteus system as used for muc-5. Technical report, NEW YORK UNIV NY DEPT OF COMPUTER SCIENCE, 1993.

Ralph Grishman, David Westbrook, and Adam Meyers. Nyu's english ace 2005 system description. 2005.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

David Harbecke and Christoph Alt. Considering likelihood in nlp classification explanations with occlusion and language modeling. In *Proceedings of ACL 2020, Student Research Workshop*, 2020.

David Harbecke, Robert Schwarzenberg, and Christoph Alt. Learning explanations from language data. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 316–318, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. See: Syntax-aware entity embedding for neural relation extraction. In *AAAI*, 2018.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

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*, pages 541–550, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. Improving name tagging by reference resolution and relation detection. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 411–418. Association for Computational Linguistics, 2005.

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*, pages 1148–1158. Association for Computational Linguistics, 2011.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. Relation extraction with multi-instance multi-label convolutional neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1471–1480, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke S. Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *ArXiv*, abs/1907.10529, 2019.

Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Seokhwan Kim, Minwoo Jeong, and Gary Geunbae Lee. A local tree alignment approach to relation extraction of multiple arguments. *Inf. Process. Manage.*, 47:593–605, 2011.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

Johannes Kirschnick, Alan Akbik, and Holmer Hemsen. Freepal: A large collection of deep lexico-syntactic patterns for relation extraction. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2071–2075, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In *The Semantic Web – ISWC 2012*, number 7649 in Lecture Notes in Computer Science, pages 263–278. Springer Berlin Heidelberg, January 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4208–4215. AAAI Press, 2018. ISBN 9780999241127.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August 2016. Association for Computational Linguistics.

Xiao Ling and Daniel S Weld. Temporal information extraction. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018a.

Tianyi Liu, Xinsong Zhang, Wanhao Zhou, and Weijia Jia. Neural relation extraction via inner-sentence noise reduction and transfer learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2195–2204, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795, Copenhagen, Denmark, September 2017a. Association for Computational Linguistics.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704, Melbourne, Australia, July 2018c. Association for Computational Linguistics.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *International Conference on Learning Representations*, 2017b.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 430–439, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 510–517, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel 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*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178, 2006.

Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado, June 2015. Association for Computational Linguistics.

Allen Nie, Erin Bennett, and Noah Goodman. DisSent: Learning sentence representations from explicit discourse relations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4497–4510, Florence, Italy, July 2019. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy, July 2019. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 49–54. IEEE, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China, November 2019. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

Jakub Piskorski and Roman Yangarber. Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer, 2013.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany, August 2016. Association for Computational Linguistics.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 826–835, Austin, Texas, November 2016. Association for Computational Linguistics.

Pengda Qin, Weiran Xu, and William Yang Wang. DSGAN: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

Prajit Ramachandran, Peter Liu, and Quoc Le. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Soumya Ray and Mark W. Craven. Representing sentence structure in hidden markov models for information extraction. In *IJCAI*, 2001.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling Relations and Their Mentions without Labeled Text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '10)*, 2010.

Bryan Rink and Sanda Harabagiu. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

Roland Roller, Eneko Agirre, Aitor Soroa, and Mark Stevenson. Improving distant supervision using inference learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 273–278, Beijing, China, July 2015. Association for Computational Linguistics.

Marc-Antoine Rondeau and T. J. Hazen. Systematic error analysis of the Stanford question answering dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 12–20, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Barbara Rosario and Marti Hearst. Classifying semantic relations in bioscience texts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 430–437, Barcelona, Spain, July 2004.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Sebastian Ruder. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

Ivan Sanchez, Jeff Mitchell, and Sebastian Riedel. Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1975–1985, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Rico Sennrich. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain, April 2017. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. UNITN: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 464–469, Denver, Colorado, June 2015. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics, 2016.

Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June 2006. Association for Computational Linguistics.

Marios Skounakis, Mark W. Craven, and Soumya Ray. Hierarchical hidden markov models for information extraction. In *IJCAI*, 2003.

Alisa Smirnova and Philippe Cudré-Mauroux. Relation extraction using distant supervision: A survey. *ACM Computing Surveys (CSUR)*, 51(5):1–35, 2018.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218, 2014.

Ang Sun, Ralph Grishman, and Satoshi Sekine. Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 521–529, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Mihai Surdeanu, David McClosky, Mason Smith, Andrey Gusev, and Christopher Manning. Customizing an information extraction system to a new domain. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 2–10, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–729, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Peter D Turney. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655, 2008.

Hans Uszkoreit. Learning relation extraction grammars with minimal human intervention: strategy, results, insights and plans. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 106–126. Springer, 2011.

C. J. van Rijsbergen. Information retrieval. In *Encyclopedia of GIS*, 1979.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. RESIDE: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Sara Veldhoen, Dieuwke Hupkes, Willem H Zuidema, et al. Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@ NIPS*, 2016.

Patrick Verga, Emma Strubell, and Andrew McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018a. Association for Computational Linguistics.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2246–2255, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. Neural domain adaptation for biomedical question answering. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 281–289, Vancouver, Canada, August 2017. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2015.

Fei Wu and Daniel S. Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy, July 2019. Association for Computational Linguistics.

Yi Wu, David Bamman, and Stuart Russell. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Feiyu Xu, Hans Uszkoreit, and Hong Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 584–591, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Feiyu Xu, Hans Uszkoreit, Sebastian Krause, and Hong Li. Boosting relation extraction with limited closed-world knowledge. In *Coling 2010: Posters*, pages 1354–1362, Beijing, China, August 2010. Coling 2010 Organizing Committee.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540, Lisbon, Portugal, September 2015a. Association for Computational Linguistics.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany, August 2016a. Association for Computational Linguistics.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal, September 2015b. Association for Computational Linguistics.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1461–1470, Osaka, Japan, December 2016b. The COLING 2016 Organizing Committee.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1183–1194, Valencia, Spain, April 2017. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Yongxin Yang and Timothy M Hospedales. Trace norm regularised deep multi-task learning. 2017.

Roman Yangarber and Ralph Grishman. Nyu: Description of the proteus/pet system as used for muc-7 st. In *MUC*, 1998.

Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296, 2009.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, Rochester, New York, USA, April 2007. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel Methods for Relation Extraction. *J. Mach. Learn. Res.*, 3:1083–1106, 2003. ISSN 1532-4435.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*, 2015.

Kelly Zhang and Samuel Bowman. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. Attention-based capsule networks with dynamic routing for relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 986–992, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.

Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *International Conference on Learning Representations*, 2017.

Appendix A

Fine-Grained Analysis of Model Errors and Datasets

Data Selection

This section contains additional details about the models used for data selection.

Models

Table A.1 show the relation extraction performances for the models on the original TACRED and the revised version. I use the same entity masking strategy as Zhang et al. (2017), replacing each entity in the original sentence with a special <NER>-{SUBJ, OBJ} token where <NER> is the corresponding NER tag. For models with 'POS/NER', I concatenate part-of-speech and named entity tag embeddings to each input token embedding.

Hyperparameters

CNN For training I use the hyperparameters of Zhang et al. (2017). I employ *Adagrad* as an optimizer, with an initial learning rate of 0.1 and run training for 50 epochs. Starting from the 15th epoch, I gradually decrease the learning rate by a factor of 0.9. For the CNN I use 500 filters of sizes [2, 3, 4, 5] and apply L_2 regularization with a coefficient of 10^{-3} to all filter

weights. I use tanh as activation and apply dropout on the encoder output with a probability of 0.5. I use the same hyperparameters for variants with ELMo. For variants with BERT, I use an initial learning rate of 0.01 and decrease the learning rate by a factor of 0.9 every time the validation F1 score is plateauing. Also I use 200 filters of sizes [2, 3, 4, 5].

LSTM/Bi-LSTM For training I use the hyperparameters of Zhang et al. (2017). I employ *Adagrad* with an initial learning rate of 0.01, train for 30 epochs and gradually decrease the learning rate by a factor of 0.9, starting from the 15th epoch. I use word dropout of 0.04 and recurrent dropout of 0.5. The BiLSTM consists of two layers of hidden dimension 500 for each direction. For training with ELMo and BERT I decrease the learning rate by a factor of 0.9 every time the validation F1 score is plateauing.

GCN I reuse the hyperparameters of Zhang et al. (2018b). I employ *SGD* as optimizer with an initial learning rate of 0.3, which is reduced by a factor of 0.9 every time the validation F1 score plateaus. I use dropout of 0.5 between all but the last GCN layer, word dropout of 0.04, and embedding and encoder dropout of 0.5. Similar to the authors, I use path-centric pruning with K=1. I use two 200-dimensional GCN layers and similar two 200-dimensional feedforward layers with ReLU activation.

Self-Attention After hyperparameter tuning, I found 8 layers of multi-headed self-attention to perform best. Each layer uses 8 attention heads with attention dropout of 0.1, keys and values are projected to 256 dimensions before computing the similarity and aggregated in a feedforward layer with 512 dimensions. For training, I use *Adam* optimizer with an initial learning rate of 10^{-4}, which is reduced by a factor of 0.9 every time the validation F1 score plateaus. In addition, I use word dropout of 0.04, embedding dropout of 0.5, and encoder dropout of 0.5.

Model	Original			Revised		
	P	R	F1	P	R	F1
BoE	50.0	32.6	39.4	51.8	35.9	42.4
CNN	72.3	45.5	55.9	79.8	53.5	64.1
CNN, masked	67.2	53.5	59.5	72.5	61.4	66.5
CNN w/ POS/NER	72.2	54.7	62.2	79.7	64.3	71.2
CNN + ELMo	73.8	48.8	58.8	82.1	57.9	67.9
CNN + ELMo, masked	72.3	53.8	61.7	79.8	63.2	70.5
CNN + ELMo, masked w/ POS/NER	69.2	59.0	63.7	76.0	69.1	72.4
CNN + BERT uncased	71.9	51.1	59.7	79.5	60.2	68.5
CNN + BERT uncased, masked	69.0	62.0	65.3	74.9	71.7	73.2
CNN + BERT cased	69.7	54.3	61.0	77.6	64.3	70.4
CNN + BERT cased, masked	71.8	61.1	66.1	78.1	70.8	74.3
LSTM	59.3	47.5	52.7	65.9	56.2	60.6
LSTM, masked	63.4	51.7	57.0	68.7	59.7	63.9
LSTM, masked w/ POS/NER	65.4	56.8	60.8	71.2	66.0	68.5
LSTM + ELMo	61.5	61.3	61.4	68.1	72.2	70.1
LSTM + ELMo, masked	63.9	64.9	64.4	69.3	75.0	72.1
LSTM + ELMo, masked w/ POS/NER	61.7	67.8	64.6	66.1	77.3	71.2
LSTM + BERT uncased	64.7	60.2	62.4	71.6	71.0	71.3
LSTM + BERT uncased, masked	65.3	64.8	65.1	70.4	74.3	72.3
LSTM + BERT cased	66.2	59.8	62.8	73.5	70.8	72.1
LSTM + BERT cased, masked	68.9	61.9	65.2	75.0	71.8	73.4
Bi-LSTM	53.3	57.4	55.3	58.6	67.2	62.6
Bi-LSTM, masked	62.5	63.4	62.9	67.7	73.1	70.3
Bi-LSTM + ELMo	65.0	58.7	61.7	72.6	69.8	71.1
Bi-LSTM + ELMo, masked	63.3	64.8	64.1	68.9	75.2	71.9
Bi-LSTM + ELMo w/ POS/NER	64.8	57.9	61.2	72.1	68.6	70.3
Bi-LSTM + ELMo, masked w/ POS/NER	63.0	65.9	64.4	67.5	75.2	71.2
Bi-LSTM + BERT uncased	65.3	59.9	62.5	71.8	70.2	71.0
Bi-LSTM + BERT uncased, masked	64.9	66.0	65.4	69.6	75.3	72.4
Bi-LSTM + BERT cased	65.2	61.2	63.1	72.1	72.1	72.1
Bi-LSTM + BERT cased, masked	68.3	64.0	66.1	74.1	73.9	74.0
GCN	65.6	50.5	57.1	72.4	59.3	65.2
GCN, masked	68.2	58.0	62.7	74.3	67.4	70.7
GCN, masked w/ POS/NER	68.6	60.2	64.2	74.2	69.3	71.7
GCN + ELMo	66.5	57.6	61.7	73.4	67.7	70.4
GCN + ELMo, masked	68.5	61.3	64.7	74.5	71.0	72.7
GCN + ELMo, masked w/ POS/NER	67.9	64.8	66.3	73.3	74.4	73.9
GCN + BERT uncased	66.3	58.8	62.4	73.1	69.1	71.0
GCN + BERT uncased, masked	68.7	64.0	66.3	74.8	74.1	74.5
GCN + BERT cased	66.5	56.4	61.0	74.4	67.1	70.5
GCN + BERT cased, masked	67.2	64.6	65.9	72.9	74.7	73.8
S-Att.	56.9	58.3	57.6	62.2	67.8	64.9
S-Att., masked	65.0	66.8	65.9	69.3	75.8	72.4
S-Att. + ELMo	64.4	65.0	64.7	71.5	76.8	74.1
S-Att. + ELMo, masked	64.0	69.4	66.6	68.9	79.6	73.8
S-Att. + BERT uncased	60.6	67.6	63.9	66.3	78.7	72.0
S-Att. + BERT uncased, masked	64.0	69.7	66.7	68.9	80.0	74.0
S-Att. + BERT cased	63.5	64.1	63.8	70.4	75.7	73.0
S-Att. + BERT cased, masked	69.2	64.7	66.9	75.1	74.8	75.0
Average	65.6	59.5	62.1	71.8	69.2	70.1

Table A.1 Test set F1 score on TACRED and the revised version for all 49 models used for data selection.