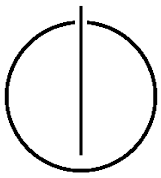
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Transfer Learning for Named Entity Linking with Deep Learning

Robin Otto



DEPARTMENT OF INFORMATICS

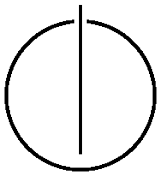
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Transfer Learning for Named Entity Linking with Deep Learning

Transfer Learning für Named Entity Linking mit Deep Learning

Author:	Robin Otto
Supervisor:	Prof. Dr. rer. nat. Florian Matthes
Advisor:	M. Sc. Ahmed Elnaggar
Submission Date:	15.05.2018



I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.05.2018 Robin Otto

Acknowledgments

First, I would like to thank Prof. Dr. rer. nat. Florian Matthes for granting me the elaboration of this thesis.

I also would like to thank my advisor Ahmed Elnaggar for his continuous feedback and encouragement throughout the thesis.

A thank-you goes to iteratec GmbH, especially to Lenz Belzner and Stefan Blum, who took the time to support me on whenever I needed help.

Without any of your support, it would have been considerably more difficult to elaborate this thesis.

Abstract

The involved challenges in named-entity linking are to differentiate between words in general and afterwards to link the occurrences of the same entities. For example, in “Paris is the capital of France. I enjoy visiting the city“ the challenge would be to associate the word “Paris“ with the location, as well as to link the occurrences of “Paris“ and “city“ to the same entity.

Current supervised neural networks designed for named-entity linking use publicly available datasets for training and testing. They address the problem of automated information extraction in general and try to maximize the accuracy and speed of these networks. This thesis focuses especially on the aspect to apply transfer learning to networks trained for named-entity linking. If this approach succeeds, one could apply these networks to enterprises and their internal datasets or other fields where richness of data is not always given. An example could be the integration of such networks in order to extract the skill base of the employees from CVs, commit histories, etc. Another instance for this approach would be the application on legal documents, like terms of service.

In order to make automated information extraction possible for datasets not publicly available, it is necessary to compare the state-of-the-art supervised deep learning neural networks coping with such problems.

Afterwards, rather than using the private datasets for training, we want to integrate and apply the best state-of-the-art neural networks by using the concept of Transfer Learning. Furthermore, we aim at evaluating the different types of transfer learning to determine the most beneficial technique for this particular problem. The expected result is to gain insight about the accuracy of state-of-the-art algorithms trained on publicly available datasets. In addition to this, the goal is a neural network that is capable of extracting information from private datasets. However, we assume this network will provide less accuracy.

Contents

1 Introduction

1.1 Overview

About 30 years ago, organizations started to automate different processes in their enterprise with the help of computers. It started with applications whose absence, from today's perspective, is inconceivable for almost every single organization in the world. An example for such a necessity was developing websites to create an interface between the customers and the enterprises. This usage of digitized data has evolved into countless areas, dependent on the individual needs of every company. [2]

Today's goals are naturally more advanced than simply creating a user interface or connecting different companies and customers through a network. A large topic in state of the art research is to make computers able to understand natural language. This process, called natural language process, is extremely difficult due to the complexity of human language. Numerous different approaches that achieve extremely well on different topics can be found online. Though, this does not apply to the legal domain, as its content is more sensitive and less data to experiment with is publicly available. Approaching the legal domain and its formulations in the context of natural language processing is still in a very early stage. Therefore, any result that can be extracted from research in this segment is highly interesting. [71]

1.2 Motivation

Written word is the central medium in the legal domain in which publications are made available. Over the last decade, digitalization confronted a lot of industry sectors with an imperative to change. The same process now also begins to apply to the legal domain. Digitalization of text can be provided in three different ways, unstructured, semi-structured or structured information. [2]

While structured information is relatively easy to understand for computers, the processing of unstructured and semi-structured is rather difficult. In the legal domain, most publications are of semi-structured nature and for this reason harder to understand. Therefore, a big value would be added by transferring these documents to fully structured documents. Legal documents contain not only irrelevant but also hardly graspable parts for non-legal stakeholders, e.g. engineers. A major improvement would be the explanation and highlighting of certain keywords. [70]

This is where named-entity linking comes into play. The process of applying named-entity linking to text eliminates the necessity to explain keywords in legal documents. Instead, the program provides an entity together with a definition for the occurrence of a keyword. Plenty approaches exist for this particular problem, most of them with the same flaw. In the state of the art, the task of named-entity linking heavily relies on the domain of the documents. A named-entity linker trained to understand technical documents for car engines won't understand keywords in a legal publication. [5]
Furthermore, as mentioned above, the legal domain heavily relies on written word as the central medium. Therefore, any tool or progress in training a computer program to be able to read text written by humans at enormous speed would grant a huge benefit to stakeholders like lawyers and their clients. [71]

In order to avoid having to solve the problem of disambiguating entities multiple times for every domain, we introduce deep learning together with transfer learning. This is motivated by the goal to generalize the task of learning the linking of entities by re-using previous efforts for a new domain. If it would be possible to keep the generalized knowledge of the original algorithm and to only discard the domain-specific knowledge, the new program would need to understand less in order to perform equally or even better. An example of context-independent knowledge of such an algorithm would be the ability to gain information from the surrounding words of a keyword. [4]

Finally yet importantly, a big factor that motivated this work is the will to actuate the progress in deep learning. Not only working on law related data, but specifically named-entity linking with deep learning and transfer learning represents a small niche in current natural language processing research. Concluding from this, exploring the current possibilities and limitations in that field motivates the elaboration of this work.

In summary, two crucial arguments motivate this work: The evaluation of the possibilities to enhance performance by recycling previous work in terms of fine-tuning an existing network as well as the analysis of the approach to include new datasets to an existing pipelines. Although there are already several applications in forms of *application programming interfaces*, like Dandelion [10] and Dexter [6], trying to employ such features, a successful approach on bringing transfer learning to named-entity linking with deep learning is yet to find.

1.3 Problem Statement

Technically, the following problems hold back any faster progress in the field of natural language processing in the legal domain.

- Data scarcity.
  Natural language processing is a subtask of machine learning. For most of the

machine learning tasks, especially in deep learning, a vast amount of data is needed to generate algorithms that achieve feasible results for the respective field. Especially in law only little data is accessible due to its sensitive nature. Hence, successful approaches with deep learning in the legal domain are rare.

- Lack of knowledge.
  Datasets that are dedicated to be used in a natural language processing task need to be preprocessed before they can be applied to a model. In terms of law, to prepare such data requires a lot of domain expert knowledge which is often not accessible by developers. The other way round, experts from the legal domain have access to and a decent understanding of feasible data, but do not have the rights to publish such data and mostly not the knowledge to use complex machine learning methods. [25]

- Absence of prior art.
  Named-entity linking with deep learning is a difficult task. The datasets have to be preprocessed in order to provide the necessary information for a network to successfully learn which entity to pick. Compared to deep learning and image recognition, only two inputs are necessary: an image and a respective label classifying that image. With this information a neural network is able to learn object detection at a decent level. The entire problem in named-entity linking is more difficult. The same word can refer to different entities depending on different factors. As mentioned in the very beginning, *Paris* can refer to a person and a location while the same picture can never refer to different labels. This complexity causes that most named-entity linking approaches use conventional machine learning methods that rely on engineered features.

1.4 Research Question and Objectives

The objective of this thesis is to gain insight on the process and results of applying transfer learning in terms of named-entity linking with deep learning. The dedicated dataset that is used in order to conduct experiments and to implement transfer learning originates from the legal domain. This dataset contains documents with bills passed by the European Parliament. For more information on the origin and creation of the dataset, see chapter 4. Necessary steps for this objective are expressed through the research questions below.

1. What kind of existing approach should be used for transfer learning?
   Before transfer learning can be analyzed or evaluated, a state of the art approach needs to be chosen which can be re-used and adapted for the purpose of this thesis. Details about the analyzed candidates for the application of transfer learning, see chapter 3. More insight on which approach has been chosen and why is provided in chapter 4. The design and concept of the desired named-entity linking approach will also be explained in that chapter.

2. Which technique of transfer learning suits best?
   Literature review and previous experience have shown that three different transfer learning techniques won recognition in the past. These three were implemented and their performance was tracked and captured accordingly. chapter 4 explains the realization of the chosen techniques together with the involved technologies.

3. Is the use of transfer learning with named-entity linking beneficial in the legal domain?
   After picking a desired, state of the art implementation, different experiments were conducted and evaluated in order to extract any possible benefits from using transfer learning compared to other techniques. See chapter 5 for the results of applying the different techniques and a comparison with other approaches. Finally, chapter 6 provides the conclusions that were drawn from the results of these experiments and highlights the limitations this thesis was restricted to.

1.5 Thesis Contribution

The major outcomes of this thesis are summarized by the bullet points below.

- Generation of a legal dataset.
  In this work, we created a new dataset for the named-entity linking task from the EUR-Lex corpus [18]. It has been automatically extracted. We put it into the same format as public benchmark datasets, see subsection 4.1.1 for details. Also, the created pipeline can be used to transform any raw text into a dataset that can be used for named-entity linking.

- Understand and modify a state of the art deep named-entity linking system [23]. By performing a lot of code archeology on that existing system, we were able to grasp its architecture. Thereby, we were able to modify the model so that it could be trained on other datasets apart from the ones it was originally created for. Thanks to this, we were able to conduct experiments on different datasets. We gained an increase in accuracy in some cases, which indicates the success of our approach. See section 5.2 for the achieved results.

- Apply transfer learning.
  In the scope of this work, we achieved to reproduce the results stated by the authors of the re-used named-entity linking system [23]. Furthermore we re-used the trained model to fine tune it on the EUR-Lex dataset [18] as well as the other way round. Thereby, we observed an increase in accuracy in at least one of the experiments.

1.6 Research Milestones

This section provides an incremental view of the milestones reached during the elaboration of this work. Figure 1.1 depicts the major milestones. First, a detailed state of the art analysis for named-entity linking with deep learning was conducted. Afterwards, we were able to apply transfer learning to the named-entity linking system by employing two datasets, one was manually annotated for the use of the entity disambiguation task, the other was the dataset generated in the scope of this work. Finally, the model was tested and based on the results some improvements were applied. The used model for the dedicated neural network even improved in accuracy after employing a transfer learning technique.

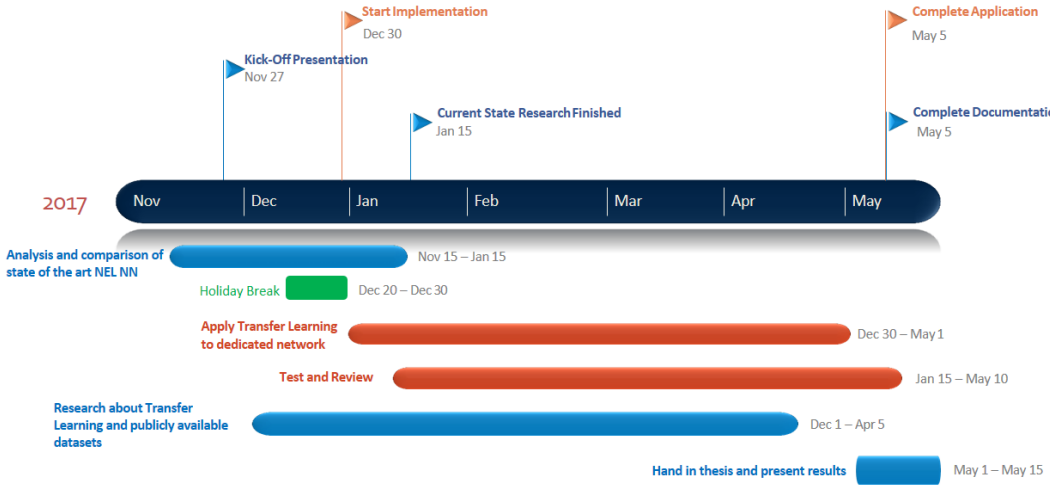


Figure 1.1: Research milestones achieved in this work.

1.7 Structure of the Work

This section shortly presents the different chapters of this thesis. After the above introduction, chapter 2 explain the basic foundations needed to follow this work. In particular, it breaks down the subjects *natural language processing*, *named-entity recognition & linking* and finally gives a brief introduction into *deep learning* with a focus on *transfer learning*.
Chapter 3 presents prior work on the topics involved in this thesis. First the state of the art in the field of *transfer learning* is depicted. Then, we show the current progress with regards to *named-entity linking in deep learning*. Subsequently, we take a deeper look into recent approaches on *named-entity linking in the legal domain*.
In chapter 4 we explain all practical challenges that have been addressed in the scope of this thesis. Starting off with the necessary preparations, we eventually present the

application of transfer learning to a state of the art named-entity linking system.
Chapter 5 shows all conducted experiments and interprets the respective results. Besides applying transfer learning, we also experimented with two other methods to be able to outline the benefits and disadvantages of transfer learning. Furthermore, a discussion including limitations to this work with respect to the achieved results is provided in that chapter.
Finally, in chapter 6, we present our conclusions drawn from the results with respect to the involved limitations. Additionally, some future work is depicted, that would have suited the topic of this thesis but did not fit in its scope due to time restrictions.

2 Foundations

The following chapter copes with the fundamental concepts of this thesis. These sections contain the knowledge required to follow the rest of the work, in particular chapter 4.

2.1 Deep Learning

This section provides explanations about deep learning that are necessary to follow the implementation of the thesis. First, this section will give an understanding about the basics of deep learning and neural networks. Afterwards, transfer learning, which is a deep learning technique, will be explained more into detail as the general understanding of this technique is fundamental for this thesis.

2.1.1 Deep Learning in General

Ever since computers have been around, different methods for enabling artificial intelligence are being explored. The experiments started with problems that were hard to solve for humans, mostly tasks that could be completely described with mathematical rules. Over the years, this transformed to problems that seemed easy and intuitive to solve for humans, since these are the tasks that are hard to describe formally. Examples for these tasks would be the recognition of objects on images or the recognition of speech. [29]

Deep learning is a topic belonging to the umbrella term artificial intelligence. On a high level view, deep learning enables computers to build complex concepts out of simple components. We will explain the concept of deep learning a bit more detailed on the basis of Figure 2.1.

It starts with raw input data in form of an image. As the goal of the algorithm, also called neural network, is to identify the object in the pixels of the image, it needs to learn a mapping from pixels to the correct object. Since the direct implementation of such a mapping is very difficult, and for some problems even impossible to do so, a different approach is necessary. To realize the complex concept, the problem has to be broken down into multiple steps representing smaller, simpler mappings. Each of these mappings is represented by a so called layer in the neural network.

The setup in this example, as you can see on Figure 2.1, starts with a pixel representation of an input image. The raw image of pixels that are fed into the network is

called input or also visible layer. Based on the input, one or more hidden layers extract features from the image growing in complexity the more layers there are. These layers are called hidden layers because their value is determined by the algorithm itself and it can not be observed. Each layer outputs the extracted features in an image.

An explicit pipeline for an image recognition neural network would be the following. First, the input data in form of images represented by pixels are fed to the network in the visible layer. In the next step, the first hidden layer tries to extract basic features from the pixels, like edges, by comparing the brightness of each pixel. Afterwards, in the next hidden layer, the network tries to identify corners and contours by clustering edges, the output of the first hidden layer. Having corners and contours as an input, the following hidden layer tries to derive a description of the image from the corners and contours to find object parts. Finally, in the last layer, the so called output layer, using the detected object parts from the previous layer, the algorithm derives a description of the image from object parts and returns an object identity as a result.

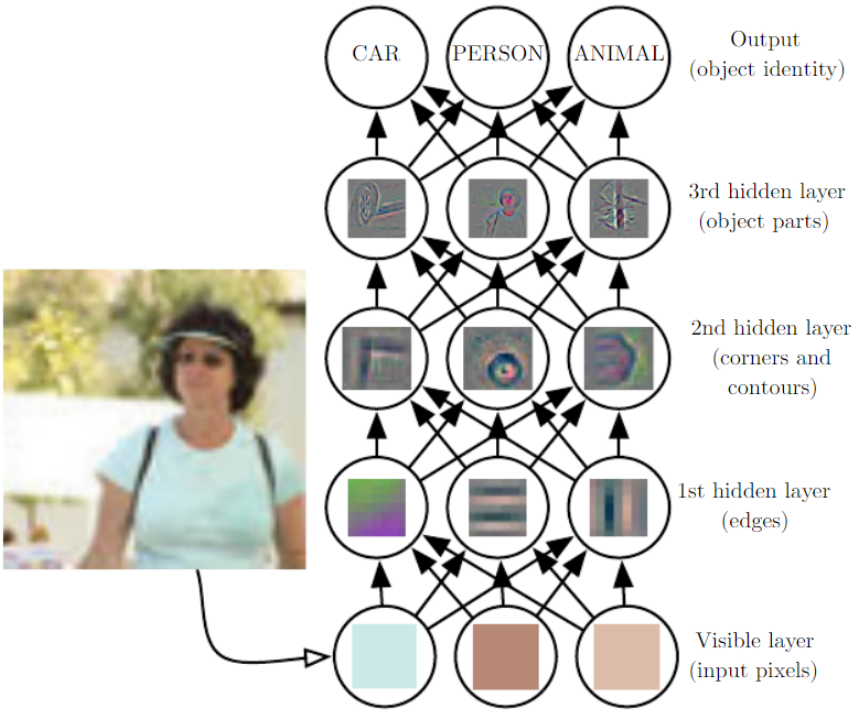


Figure 2.1: An illustration of the deep learning process for image recognition [29].

2.1.2 Transfer Learning

After a general introduction to deep learning, this subsection highlights the technique of transfer learning. First, we give a general definition. Afterwards, the possible scenarios as well as the potential benefits are presented.

2.1.2.1 What is Transfer Learning?

The field of deep learning covers an extremely broad spectrum. Especially of importance for this thesis is a technique called transfer learning. In traditional supervised machine learning there is a defined task the developers want the network to perform. For that task, a huge amount of labeled data is needed in order to train the network. This labeled data originates from a certain place, a so-called domain. Without a decent amount of input data, the network is most likely not able to modify its weights well enough to reach a certain accuracy. In particular, this problem exists when using a random initialization for weights and biases at the beginning of the training phase of a network. The left hand side of Figure 2.2 depicts how the network converges to a solution with no prior knowledge about the problem statement. Clearly, the algorithm needs many backpropagation steps to tune the weights into the right direction. For the tuning of the weights to function properly, a big dataset is necessary in order to give the network a sufficient amount of references for the optimal solution. The usual approach is to train

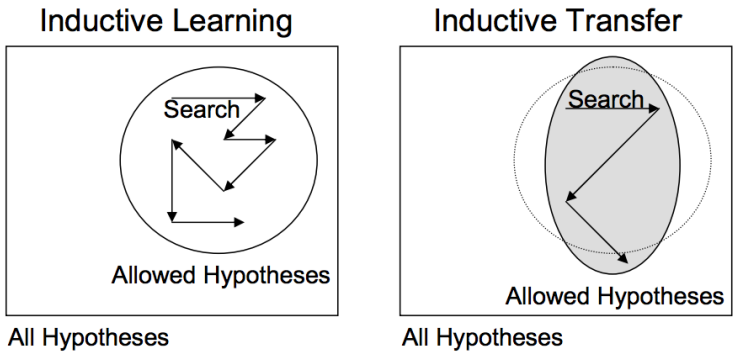


Figure 2.2: Inductive learning compared to transfer learning [11].

the network with the labeled data from a certain domain and thus make it perform as good as possible for the respective domain. This approach is optimal for classifying data from that exact domain. [62]
Although this might suffice in many cases, its benefits come to an end quickly if that same network has to perform on a slight different task or domain. For example, when a network is trained to perform on detecting people on pictures at daytime, chances are, the algorithm will perform worse by a decent amount if applied to images with people at night time. The reason for this behavior comes from the training stage. Every neuron in the network has a weight and bias influencing its output. These weights and

biases have been adapted to perform in the best way possible on the training data it has seen. Therefore, due to the bias , which is optimized on pictures of people at daytime, the network is most likely not able to generalize. Hence, it won't be able to perform decently on pictures from a new domain. [60]

This is where transfer learning comes into play. With the use of transfer learning, the problem depicted above can be solved without the need to train a new network on huge amounts of labeled data from the new domain. In transfer learning, a model A is trained on a dataset from a source domain for a source task. By doing so, the neural network gains insight on how to solve the source task and tries to maximize its performance.

The theory behind transfer learning is to not apply the whole model to the new domain or task, but to pertain knowledge about generalized features in both, the source and target task. Applying that to the problem of detecting people at night time, the goal is to store the knowledge about how to detect edges, contours and even objects in images, while discarding any learned features that specialize on a certain domain. Research showed that these features are mostly learned in the low-level layers of a network, the implementation would result in keeping these low-level layers and discarding the last layer of a network in order to keep the generalization. [60] See Figure 2.3 for a depiction of the basic transfer learning setup.

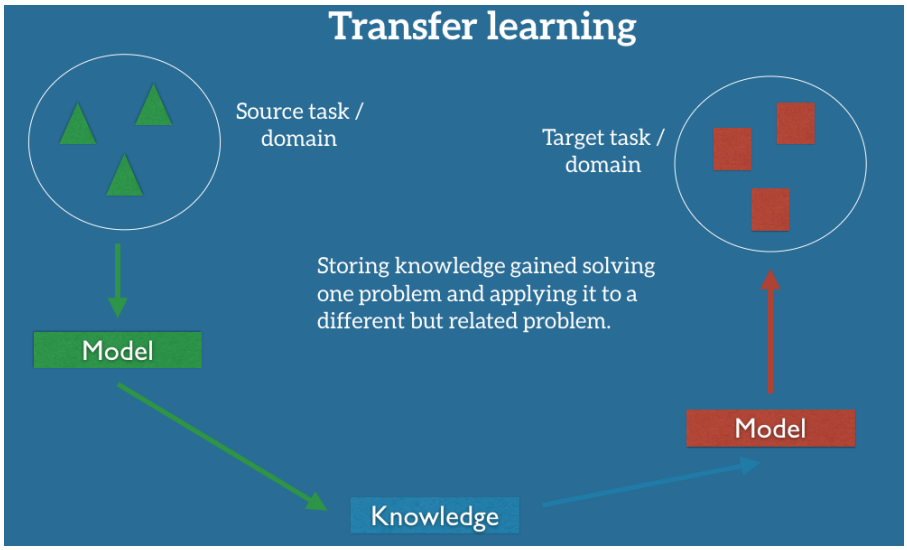


Figure 2.3: The typical transfer learning setup [60].

2.1.2.2 Scenarios of Transfer Learning

Assuming somebody considers reusing a pre-trained network with a new dataset, it is very likely to be observed that the original and the new dataset are situated in one of the

four following categories inspired by [11]. All four scenarios are depicted in Figure 2.4.

1. Scarcity of data & low similarity.
   This approach is the least promising in terms of transfer learning. As explained above, the original neural network can be interpreted as follows. The lower layers of the network serve as basic feature extractors. In image recognition, for example, in the first layers the network learns to recognize edges, corners and contours. Later layers will then learn more complex structures. At last, the output layers identify objects on the images.
   When having a dataset in this state, the only possibility to reuse the original network is by trying to freeze the first layers and to fine tune the more advanced ones. The theory behind this approach is to retain the capability of extracting the features that are similar in the original and the new datasets. Transferred to the image recognition example, to detect objects, every image classifier needs to learn how to recognize corners and edges first. The goal is to keep this skill for the new dataset. This option though, as mentioned above, is not known to promise very good results.

2. Scarcity of data & high similarity.
   In this scenario, the new dataset is rather small and thus its size does not suffice to reach a decent accuracy by training a neural network from scratch. But since the new dataset and the original input to the pre-trained network are highly similar, it is plausible that by adjusting the output layers of the original network the accuracy on the new dataset will be on a decent level. In this case, the lower layers of the original network serve as a feature extractor for the general problem statement that confronts both of the datasets.

3. Richness of data & low similarity.
   If the new dataset contains a big amount of input-output pairs which are highly dissimilar compared to the original dataset, training a neural network from scratch would be the better approach. There are two main reasons for this. First, in this situation, the network can learn all the needed features from the big new dataset, thus there is no need to inherit these skills from a pre-trained algorithm. Second, as the datasets do not have a lot in common, spoiling the network with the original dataset would not help with converging to the correct solution for the new dataset, as the weights will adapt into the wrong direction.

4. Richness of data & high similarity.
   Having a big dataset which is fairly similar to the original dataset is obviously the optimal scenario. This state indicates that there is no need to discard any of the previously acquired skills of the algorithm as the features between both datasets are similar. In order to ensure a high accuracy with the pre-trained model all the layers should not be changed. In this case, the only modification needed to apply the neural network to the new dataset is to keep the layers but to fine tune the

pre-trained model of the algorithm in order to minimize the original influence of the dataset on the weights and biases. Fine tuning means to reload and retrain the pre-trained network with the new data. [11]

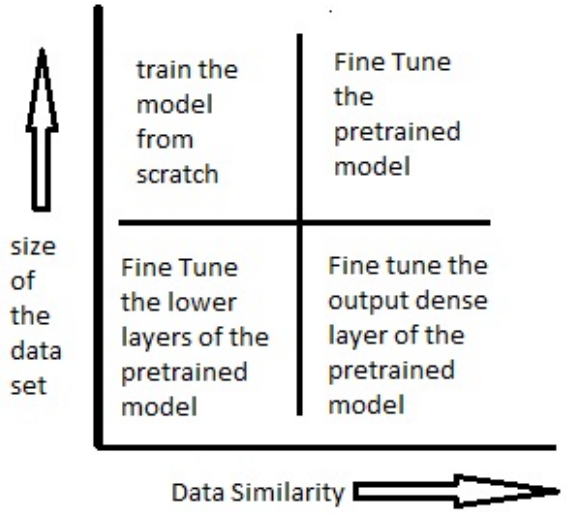


Figure 2.4: The typical transfer learning scenarios [11].

2.1.2.3 Benefits of Transfer Learning

When considering the use of transfer learning, there are several benefits that can occur by fine tuning a network for a target task. As stated in [52], there are three potential benefits of transfer learning.

- Better start. The initial performance of the network is higher in a way that, with respect to the used metrics, it provides higher accuracy in its first iteration. This is an expected behavior because the weights of the model are already adapted towards the source dataset. Since the similarity is usually high when using transfer learning, the optimal weights for the target task are close to the learned values for the source task.
- Steeper slope. In this case, the algorithm improves faster in comparison to not using a pre-trained algorithm. Faster improvement means peaking its accuracy in lesser time.
- Higher asymptote. The maximum accuracy that can be achieved by the network using a pre-trained network outperforms the potential peak without the use of transfer learning.

In the optimal case, all three benefits occur when using transfer learning. Figure 2.5 visualizes the two learning curves with and without transfer learning, respectively. For transfer learning to succeed, i.e. to trigger some or all of the benefits stated above, it has to be ensured that the involved datasets fulfill the requirements explained in subsection 2.1.2.2.
If the datasets are not rich in size or are not similar with respect to the above combinations, performance might even decrease. In these scenarios, transfer learning techniques should not be employed.

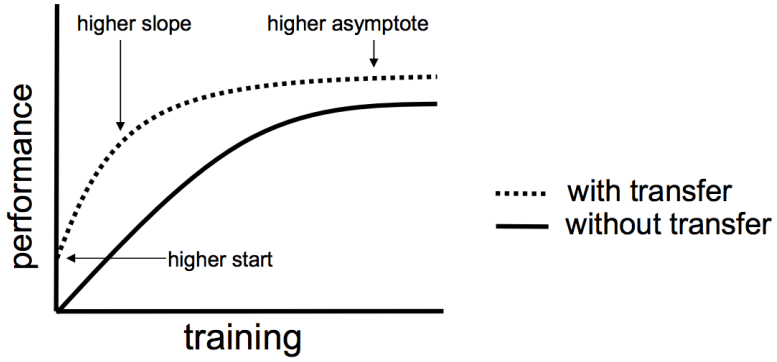


Figure 2.5: The potential benefits of transfer learning [4].

2.2 Natural Language Processing (NLP)

The expression of natural language processing is around since approximately the 1950s. It is a method to modify unstructured information written by humans in a way so machines can understand it. Natural language processing is "the field of study that focuses on the interactions between human language and computers" as stated by Matt Kiser in [4]. By implementing such an interface between the human language and computers, machines are able to understand the way humans speak. Hence, NLP enables the possibility to build real-world applications that solve many complex tasks.

Sounding rather simple in theory, the actual implementation of natural language processing is a difficult task. On the one hand, human language is unstructured, not plain and often inaccurate. More frequent than one would expect, situations occur where even people can not extract the intended meaning of a sentence. The technical term for such sentences is called syntactic ambiguity. To outline the difficulty of the realization of the NLP interface, I will state a few examples for ambiguous sentences (taken from [51]):

- "The professor said on Monday he would give an exam."
- "The burglar threatened the student with the knife."

Although certain meanings can be concluded from reasoning, i.e. interpreting the second sentence as "a burglar with a knife threatened a student" is more likely than "a burglar threatened a student that was in possession of a knife", the reader can not be entirely sure what the author's intended meaning was. See Figure 2.6 for another example on how ambiguous natural language can be. Obviously, it is even harder to implement such a knowledge for a machine. Nonetheless, there are many state-of-the-art algorithms that perform pretty well on certain tasks. Namely, the main tasks in natural language processing are sentence segmentation, tokenization, stemming and named-entity recognition. Sentence segmentation finds sentence boundaries like start and end in a given text. Tokenization subdivides a sentence into different words and numbers. The stemming process strips words to its *stem*. For example, *drinking* would be striped to *drink*. Named-entity recognition is also further explained in 2.3. A system with that capability is able to identify entities of a certain type, e.g. persons or locations. [26] Examples for real-world applications of natural language processing are automated text summarization, extracting sentiments from a text and word stemming [38].

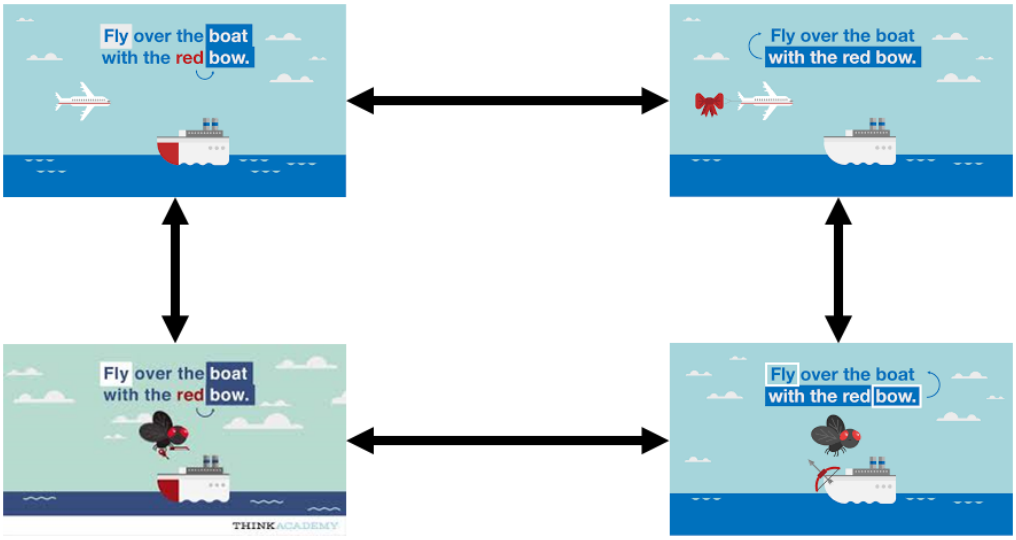


Figure 2.6: Ambiguity in NLP adapted from [35].

2.3 Named-Entity Recognition & Linking

Named-entity recognition as well as named-entity linking are sub domains of information extraction. Both of these tasks are used to extract added value from semi-structured text. The desired output is linked data that provides additional information about certain keywords in a document [65]. In the following, the core concepts will be explained further.

2.3.1 Named-Entity Recognition

In the context of natural language processing, the process of applying named-entity recognition can be broken down into two consecutive tasks.

1. Detect named-entities.
   As identifying such entities in continuous text is a complex task for a computer, approaches mostly use an ontology. This ontology includes semantic information about named-entities. Since precision is a very important factor especially in the legal domain, the used ontology needs an even higher coverage than for other fields. An example for semantic information could be the indication that the occurrence "Court" refers to an "organization" or that the occurrence "stole from the store" may refer to a crime. [5]

2. Label the found entities with a tag.
   After finding the occurrences of the named-entities with the help of ontologies, these still need to be classified and labeled according to the reference they represent. Taking the example from above, every occurrence needs to receive the correct tag, meaning every occurrence of the word "Court" is labeled with the tag "organization". [56]

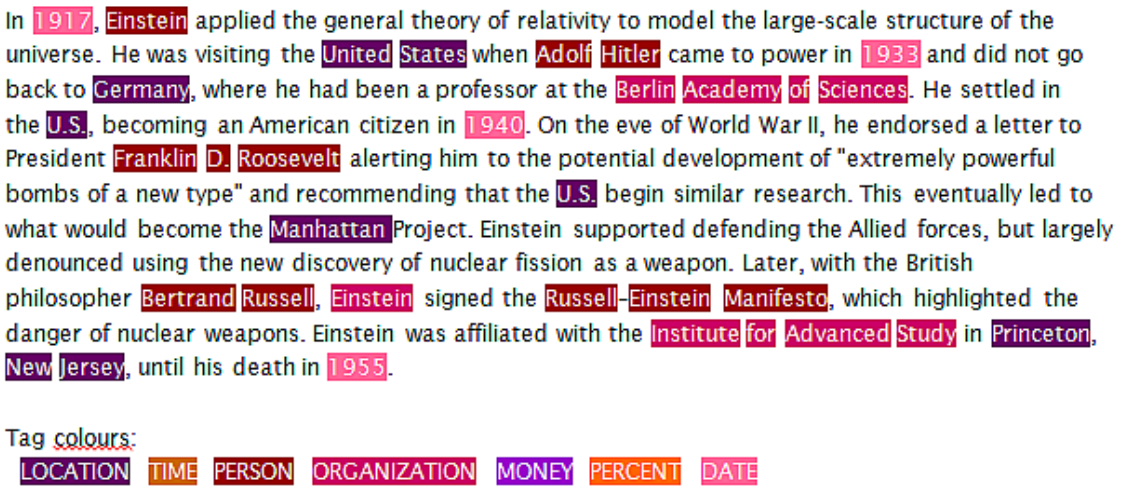


Figure 2.7: Example of a Wikipedia article for Albert Einstein, tagged with the Stanford NER tool [47].

The task of named-entity recognition is always the preceding task for named-entity linking. This technique can be very helpful for automatically creating datasets needed for solving the task of named-entity linking. A typical result of applying named-entity recognition to text is shown in figure 2.7. This example was created by the StanfordNER tool, implemented by the Stanford NLP Group [20]. Since the scope of this thesis focuses

only on the disambiguation task, named-entity recognition will not be evaluated in this work.

2.3.2 Named-Entity Linking

The task that will be explained in the following section has many different expressions throughout the state of the art papers. Some define it as *named-entity linking or disambiguation*, others call it *co-reference resolution* and in rare cases it is referred to as *wikification*. In this thesis the task of linking mentions to the correct entity will be represented by *named-entity linking (NEL)* from here on, as this term provides the most expressiveness. The concept of named-entity linking can be broken down into several sub-tasks.

2.3.2.1 Setting up a Knowledge Base

The first step in NEL is always to set up some sort of knowledge base (KB). A KB is a database that contains structured information about every possible entity that should be known by the NEL system that will be implemented for a certain task. For an example of how such a KB might look like and what kind of data it contains, see Figure 2.8. Only with a knowledge base containing a sufficient amount of information about each entity, occurrence of such entities can be found an linked respectively. [56] Setting up the knowledge base is the foundation for implementing a NEL system. Having created the structured database, one can now focus on the actual task of named-entity linking.

2.3.2.2 Disambiguate Entities & Link to KB

Before the found entities can be linked to entries in the knowledge base, a disambiguation step is needed. According to [56], this includes three main challenges that have to be overcome.

- Dissimilarities in Name. Although it might be perfectly clear to the human eye that *United States of America, USA or the States* all refer to the United States of America, a system must be capable of handling these variations in an occurrence string.
- Entity Ambiguity. Many words have multiple different meanings. Take the word *break* as an example. *Break* has many different meanings. E.g., on the one hand, it can refer to taking a break from work, whereas a synonym would be a recess. On the other hand, it could also stand for the breaking of a bone, a fitting synonym here would be a fracture. Taking into account the word context around the entity mention, the NEL system must be able to identify the correct meaning for such an ambiguous entity.
- Entity Absence. By working with huge corpora, the chance of having a decent number of entities without any entry in the KB is rather high. The named-entity linker also needs to have a fallback for that problem.

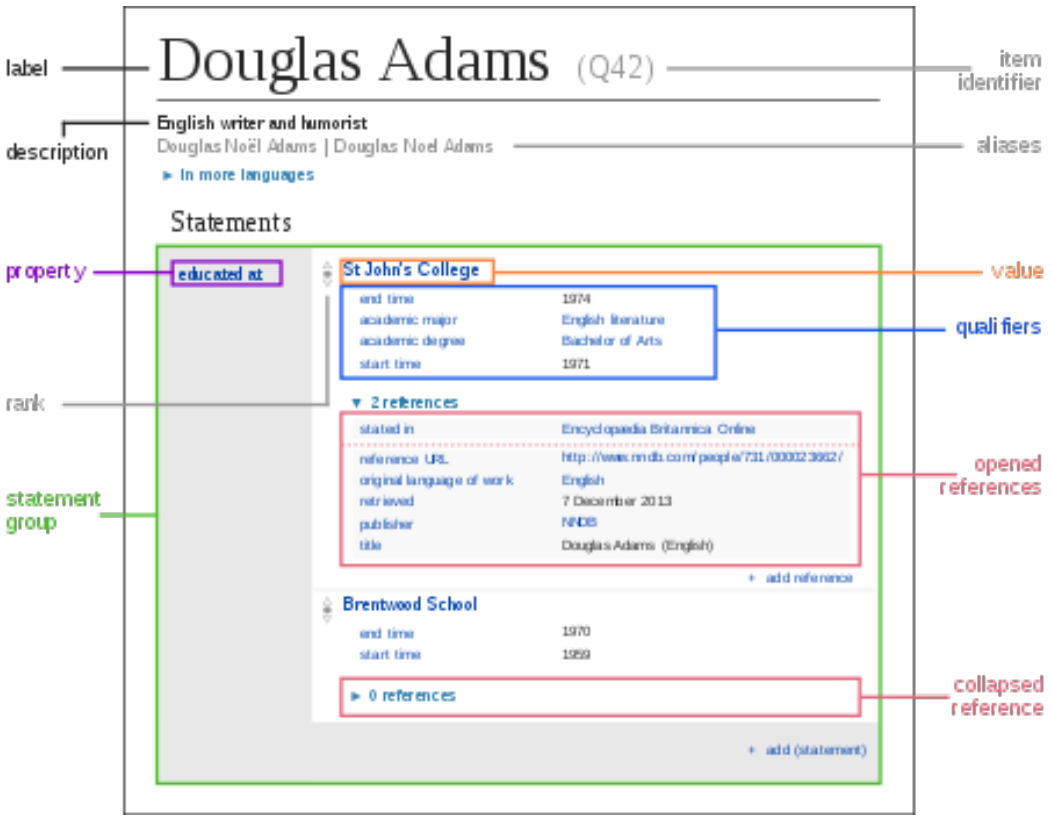


Figure 2.8: An example of the structured information contained in a knowledge base [39].

2.3.2.3 Summary

Compressing the above explanation of named-entity linking, the process can be summed up by three statements.

1. Detect mentions by determining linkable phrases.
2. Link generation by identifying candidate entities for each mention.
3. Disambiguation of entities through the context of the respective occurrences.

How state of the art NEL systems cope with these challenges and what accuracies they achieve is explained in chapter 3.

3 Related Work

This chapter presents the state of the art in the involved topics of this thesis. First, we highlight the progress of transfer learning in general. Afterwards, we outline how prior research addresses the problem of named-entity linking with a special focus on deep learning and transfer learning, as well as the legal domain.

3.1 Transfer Learning

The idea behind transfer learning is to reuse knowledge that has once been gained by a neural network on a new task. As explained in subsection 2.1.2, this can be especially helpful in scenarios where only little data is available.
Sawada et al. [62] point out that in the state of the art in supervised transfer learning [34, 59, 53], most researchers take a similar approach. First, a base model is constructed that has been trained on source domain data using a source cost function. Then, a second model is implemented based on the target domain data. This time, the hidden layers from the original model are used as initial values and the output layer is replaced. Additionally a second cost function for the new model is applied. They furthermore state that this approach in fact outperforms non-transfer learning when the source and target domain are similar. Though, it can appear that a neural network will perform poorly with respect to classification. This can be caused by training the output layer on small amounts of data compared to the original datasets, as it tends to overfit on the new training set. Figure 3.1 depicts a concrete example of an implementation of transfer learning for a convolutional neural network implemented by indico IO [45]. The visualization highlights the core of transfer learning. In particular, the convolutional neural network is trained on a source task with an original classifier. Everything is then applied to a new input, the developers only changed the classifier. Sawada et al. [62] try to prevent the overfitting by also taking the output layer into account when applying transfer learning. For the transferred output layer, a new cost function is calculated. The difference is that this new cost function is not constructed based on the target domain, it rather consists of two cost functions that get linked by evaluating the relationship between source and target data. In the case of [62] the source data originated from publicly available image recognition datasets, like MNIST [40] and CIFAR-10 [43]. The target domain dataset was generated from two-dimensional electrophoresis images, that can indicate diseases of patients. The resulting model of applying deep neural networks to the images achieved an accuracy of over 90%. [62]

Further approaches make use of transfer learning techniques by transferring the knowledge gained from simulations to real-world applications, for example for autonomous driving [73] or for robotic movement [61]. In case of Google's translation system [36], they transferred knowledge across different languages and therefore only had to train a neural network on a single language. The results are extremely good, approved by the vast number of people using their live system[1] as a translator.

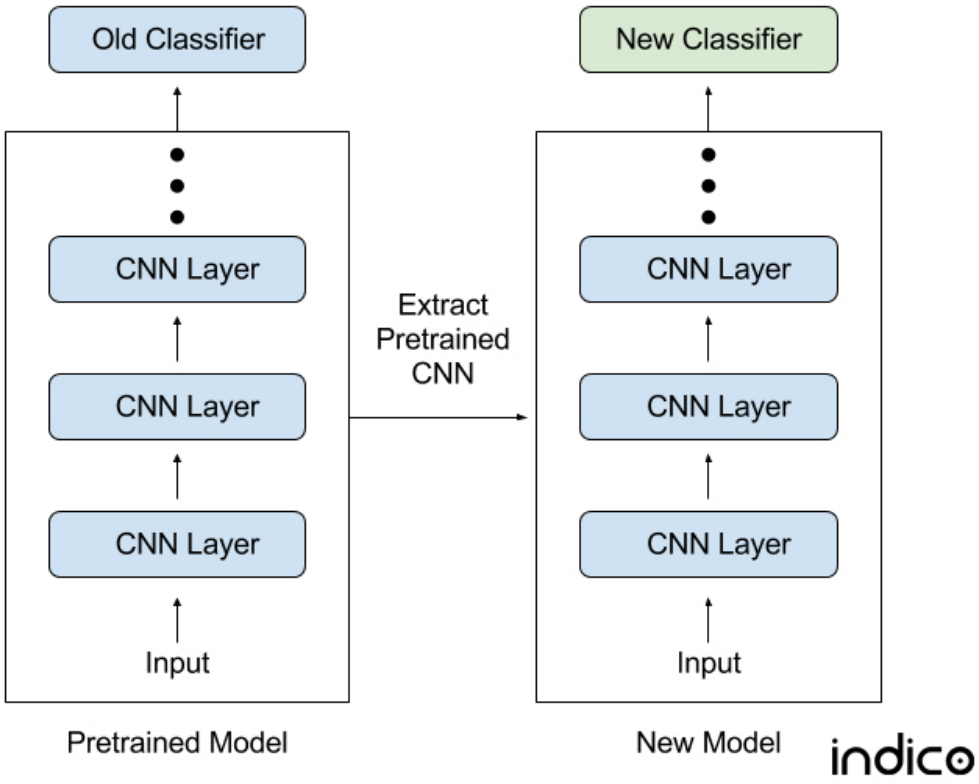


Figure 3.1: The Indico IO transfer learning approach on convolutional neural networks [45].

3.2 Named-Entity Linking with Transfer Learning

Over the last years, multiple studies [21, 69] focused on the task of named-entity linking as a subtask to natural language processing. Avirup et al. [64] point out that most of the approaches use Wikipedia as a knowledge base in their work. Especially when creating an named-entity linking system for a certain domain, expert knowledge from that area is a huge advantage. Only a small part of that expert knowledge, which would

[1] https://translate.google.com/

be needed to reach higher accuracies in many cases, can be found in Wikipedia. Thus, Avirup et al. [64] present another approach called Open-DB NED. They use a domain independent, database-driven feature generation meaning the features of the algorithm will be adapted depending on the current knowledge base. This approach defines different count functions as features which can be used independent of the domain and the knowledge base. Their counts are based on the attributes of the different entities as well as the similarities between entities. [64]

Another approach provides a language independent entity linking. Radu et al. [65] derive features, that are able to discriminate between entities across languages. They state that all their features base on a similarity measurement between the context of the mention and the correct entity. Furthermore, Radu et al. indicate that features such as a category count, i.e. for an entity like *Lincoln*, count occurrences in context of words like *cities in Nebraska*, can be applied in other languages without retraining the algorithm. The presented results in [65] show, that their system trained in English, still outperforms other state of the art systems when tested in Chinese and Spanish.
Though numerous named-entity linking approaches exist that make use of some sort of transfer learning, a successful approach using deep learning techniques for transfer learning on a named-entity linking system is yet to find.

3.3 Named-Entity Linking in Law

The success of named-entity linking is highly dependent on domain specific knowledge. Making a machine learning algorithm learn that domain specific knowledge, it needs a huge amount of training data. In the legal domain, only few corpora exist, most of them relatively small in size. [27] The scarcity of data is one of many reasons why named-entity linking in terms of law has not gotten much attention compared to named-entity linking in other domains. The only notable approach that is presented here is a low-cost, high-coverage system for named-entity recognition and -linking for legal texts. The authors Cardellino et al. [5] implemented a system for named-entity recognition, classification and linking. While they focus on a legal corpus for the recognition and classification task, entity linking was only evaluated on a Wikipedia corpus. Therefore, they also did not entirely tackle the problem of named-entity linking in the legal domain.

3.4 Candidates for Thesis

None of the above candidates is adequate for this thesis. The approaches that already use transfer learning don't seem to use deep learning techniques to employ the knowledge transfer. As this is a prerequisite for the thesis, we had a look into the research section named-entity linking with deep learning. We narrowed down our candidates for transfer learning with deep learning to three competitors. Two of these named-entity linking systems are explained in this section. The third and dedicated candidate chosen

for this thesis is explained in subsection 4.1.3. The first candidate uses convolutional neural networks to capture semantic similarity for entity linking [21]. The authors feed three types of information from the input document and two types of documents from the respective entity through their convolutional network. Francis-Landau et al. [21] took into account the mention, its immediate context and the entire document as a source document, and the entity title with its Wikipedia article as a target entity link. The output of this stage is a vector that is informative about the relatedness between a source document and a potential target entity. This vector is then fed to a sparse model that calculates possible entity candidates and chooses between them with the help of the output of the convolutional neural network. [21] The stated accuracy on baseline datasets is significantly lower than current state of the art systems. Therefore we do not consider this our preferred competitor.

The second approach implements a named-entity disambiguation system for noisy text. Eshel et al. [15] developed an entity linking system for a noisy dataset and state that this task is more challenging than creating a system for the state of the art datasets, like AIDA-CoNLL. They state the goal to capture the noise around the local context of a mention through a neural network. Although they perform extremely well on the WikilinksNED dataset created in the scope of the paper for the approach, they can not reach state of the art results on baseline datasets used for comparing named-entity linking systems. A depiction of their implemented model is shown in figure 3.2. The color associations show how the network interprets the mention and its surrounding context. The network takes into account the left and right context of a mention, as well as a respective candidate list. With this information, the network is able to output its best predictions of entities. Finally, a classifier chooses the best result of the outcome and outputs it as the correct entity. Although Eshel et al. [15] do outperform the approach presented above, they are still lacking performance compared to the state of the art. Therefore we exclude both of these approaches from further usage.

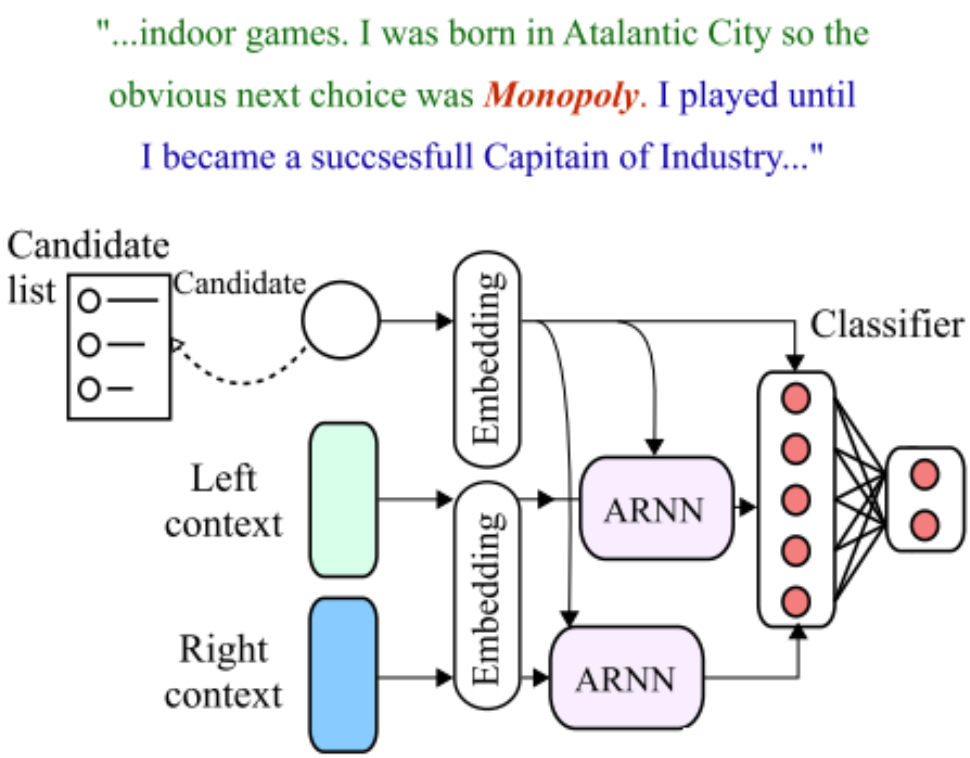


Figure 3.2: Named-entity disambiguation system for noisy text [15]. ARRN represents a recurrent neural network that uses gated recurrent units to associate candidates and context word vectors correctly.

4 Implementation

The previous chapter outlined the conceptualization of this thesis. This chapter demonstrates the necessary practical steps to realize the concept and how they were dealt with.

4.1 Preliminary

Before transfer learning can be implemented, there are a few prerequisites that need to be fulfilled. This section outlines the requirements for this work to make transfer learning for named-entity linking possible.

4.1.1 Datasets

This section will outline the details about the datasets that are involved in all experiments. First, I will give some insights on what publicly available datasets look like. Afterwards, the details about the generation of the EUR-Lex dataset, created in the scope of this thesis, are presented. Statistics about these datasets can be found in Tables 4.1, and 4.2. Furthermore, this section defines the architecture of the selected implementation and shortly presents the respective results.

4.1.1.1 Public Benchmark Datasets

The following datasets are used in state of the art named-entity linking systems to measure the performance of the developed algorithms. They are publicly available and mostly manually annotated.

1. AIDA-CoNLL [33].
   This dataset is widely used for public benchmarks on the entity disambiguation task. In terms of entity disambiguation, the AIDA corpus is the biggest manually annotated dataset available. It is divided into three parts, AIDA-train, AIDA-A, denoted as the validation set, and AIDA-B acting as a test set.
2. WNED [1] [30].
   This is a collection of datasets automatically created from the Wikipedia corpus. The included datasets are also used for public benchmarks, but since they are generated automatically and hence less trustworthy, they are mostly used for testing a named-entity linking system, not for training it.

[1] http://lemurproject.org/clueweb09/FACC1/

Dataset	Number Mentions	Number Documents	Gold Recall
AIDA-train	18,848	946	100%
AIDA-A (valid)	4,791	216	96.9%
AIDA-B (test)	4,485	231	98.2%
MSNBC	656	20	98.5%
AQUAINT	727	50	94.2%
ACE2004	257	36	90.6%
WNED-CWEB	11,154	320	91.1%
WNED-WIKI	6,821	320	92%
EURLEX-train 1k	1,853	1,118	87%
EURLEX-test 1k	333	185	87%
EURLEX-train 20k	33,937	17,352	87%
EURLEX-test 20k	11,674	4,580	87%

Table 4.1: Entity Disambiguation Datasets. *Gold Recall* represents the percentage of mentions which have the ground truth entity in their respective candidate set. This table was adapted from [23].

3. MSNBC [9], AQUAINT [49] & ACE2004 [57].
   All three of these datasets are taken from different news corpora, all written in English. The news datasets are small in comparison to the AIDA-CoNLL datasets but are also manually annotated.

4.1.1.2 EUR-Lex

This dataset was created from the corpus publicly accessible on a sub-page of the official homepage of the European Union[2]. The website provides free access to multiple topics including for instance the so called *Official Journal of the European Union*, EU law with subjects such as regulations and directives or summaries of legislations. [18] For the scope of this thesis, only documents from the journal and the EU law were used. To be able to use the content of the EUR-Lex website, important data with respect to named-entity linking had to be extracted. The following steps were involved in the extraction of the necessary data.

1. Download documents in HTML format from the EUR-Lex website and extract the content into separate text files. A python framework has been used to easily process all necessary information in the relevant HTML documents. For further details on the framework, see 5.1.

2. Load content from text files and send them to the application programming interface (API) of *Dandelion's* named-entity linking system. [10] The response from that API contains annotations with the respective offset in the original text together

[2] http://eur-lex.europa.eu/homepage.html

with the ground truth entity. The accuracy provided by the API on the EUR-Lex data was taken as the gold recall rate in table 4.1 for this dataset.

3. Store response in a format similar to public benchmarks datasets. In order to ensure the usefulness of the automatically created dataset for the task of named-entity linking, the dataset is stored similar to the format of the WNED datasets. This format can be seen in Figure 4.1. Each dataset folder consists of one folder containing the text files to be parsed and one XML file with the annotations for the text. This XML file consists the following information about each mention.
    - Mention: the mention for which an entity exists in the Wikipedia knowledge base.
    - Wikiname: the name of the entity in the Wikipedia knowledge base.
    - Offset: the offset of the first letter of the mention in the respective document.
    - Length: The number of characters of the mention.

After the generation of the dataset, we divided it into two parts. One contains the annotations of all documents that were processed from the EUR-Lex corpus, the other contains annotations from only a restricted set of approximately 1,000 documents. As the entire dataset was generated from roughly 20,000 documents, it will from now on be referred to as EUR-Lex 20k, whereas the smaller dataset will be referred to as EUR-Lex 1k. As stated in Table 4.1, the EUR-Lex 20k corpus dedicated for training contains more entities than the original training set from AIDA-CoNLL (compare figures for AIDA-train and EUR-Lex-train). This means, in this case we actually have more data for the new task in comparison to the original task the algorithm a was developed for. This indicates that transfer learning is not the desired approach considering this legal dataset. As one of the objectives of this thesis is to investigate possible benefits of transfer learning in named-entity linking, a smaller dataset was created in order to simulate the case where one does not have as much data.

Dataset	Total Number	Number of frequent pairs (>50)	Number of unique pairs
AIDA-train	4,012	35	1,775
AIDA-A	1,642	5	877
EURLEX-train 20k	4,251	83	2,286
EURLEX-test 20k	987	55	524

Table 4.2: Frequent mention-entity pairs in the EUR-Lex and AIDA datasets.

4.1.1.3 Merged Dataset

Last but not least, we added an additional dataset to increase the variety of experiments and to be able to draw further conclusions. This dataset consists of the AIDA-CoNLL and the EUR-Lex corpus combined into a joint train and test dataset. The training set contains 52,785 entities whereas the test set consists of 16,465 entity mentions.

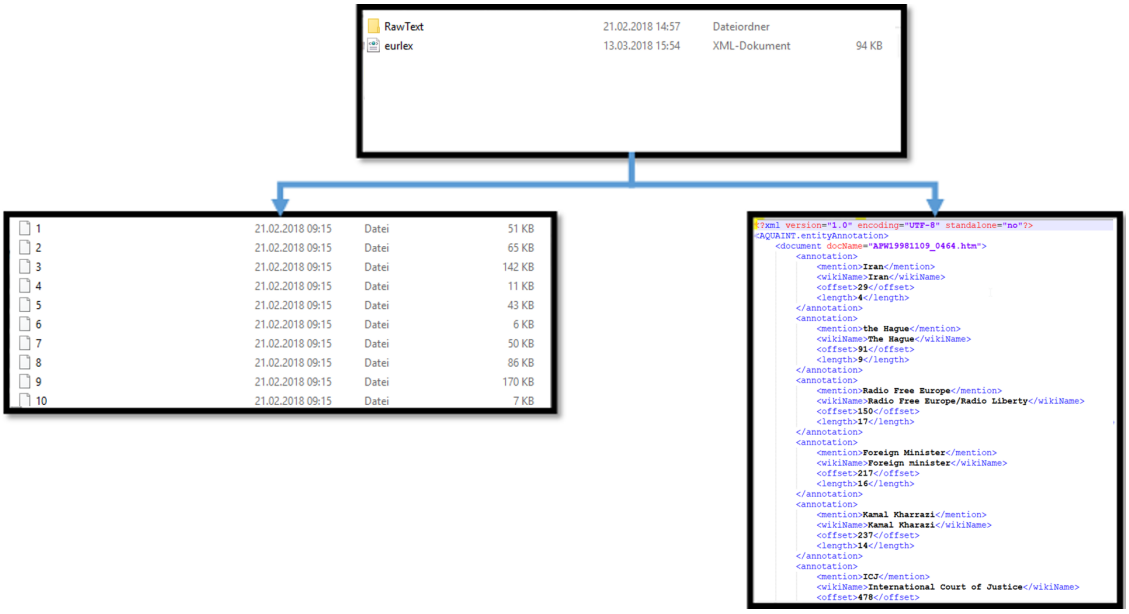


Figure 4.1: Format of the public benchmark datasets in the WNED collection.

% of similar entities	Aida-train	Aida-test	EUR-Lex-train	EUR-Lex-test
Aida-train	100	-	-	-
Aida-test	53	100	-	-
EUR-Lex-train	7	4	100	-
EUR-Lex-test	20	13	70	100

Table 4.3: Similarities between the datasets.

4.1.2 Choose a Named-Entity Linking System

As described in subsection 2.1.2, transfer learning in terms of deep learning is not about creating a neural network architecture from scratch for a specific dataset, which would then be tailored for the specific problem. It rather focuses on re-using a model that has been created for and trained on another dataset, whose domain is in some sort similar to the new dataset.

Hence, the focus of the thesis is to find a state of the art deep named-entity linking system. Afterwards, the goal is to apply transfer learning to that system to be able to re-use it for another dataset.

This section presents how such a system has been chosen. First and foremost, the machine learning metrics *accuracy* and *F1 score* played an important role in choosing an adequate deep named-entity linking system. A definition for these metrics as well as a justification for employing them can be found in subsection 5.1.3. Searching in the pool

of state of the art named-entity linking models, with the criteria defined above, leads to numerous projects that have been made open-source. Another requirement that must be fulfilled by the desired network of choice is to use deep learning. Refining the search with the machine learning metrics and the restriction to deep learning architectures, the number of possible systems reduced drastically. Only one competitor satisfied all requirements to be adequate for the scope of this thesis.

4.1.3 Deep Learning Architecture

The architecture of the chosen implementation will be explained in the this section. The paper for this approach [23] was published in the scope of the *Conference on Empirical Methods in Natural Language Processing (EMNLP)* [3] in 2017. The developers present a new deep learning model for named-entity disambiguation. According to the authors, it combines aspects from two research perspectives. The approach combines benefits of deep learning with traditional machine learning experiments. Their developed architecture achieves F1 scores of state-of-the-art systems and even outperforms many approaches.

4.1.3.1 Overview

The input for the architecture implemented in [23] needs to be provided in the following format.

- Document name: The name of the document a mention occurred in.
- Mention: The mention that has to be linked to an entity.
- Context: The left and right context window of 100 words around the mention.
- Candidates: The set of entity candidates for the respective mention.
- Ground truth: the correct entity from the candidate set for the mention.

The model consists of the following three different parts.

- Entity Embedding.
  The entity embedding contains the semantic meaning of entities. For each entity, the embedding is inherited from pre-trained word embeddings, for example *GloVe* [54]. In this approach, the authors embedded both, words and entities, together. This gives the opportunity to calculate geometric similarities between words that surround a certain entity and the entity itself.
- Local Model with Neural Attention.
  Encouraged by prior work [42], the authors assume that only some context words are important for the disambiguation step of a mention. For each entity with the

[3] http://emnlp2017.net/

respective context words, a context score is calculated, ranking the importance of each context word with respect to the current entity. To remove some noise, the algorithm only considers the top 20 words from the local context. The goal for the local model is then to find a function that assigns a higher score to the correct entity that to the other entity candidates.

- Collective Disambiguation.
  To put it all together, the authors propose to use a deep learning architecture. This architecture resolves ambiguous mentions using a conditional random field with parametrized potentials based on [41]. Conditional random fields provide the functionality to label and divide any kind of sequences. This approach solves the problem of mapping an occurrence into a set of possible classes. The detailed problem description and implementation of a conditional random field is explained in [41]. In the case of named-entity disambiguation this relates to associating a mention with an entity contained in a candidate set.

The combination of the entity embedding, the local model and the conditional random field then gives the deep named-entity disambiguation approach proposed in [23].

In a nutshell, the model's pipeline can be summed up by the following three key phrases.

- Create an entity look up table or map $\rightarrow$ *entity embedding*.
- Compute individual context scores for each entity $\rightarrow$ *local model*.
- From a set of candidates, select the correct entity from the look up table, taking into account the context score $\rightarrow$ *collective disambiguation*.

4.1.3.2 Experiments

The conducted experiment that was used as a baseline for state of the art named-entity linking systems is depicted in figure 4.2. The authors from [23] trained, validated and tested their model on the public benchmark dataset AIDA-CoNLL. AIDA-A,the validation set, was used for the adaption of the learning rate of the algorithm. Amongst other datasets, automatically generated from different news corpora, they tested their model on AIDA-B, which is the denoted test dataset from the AIDA-CoNLL collection. As described in [23] the performance peak with respect to the AIDA-B dataset was 92.2%. This result competes with most and even outperforms some of the systems that account for state of the art results.

4.1.4 Prerequisites for Transfer Learning

After the concept of the chosen named-entity linking system has been described, we need to modify the implementation to make it applicable for transfer learning. The

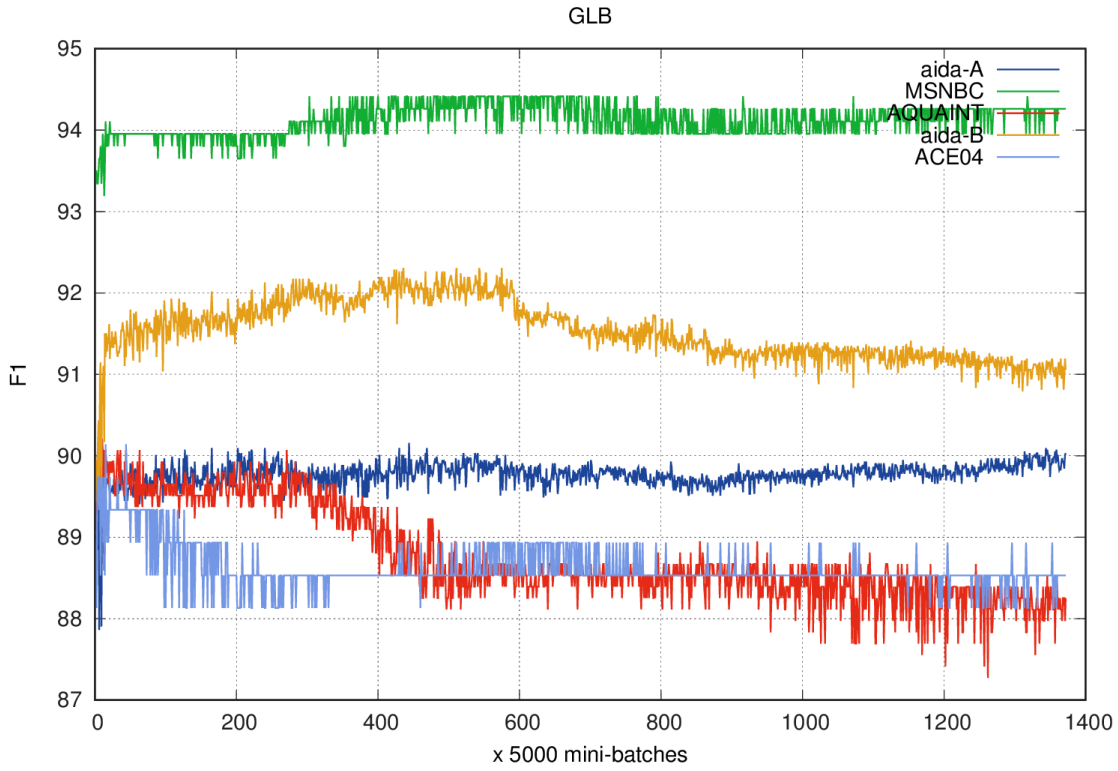


Figure 4.2: The original F1 scores for the public benchmark datasets achieved by [23].

following steps are involved in employing this technique with the network described above. The technical aspects and implementation of all involved conceptual changes are explained in chapter 4.

4.1.4.1 Train Original Model

The publicly available GitHub repository of [23][4] doesn't provide any sort of pre-trained model. Therefore, before coming to the part of transfer learning, it was necessary to train the original network. By training the model on the public benchmark datasets as described in 4.1.1, we created and saved a model that will later be loaded to apply transfer learning. The term *saving a model* always refers to the process of saving the current weights and biases of a network at a certain training progress into a file. If one wants to reuse the previously achieved results, it means that the saved weights and biases are loaded from that file and applied to the network's architecture. This pre-trained model achieved the F1 scores depicted in figure 4.2. The authors decided to save a model and to decrease the learning rate whenever the validation F1 score exceeded 90%. By having a closer look at the figure 4.2 and by analyzing the log files for the training run, it turns out that the highest validation F1 score was achieved at epoch

[4]https://github.com/dalab/deep-ed

444. Hence, whenever we talk about using the original pre-trained model, we refer to the model saved at epoch 444 in the original training stage.

4.1.4.2 Modify Preprocessing

The goal is to modify the network in a way so that it can be re-used for any new dataset the originates from a similar problem domain. In other words, we had to implement preprocessing scripts that extracted necessary information from raw text to build an entity disambiguation dataset. The public benchmark datasets explained in the following chapter served as an orientation for the necessary structure. We implemented a preprocessing pipeline in Python with the following capabilities. It is able to interpret any kind of text and to feed it to an API to gain information about possible mentions of entities from a knowledge base build on Wikipedia data in said text. Finally, the pipeline outputs a dataset according to the structure of named-entity linking benchmark datasets.

4.1.4.3 Adapt Train & Test Stage

The following conceptual changes were needed to apply transfer learning to the model created in [23].

- Redirect input pipeline. We want to form experiments on the existing architecture including a new dataset. Therefore, it is necessary to provide the opportunity to easily exchange the datasets that are taken into account by the algorithm.
- Load a pre-trained model. Instead of starting from scratch with respect to the learning progress of the network, we need to be able to load a pre-trained model. Hence, an implementation of some sort of switch is needed to indicate whether the model should be trained with or without previous knowledge. In case a pre-trained model should be used, the loaded weights need to be applied before starting the training.
- Change test conditions. Since the dataset changed, we also need no values for the metrics, that indicate whether a model performs extremely well. Thus, we need to inspect the performance of the network on the new dataset. Afterwards, we can determine a good threshold for the F1 score of the new dataset.

4.2 Preprocessing, Train & Test

Highlighted in section 4.1.3, there are multiple necessary steps involved in making the chosen named-entity linking network *transfer learning ready*. The first step is to rebuild the model with the results stated in [23]. Although the authors don't provide a pre-trained model of their architecture, a guideline on how to rebuild their best result is attached on the GitHub repository page. With this guideline and some modifications we were able to recreate a model with decent results.

4.2.1 Preprocessing

After completing the first step of rebuilding the original model, we can now focus on adapting the pipeline so that the EUR-Lex dataset is applicable for the architecture. First, it was necessary to build the legal dataset from the EUR-Lex corpus. The generation process of the dataset is explained in subsection 4.1.1.2. The EUR-Lex dataset format was inspired by the publicly available WNED datasets as explained in 4.1.1. The current preprocessing pipeline included in the GitHub repository of [23] already provides the capability to transform datasets in WNED format into an input that is accepted by the network architecture. The structure of that input is a comma-separated value file as explained in 4.1.3. Therefore, by considering the needed format during the creation of the dataset, no additional steps are needed for the preprocessing.

4.2.2 Training

First, we needed to include all new datasets in the entity embedding training. Without this step, the network will later maybe predict the correct entity from the candidate set, but cannot link it to a knowledge base entry as it has not been included in the embedding. The entity embedding serves as a lookup map for entities. This lookup map represents the knowledge base in the scenario of [23]. Figure 4.3 depicts the architecture

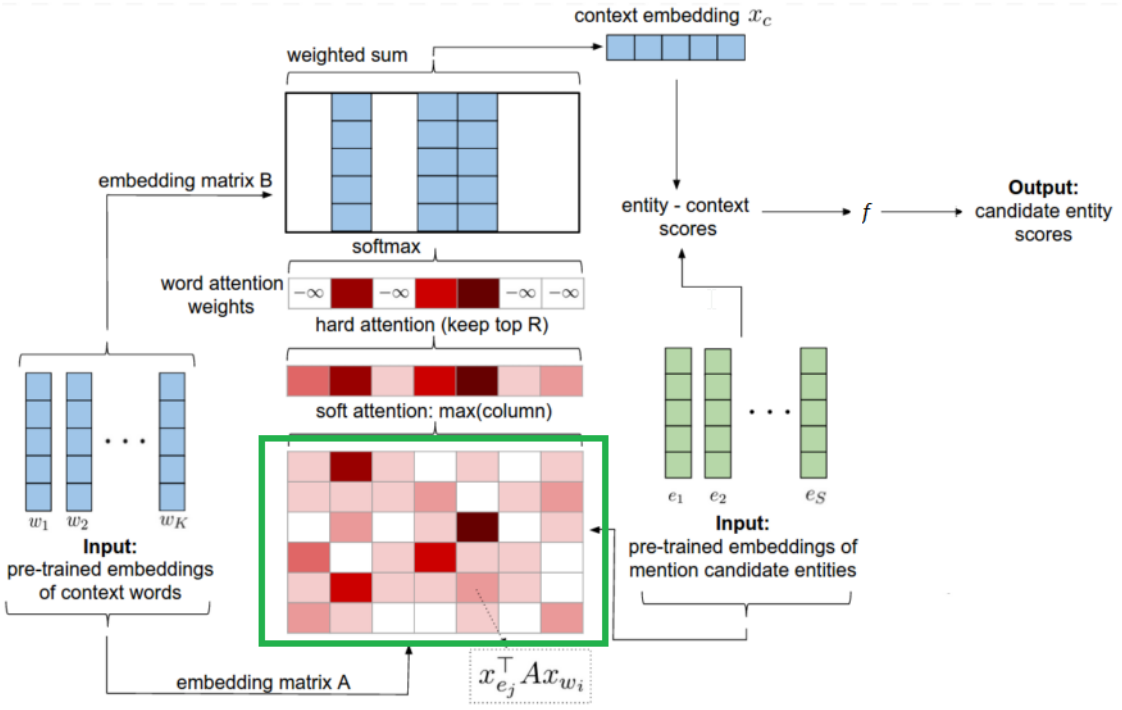


Figure 4.3: The neural network used for local context scores of entities adapted from [23].

of the local model implemented by Ganea et al. [23] and indicates (green) what part

had to be replaced by the new entity embedding. As stated by the authors, function f is responsible for determining the context scores. They furthermore indicate that a flexible choice of f is important for the performance of the model. Hence, they used a neural network for f consisting of two fully connected layers of 100 hidden units and ReLU non-linearities.

After the creation of that entity embedding, the focus was to create baseline models. With these baseline models, we were able to compare the transfer learning results to results achieved by training on a single dataset. As highlighted above, with the help of the guideline by Ganea et al. [23] and some small modifications, rebuilding the original model was relatively simple. Afterwards, the same procedure was applied to the EUR-Lex dataset.
First, the legal dataset was divided into a train and test set, similar to AIDA-CoNLL. Table 4.3 shows the overlapping percentage of entities in the respective train and test sets. The overlap of Aida-train and test is 53% while EUR-Lex-train contains 70% of the entities included in the test set. The gap of 17% comparing the two corpora is not optimal but the best that could be achieved without involving a huge amount of manual work. After the division of train and test set, the model was trained on these sets. Similar to the original training phase, the best weights with respect to the achieved F1 score on the EUR-Lex-test set were saved into a file for further usage.

The next step is to train a model on the merged dataset of EUR-Lex and AIDA-CoNLL. Prior work from the field of natural language processing did not seem to include such an approach of combining two datasets from different fields in their experiments. In our case, the combination of a well generalized, hand-crafted dataset with an automatically extracted dataset from a niche domain could force a model to learn the specialization better compared to employing a transfer learning technique.

The above steps were necessary to implement the actual focus of this work. With the saved models for single training, transfer learning could be easily applied. First the model's weights had to be reloaded and applied to the mode architecture. Afterwards, the datasets on which the model should be trained had to be exchanged to either EUR-Lex or AIDA-CoNLL, respectively. Subsequently, we could start the training process for the target dataset with the optimal weights for the source dataset.

4.2.3 Testing

Through the implementation of features that make it possible to exchange datasets, reload models and plot the results, testing on different models was straight forward. We employed F1 score and standard accuracy in the testing phase to draw conclusions on the performance of an approach. Subsection 5.1.3 provides a definition for the metrics as well as an explanation on why they have been used. Generally, we tested in three different topics: single training, joint training and transfer learning. Transfer learning

was applied on each model trained on the EUR-Lex and AIDA-CoNLL dataset. This makes it possible to conclude whether fine tuning a model trained on a generalized dataset outperforms training on a niche dataset and fine tuning with the help of a well generalized one.

5 Experimental Study and Performance Evaluation

This chapter provides detailed information on all experiments conducted in the scope of this thesis. Besides transfer learning approaches, also additional experiments have been included in order to compare each method and conclude the benefits and disadvantages of each of them.

5.1 Experimental Setup

This sections highlights the basic conditions for the execution of all experiments conducted in the scope of this thesis. This includes hardware specifications, details about the involved software, as well as the metrics used to measure the performance of the different approaches.

5.1.1 Hardware

iteratec GmbH[1], who supported me in the elaboration of the thesis, provided a machine learning computer for conducting any kind of experiments. The machine contains a NVIDIA GeForce GTX TITAN X[2] graphics card with 12 GB memory. Furthermore, it has an Intel Core i7-5820k[3] central processing unit. This CPU contains 6 cores and has a base frequency of 3.30 GHz. The RAM size of the machine learning computer is 16 GB. With this setup, the average training time for the involved datasets was 60 hours. The training process was stopped as soon as the network started dropping in F1 score and accuracy on the respective validation set.

5.1.2 Software

The technological stack involved in the elaboration can be divided into two different fields of work.

- Dataset aggregation.
  The scripts that were created to gather the necessary data from the EUR-Lex [18] corpus have been implemented in Python[4]. With the help of the python library

[1] https://www.iteratec.de/
[2] https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-x
[3] https://ark.intel.com/products/82932/Intel-Core-i7-5820K-Processor-15M-Cache-up-to-3_60-GHz
[4] https://www.python.org/

Scrapy [63] we could easily gather and download 20,000 documents from the EUR-Lex homepage. For the ease of parsing and extracting information from HTML documents, we used the python library BeatifulSoup [3]. This library provides intuitive access to the HTML content and thereby remarkably removes any complexity from the processing of relevant informations in such documents.

- Machine learning.
  The dedicated named-entity linking system [23] in this thesis implements all models used for the experiments in the Torch framework [24]. Torch is a framework for the Lua programming language[5] for scientific computing.

- Version control.
  In order to keep track of the numerous modifications implemented in the deep entity disambiguation system, we used GitHub[6].

5.1.3 Metrics

The following criteria played an important role in choosing an adequate deep named-entity linking system and in measuring the quality of our experiments.

- *Accuracy*.
  The standard accuracy is the most intuitive measurement. It provides information on the percentage of correctly classified inputs.
  $accuracy = \left| \frac{groundtruth \cap results}{groundtruth \cup results} \right|$

- *Precision*.
  Precision indicates how many of the classified mentions were correctly classified.
  $precision = \left| \frac{groundtruth \cap results}{results} \right|$

- *Recall*.
  Recall represents the number of examples with ground truth assignment that have been guessed by the model.
  $recall = \left| \frac{groundtruth \cap results}{groundtruth} \right|$

- *F1 score*.
  The F1 score is the harmonic average of *precision* and *recall*. Therefore, it acts as a good performance measurement in classification problems. Thus, it is the most common metric in named-entity linking and the generally accepted measurement for comparing the performances of different systems.
  $F1 = \frac{2 \times precision \times recall}{precision + recall}$

These metrics are the most common in terms of classification problems, especially in entity disambiguation. A definition and more detailed explanations for accuracy, F1

[5] http://www.lua.org/about.html
[6] https://github.com/

score and other classification metrics can be found in [67] and [66].

Searching in the pool of state of the art named-entity linking models, with the criteria defined above, leads to numerous projects that have been made open-source. Another requirement that must be fulfilled by the desired network of choice is to use deep learning. Refining the search with the machine learning metrics and the restriction to deep learning architectures, the number of possible systems reduced drastically. Only one competitor satisfied all requirements to be adequate for the scope of this thesis.

5.2 Results

This section presents the approach and results on the different experiments that we conducted in the scope of this thesis. For every experiment, we provide a graph indicating the learning progress of each respective model. These graphs shows the development of the F1 scores over training time. For an accumulated statistic of all F1 scores and accuracies of all experiments, see table 5.2. For an interpretation of the presented results, see section 5.3.

5.2.1 Single Training

This section includes all experiments conducted with a single dataset. This means the model has only been trained on one dataset with a respective validation set. No fine tuning of any pre-trained network is involved. Here we execute our baseline experiments, whose serve to draw conclusions on the use of transfer learning. Two experiments are involved, training a model according to [23] and applying the very same process on the EUR-Lex dataset. Interesting values with respect to performance measurements in single training are only the training and validation F1 score

- Recreate results of [23].
  As the available resources for the named-entity linker do not contain a pre-trained model, we had to recreate a network with the stated accuracies in [23]. For a description of the experiment we recreated, see subsection 4.1.3.2.

- Train on EUR-Lex. To have a broader basis for later transfer learning experiments and performance comparisons, we implemented a similar training process for the EUR-Lex datasets as we did in the original approach. Training and testing have been done with EUR-Lex only. The highest achieved F1 score by training on the EUR-Lex corpus only, was 98.43% on the training set and 98.01% on the test set. Since the F1 scores are similar, we assume no overfitting on the training set of the model. The achieved F1 scores can be found in figure 5.1.

Both of the created models in this stage have been saved and will be reused in the transfer learning step.

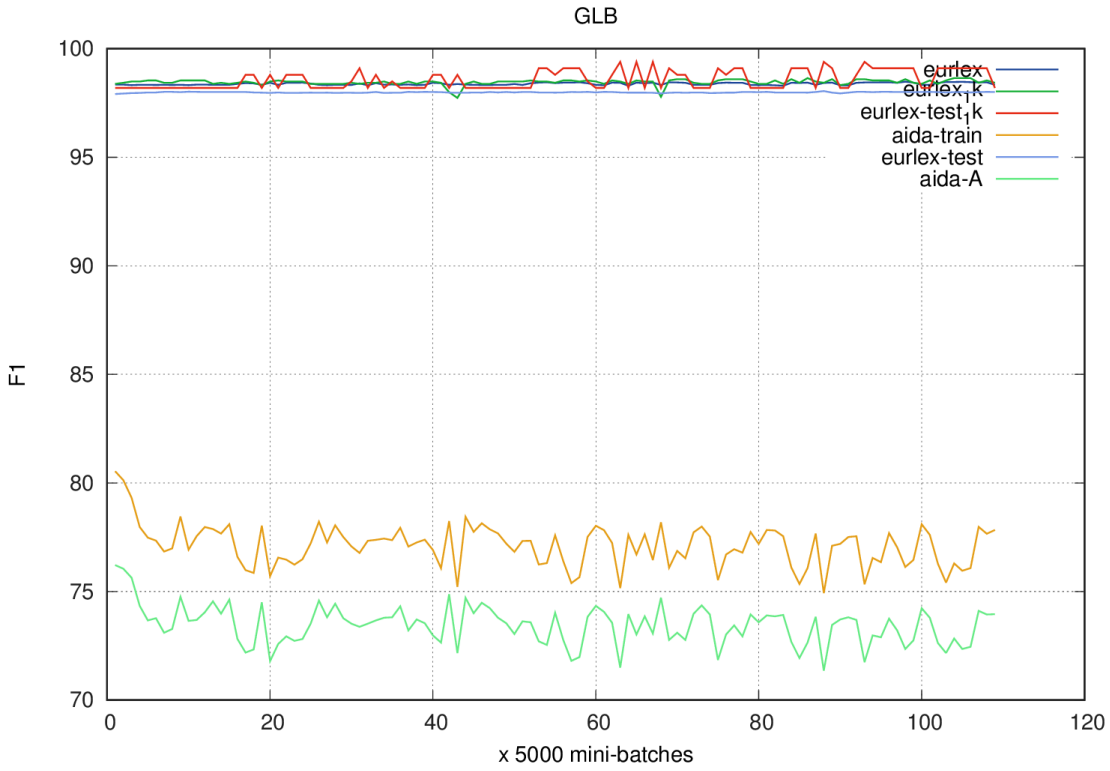


Figure 5.1: The F1 scores during single training on the EUR-Lex dataset.

5.2.2 Joint Training

As an alternative to transfer learning and single training, we introduced what is called joint training. In this approach, the training sets from the original training set, AIDA-CoNLL, and the dataset generated in the context of this thesis, EUR-Lex, are merged (details about the datasets in section 4.1.1). Afterwards, the algorithm is trained from scratch with the merged training and test set. The idea behind that implementation is to force the network to learn a broader spectrum of entity disambiguations among all datasets involved. Relevant values with respect to the datasets are the F1 scores achieved when testing on Aida-CoNLL and EUR-Lex. In case of success, the resulting network would then provide more generalization and therefore is of more use for an actual application of the knowledge. The graph of the plotted F1 scores for this experiment can be found in figure 5.2. The performance peak for EUR-Lex was observed at 97.36% for the training, and 97.19% on the test sets. Regarding Aida-CoNLL, the joint training reaches its maximum at 87.50% on the training, and 85.29% on the test set. On the joint training set, the model achieved 93.67% on the training set, whereas the test set was at 93.42%. We conclude from the similar F1 scores between the individual training and test sets, that the resulting model also does not overfit on either the joint, the EUR-Lex or the Aida-CoNLL datasets.

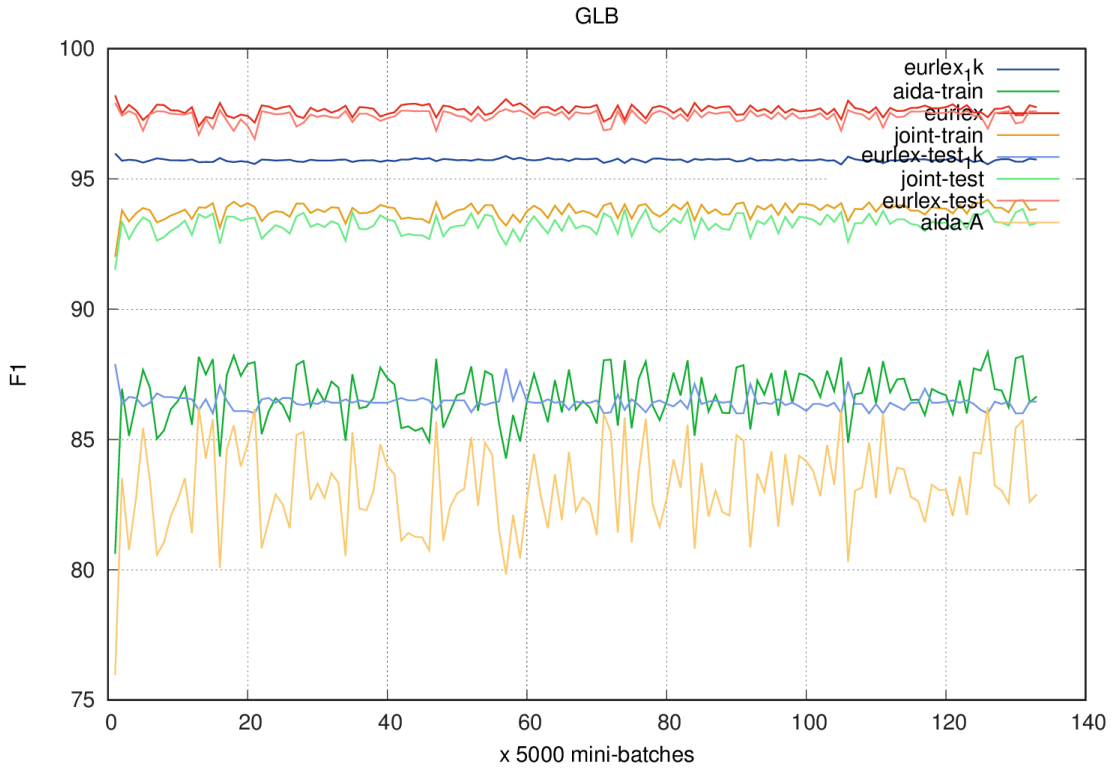


Figure 5.2: The F1 scores during joint training on the merged dataset of EUR-Lex and AIDA-CoNLL.

5.2.3 Transfer Learning

This section outlines all experiments conducted in terms of transfer learning. The involved datasets in this stage were the entire EUR-Lex dataset, as well as the smaller version and the AIDA-CoNLL train and test corpora. To figure out which of the transfer learning techniques that are depicted in Figure 2.4 should be applied, the involved datasets have to be compared. Depending on the richness of data and the similarity to the source dataset, the optimal technique can be applied in each case. Results for each of experiments as well as the chosen technique are explained in the subsections below.

5.2.3.1 EUR-Lex applied on AIDA-CoNLL

Here, the algorithm was trained on the source dataset AIDA-train. When the F1 score on the validation set AIDA-A started dropping, we stopped the training and saved the model's weights for the use of transfer learning. Afterwards, we loaded the saved weights and applied them to the model architecture. Then, the training and test sets were replaced by the ones from the EUR-Lex corpus. Subsequently, we started the training process again, now starting with the optimal weights for the AIDA corpus.

The source dataset, AIDA-train, and the target dataset, EUR-Lex-train, have a high similarity. Although only 7% of the entity-mention pairs in the EUR-Lex 20k dataset are also contained in the AIDA dataset, the majority of these similar entries occurs very frequently in each of the datasets. Additionally the size of the EUR-Lex corpus exceeds the extent of the AIDA dataset (see table 4.2). Concluding from these facts, we have a richness of data and a high similarity between the two datasets. This scenario suggests the fourth transfer learning technique explained in subsection 2.1.2.2. Therefore, we start the training without any modified layers and let the network adjust its weights in direction of the new dataset. Figures 5.4 and 5.3 show the highest F1 scores achieved when applying transfer learning on the model trained with the AIDA-CoNLL dataset. In case of the EUR-Lex 1k dataset we achieved 99.73% on the training and 98.90% on the test set. When fine tuning with the full 20k corpus, the training set peaked at 98.49% while the maximum F1 score of the test set was 98.01%. In both cases, the training F1 scores exceeded the test results. This indicates slight overfitting but the difference between the results is, in our opinion, still within reasonable margin.

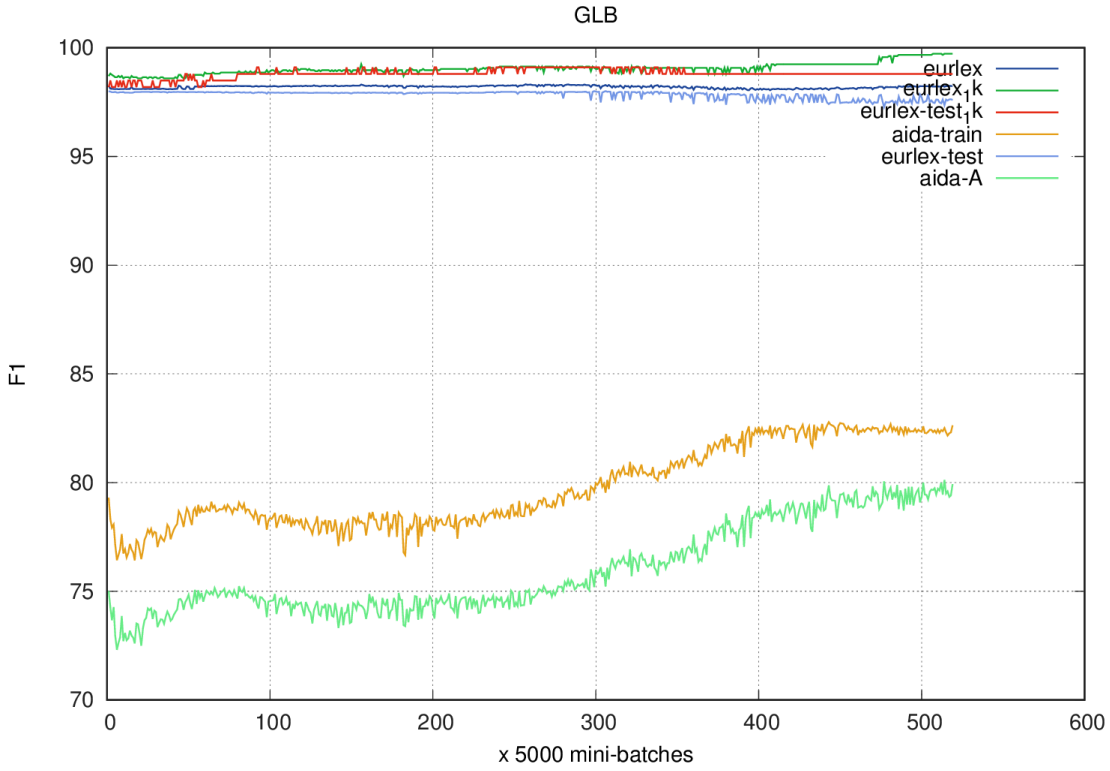


Figure 5.3: The F1 score for the algorithm trained on AIDA-CoNLL corpus during fine tuning with the EUR-Lex 1k dataset.

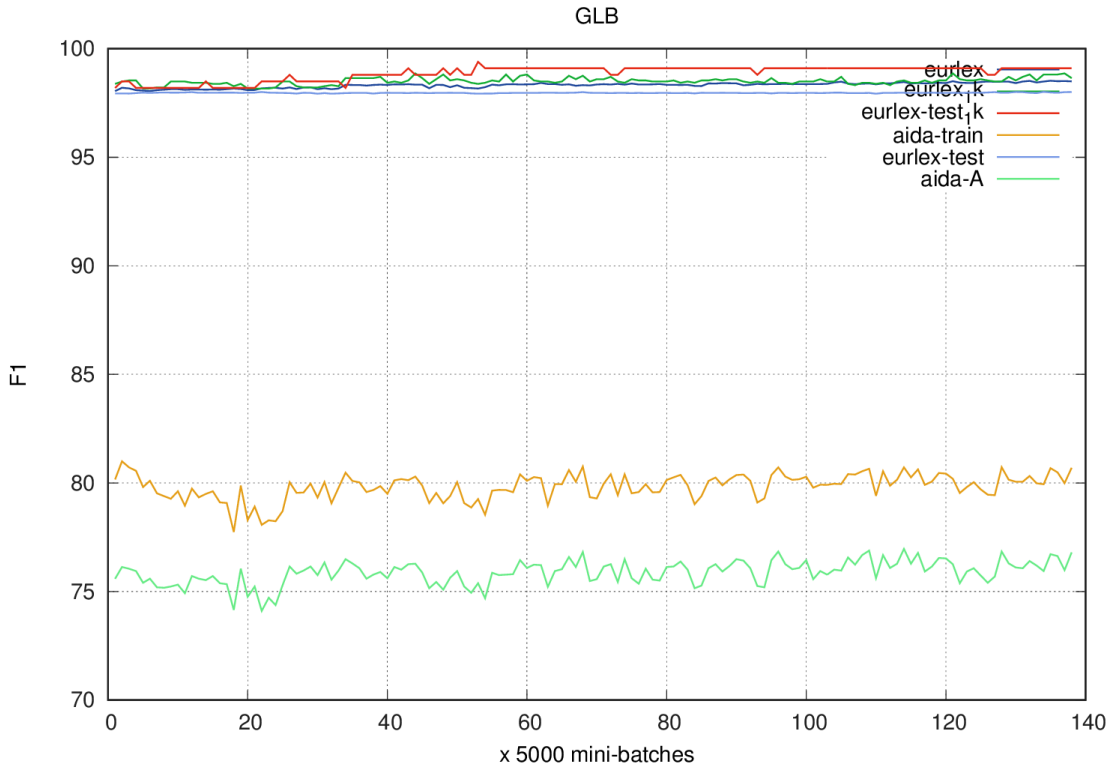


Figure 5.4: The F1 score for the algorithm trained on AIDA-CoNLL corpus during fine tuning with the EUR-Lex 20k dataset.

5.2.3.2 AIDA-CoNLL applied on EUR-Lex

This approach is from a technical point of view exactly the same as the above scenario. The only difference is the change of the source and target datasets. First, we train the neural network on the EUR-Lex dataset. As soon as the F1 score for the EUR-Lex-test set starts dropping, we stop the training and save the model weights for reusing the model. The same statistics from above apply again in this scenario: We have rich data and a high similarity with respect to the frequent mention-entity pairs. As the EUR-Lex dataset is bigger than AIDA-train, this approach is expected to increase the F1 score of AIDA-train compared to training it from scratch. The results are depicted in figure 5.5. It is clearly recognizable that network adjusted its weights away from the source dataset, as the F1 score for EUR-Lex decreased. Furthermore, the F1 score on the new training set, AIDA-train, rose by over 5%. The peak of the F1 score for AIDA-train was 93.41%. This value exceeds the performance peak of the neural network trained from scratch on AIDA-train by more than 1%.

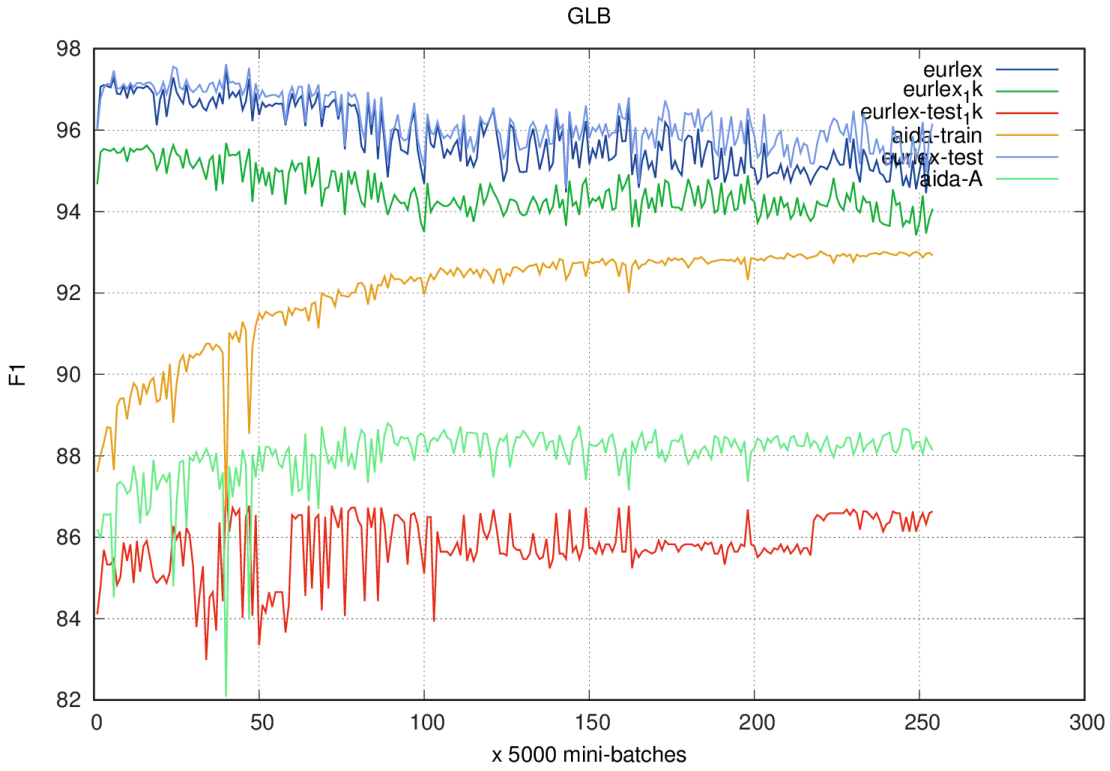


Figure 5.5: The F1 score for the algorithm trained on EUR-Lex 20k corpus during fine tuning with the AIDA-CoNLL dataset.

5.3 Discussion

This section shows the conclusions we made from the results of all experiments. We try to draw a connection between the single experiments and to highlight where significant information could be gathered. Afterwards we set it into relation with the involved limitations in this work.

5.3.1 Single Training

The single training on EUR-Lex and AIDA-CoNLL was conducted to provide a baseline result we could compare transfer learning to. First, we want to analyze the results of the single training on the legal dataset. Figure 5.1 visualizes the results achieved when training on the EUR-Lex corpus only. It is clearly visible, that the network performs extremely well right after the first epoch. The achieved F1 score already exceeds 97.5%. This indicates a fast convergence towards the dataset. The network's weights seem to adapt rapidly in direction of the EUR-Lex corpus. We assume there are multiple reasons for that behavior. First, our EUR-Lex dataset contains almost 34,000 entries of entity-mention pairs. These entries consist of 4,251 different pairs, where 83 of these

pairs occur with a high frequency. These numbers indicate that there is little variety of entities in the legal corpus. Hence, we assume the network can easily learn to predict the correct entity for a mention due to repetition. For example, the mention-entity pair with the highest frequency occurs in over 9,000 entries.
Additionally, the network implemented by [23] has engineered features that take the context into consideration when choosing from a set of possible entities. Thus, especially because the context of the high frequent entities is mostly the same in every occurrence, we assume the network is able to learn which entity to choose extremely fast with the help of the context score. This assumption would also fit the results displayed in figure 5.1. For almost 200 epochs, the network does not improve compared to the first epoch. Then, after about 500 epochs, it clearly overfits on the training set with an F1 score of almost 100%, while the F1 score on the test set decreases. Furthermore, with F1 scores of below 80% on the AIDA-CoNLL train and test sets it performs by far worse than the network trained by [23].

After analyzing the training stage with the EUR-Lex dataset, we compare it to the results of the original network. Figure 5.6 depicts how the two networks performed on the involved datasets. The bar graph on the left shows the F1 scores achieved by training on AIDA-train and validating with Aida-test. The conclusions made by [23] and the fact that the difference between the train and validation F1 score are small indicates that the network is not overfitting. It achieves 92.36% and 90.1% on train and test set, respectively. It is remarkable, that the network still performs extremely well when testing on the EUR-Lex datasets. We assume that this is caused by the number of similar entities in both corpora. Although table 4.3 shows that EUR-Lex and AIDA training sets only share 7% of the entities, the ones that are similar in both sets occur with a high frequency in the EUR-Lex corpus. Concluding from this, the similar entities in combination with the context scores calculated for each entity explain why training on AIDA also results in high F1 scores for EUR-Lex.

5.3.2 Joint Training

Overall, the joint training approach did not satisfy the assumptions. While the initial hope was that the network would learn the generalized as well as the niche dataset in one training approach, the outcome showed a different behavior. Figure 5.2 shows that during the training process of over 130 epochs, the network did not improve in F1 score or accuracy on any of the involved datasets. Even the dedicated joint train and test sets showed no in increase in F1 score. To draw a conclusion on the joint training approach, the results are compared to training on the single datasets, respectively. Figure 5.7 highlights the gap in F1 score for the separate datasets. The left side of the graph shows the F1 scores when training on AIDA-CoNLL and EUR-Lex. The right hand side shows the results when training on the merged dataset of both. Comparing each colored bar on both sides, it is obvious that the F1 score decreased on every single dataset. Therefore we conclude that the assumed behavior did not occur. Instead, it

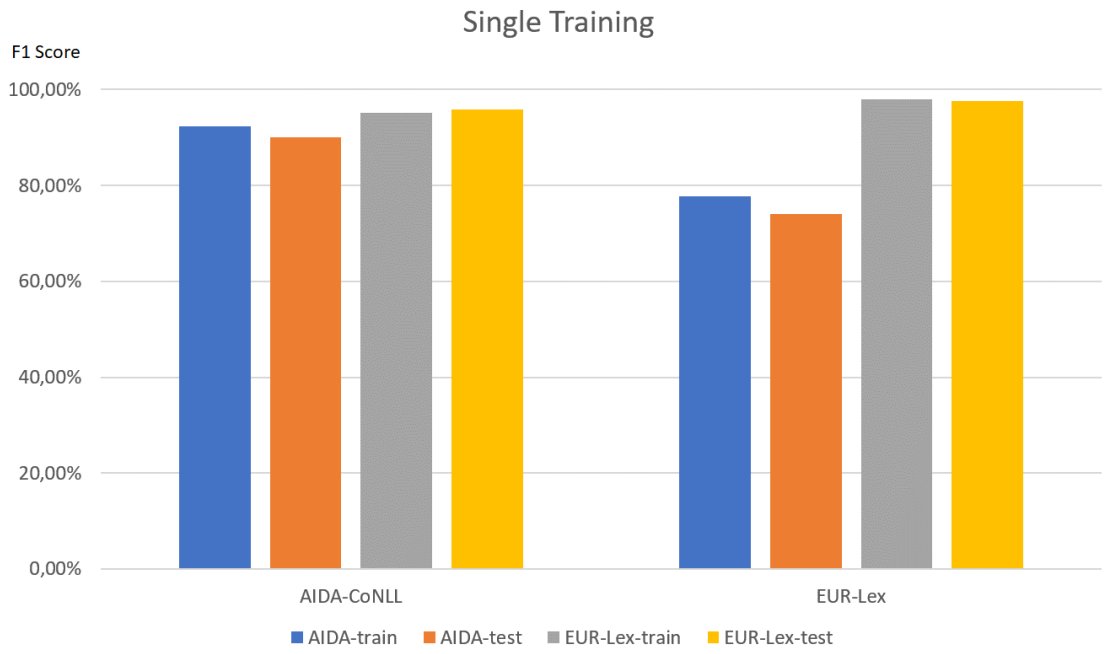


Figure 5.6: Comparison of the results in single training.

seems that the model was not able to propagate its weights towards a value that would fit both datasets. Hence, the values landed somewhere in between both optimal states and thereby produced worse results with respect to both datasets. We conclude that this approach is not recommended in any future approach for named-entity linking with this specific deep entity disambiguation system developed in [23]. This does not exclude that joint training could work with other datasets and/or another neural network for named-entity linking.

5.3.3 Transfer Learning

Three different approaches are taken into account for judging on the success of transfer learning in a deep named-entity linking system.

- AIDA-CoNLL $\rightarrow$ EUR-Lex 1k.
  Figure 5.3 shows that in the first 400 epochs of fine tuning the original model on the smaller dataset does not improve in F1 score. In particular, the network immediately adapts its weights towards the EUR-Lex corpus in the first epoch. The performance starts with an F1 score of over 99% and improves up to 99.73%. Compared to single training on the EUR-Lex dataset, the achieved accuracies and F1 score were close to 99%. In the case of fine tuning with the 1k dataset, figure 5.8 has not as much meaningfulness compared to the other transfer learning approaches, as the difference in the F1 scores of fine tuning and single training is

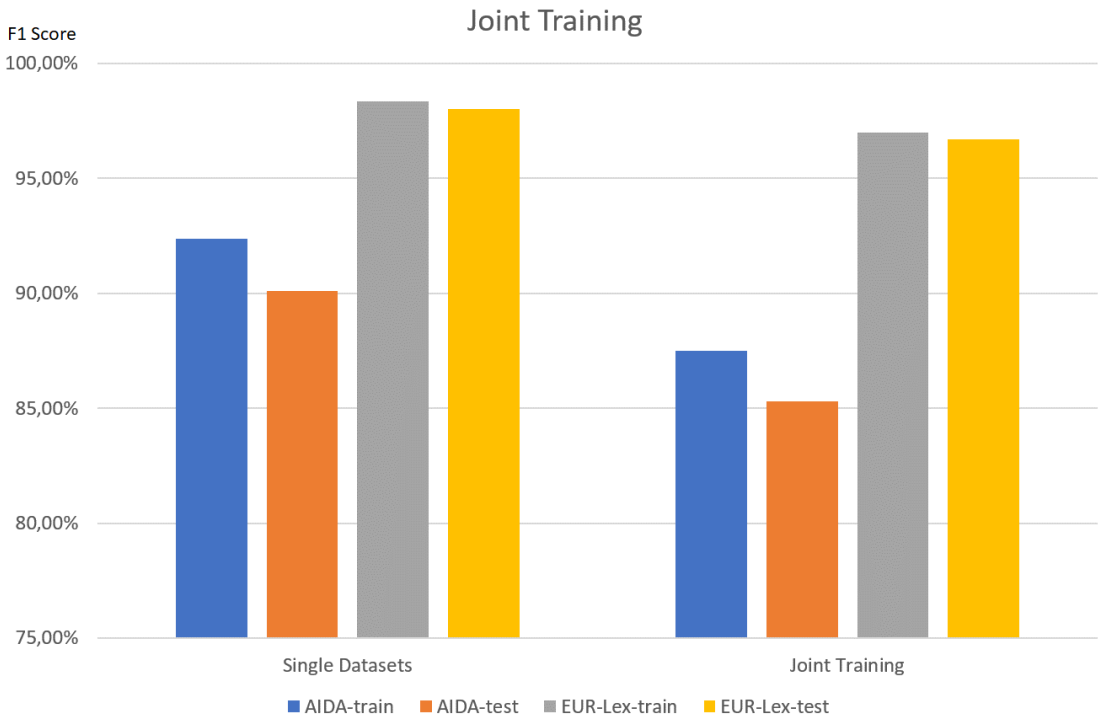


Figure 5.7: Comparison of the results in joint training.

located within 0.7%. The F1 scores for the AIDA-CoNLL datasets also improved comparing the single training on EUR-Lex to the transfer of learning case AIDA-CoNLL → EUR-Lex 1k. This is the expected behavior as the model has not seen the AIDA-CoNLL dataset in the EUR-Lex single training scenario whereas the pre-trained model in the transfer learning case was trained on AIDA-CoNLL. From the increase in F1 score and accuracy when fine tuning the model trained on AIDA-CoNLL, we conclude that the transfer learning approach succeeded for the case of having not enough data for single training.

- AIDA-CoNLL → EUR-Lex 20k.
  This approach indicates the same conclusions as the experiment above. Consulting figure 5.9 and table 5.2, it can be observed that the F1 score increased for the EUR-Lex 20k dataset when employing transfer learning. Additionally, similar to the 1k approach, also the F1 scores of the AIDA-train and test set improve compared to the single training of EUR-Lex. Hence, the model does not only improve in performance with respect to the EUR-Lex set, it simultaneously keeps certain knowledge about resolving named-entity on the AIDA-CoNLL corpus.

- EUR-Lex → AIDA-CoNLL.
  This approach produced promising and interesting results. Figure 5.5 nicely depicts the learning curve of the model with respect to the AIDA-CoNLL dataset.

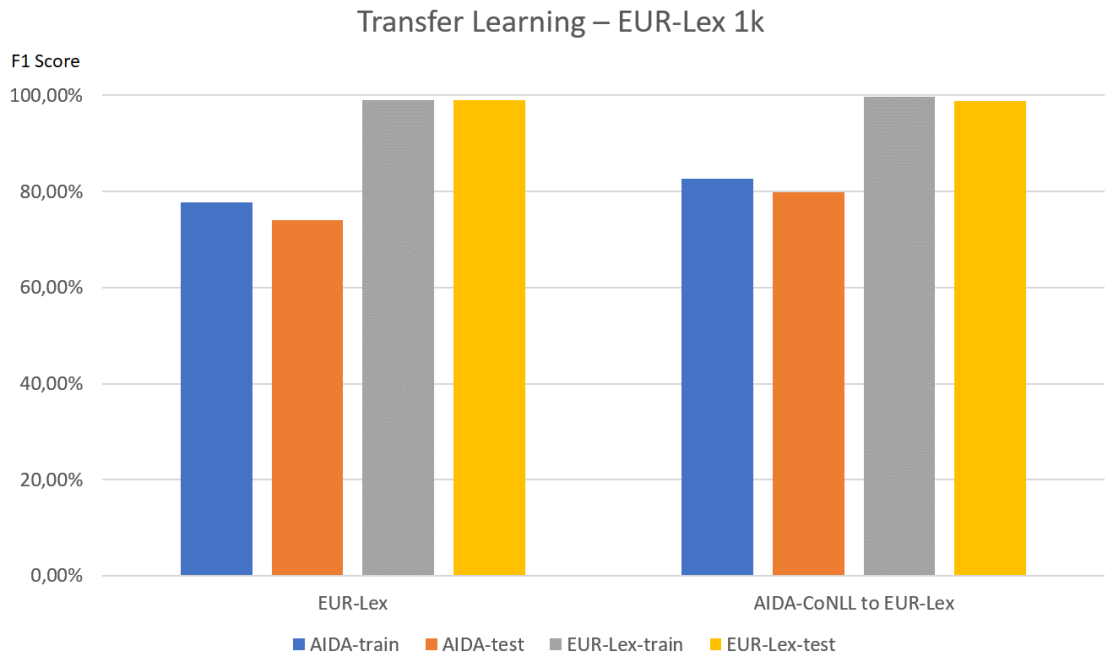


Figure 5.8: Comparison of the results for AIDA-CoNLL → EUR-Lex 1k.

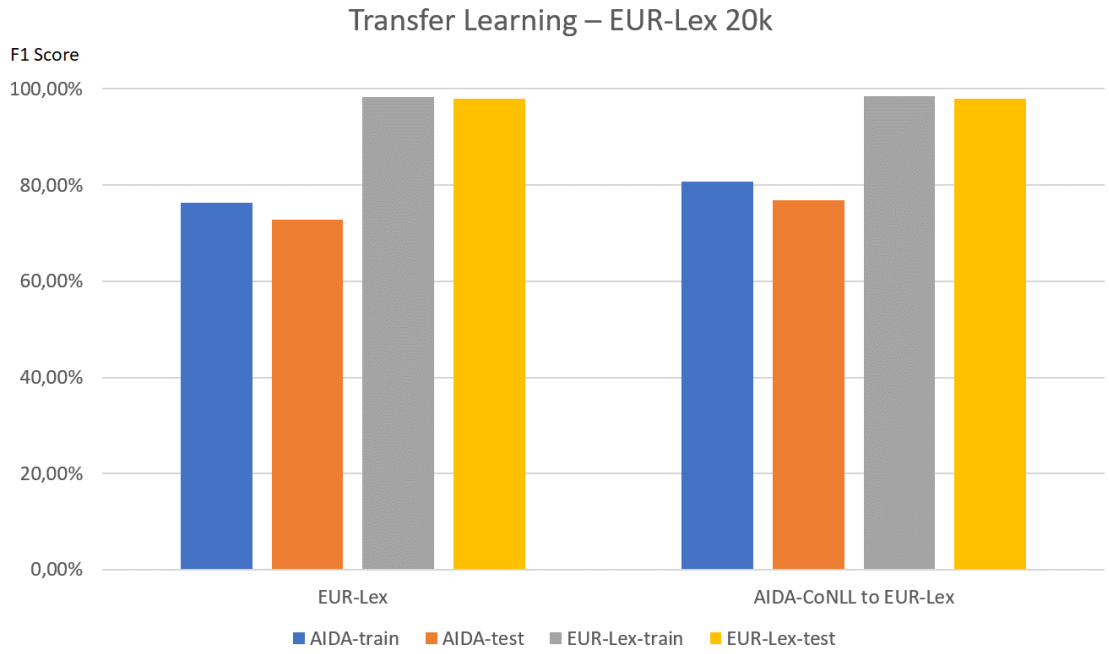


Figure 5.9: Comparison of the results for AIDA-CoNLL → EUR-Lex 20k.

Although it started with a bias towards the EUR-Lex 20k dataset, which is bigger compared to the AIDA corpus, the network was able to learn the better generalized dataset with decent performance. With a F1 score of 93.41% after transfer learning, compared to 92.36% in single training, it outperforms the original approach of Ganea et al. [23] with respect to the AIDA-train dataset. Despite the fact that the F1 score increased when fine-tuning the model, it has to be noted that the networks performance decreases on the AIDA-test dataset by 1.3% from 90.1% to 88.8%. This indicates that the network could be overfitting on the AIDA-train dataset in this transfer learning approach. Interpreting the above observations, it is hard to draw an objective conclusion. The benefits of this technique can not be ranked with respect to its downsides. Therefore, a clear advantage of using transfer learning after training on the automatically created dataset could not be observed.

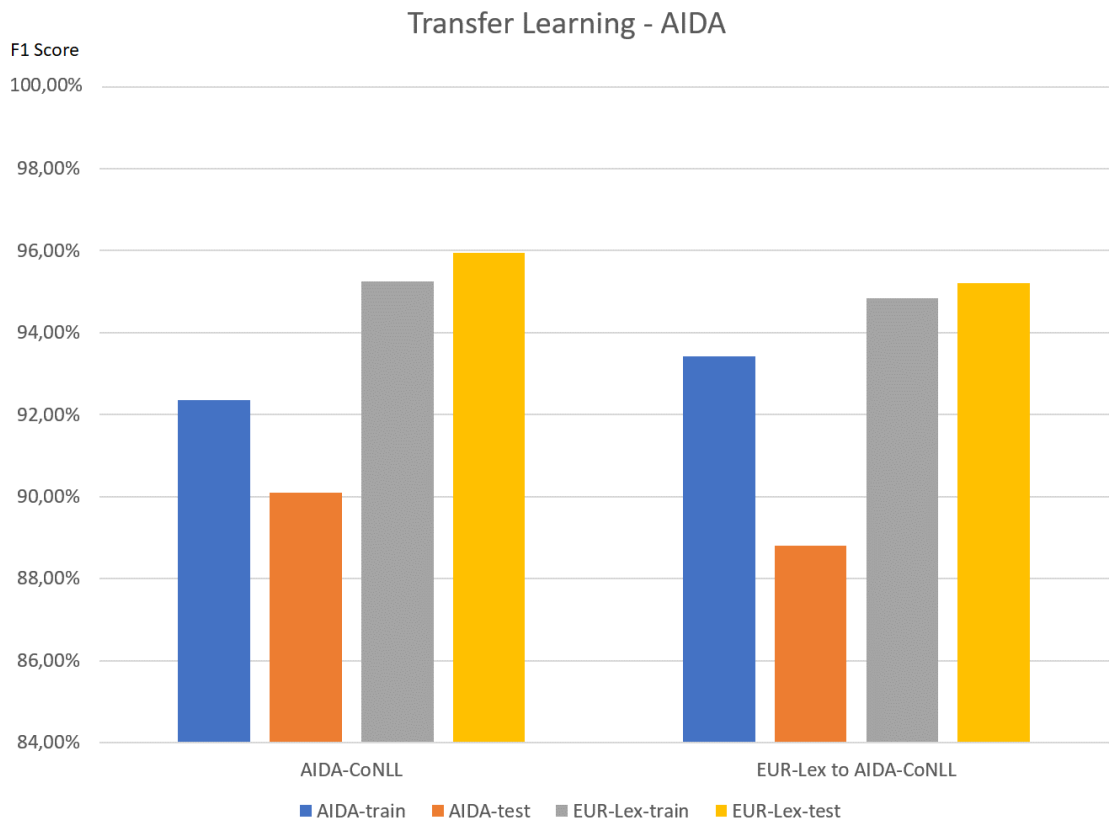


Figure 5.10: Comparison of the results for EUR-Lex 20k → AIDA-CoNLL.

5.3.4 Summary

Gathering all the above results, the following suggestions can be made. The outcome suggests that applying transfer learning when the datasets are similar and rich in size, results in a higher F1 score for the smaller dataset if the bigger one is used for training the network from scratch. This indicates that the resulting model has a higher skill level than the model trained from scratch. Especially in the case of training on AIDA-CoNLL and fine tuning with the EUR-Lex 1k dataset,we observed an increase on the train and test set. As explained in subsection 2.1.2.3, producing a model with higher skill is one of the expected benefits when using transfer learning in the current scenario. Additionally, the model did not only have a higher skill but also reached its performance peak faster compared to the single training. After the first epoch, both fine tuning approaches with the EUR-Lex dataset outperformed the single training. This better start is the second of three benefits that can be acquired by using transfer learning. Finally, by comparing figure 5.5 and figure 4.2, a steeper slope in the transfer learning approach is observed. Employing the deep learning foundations of transfer learning from subsection 2.1.2, the figures indicate that all three possible benefits can be observed in the experiments. Although the difference in the results was not immense, a trend could be observed, clearly indicating that transfer learning is beneficial in named-entity linking. Figure 5.1 summarizes the observed behaviors in the different experiments. The row with the single training results serves as a baseline result. The green values indicate an increase in performance with respect to the F1 score the network achieved when testing it on the respective dataset. Generally, the model increased on most of the datasets. Although, we can only truly identify a benefit of transfer learning, whenever testing on the train ***and*** the test set resulted in a higher F1 score. This only happened with respect to the EUR-Lex 1k dataset. For joint training, as already explained above, we couldn't observe any improvement and therefore conclude that in our setup the technique was not beneficial.

	Single Training	Joint Training	Transfer Learning
AIDA-train	92.36%	87.50%	93.41%
AIDA-A	90.1%	85.29%	88.8%
EUR-Lex-train 1k	99.02%	97.14%	99.73%
EUR-Lex-train 20k	98.34%	97.36%	98.49%
EUR-Lex-test 1k	98.29%	90.41%	98.90%
EUR-Lex-test 20k	98.01%	97.19%	98.01%

Table 5.1: F1 score comparison between all experiments. The single training value represents the baseline result joint training and transfer learning are compared to.

5.3.5 Limitations

This subsection highlights the limitations this work was restricted to. Generally, the constraints can be divided into the following topics.

- Dataset creation.
  In the current state, research in natural language processing in the legal domain is often prevented due to a scarcity of data. Therefore, the EUR-Lex dataset was created in the scope of this work. It was automatically generated with the help of different Python frameworks as explained in 4.1.1.2. Due to the fact that it was not hand-crafted, it contains a lot of duplications. As manually sorting out entities was not an option in the scope of this thesis, there was no option but to analyze the dataset and employ the statistics in the interpretation. By removing repetition from the dataset, a model trained on that dataset would probably outperform the results achieved in this thesis.

- Prior work.
  Publicly available approaches for deep named-entity linking are scarce. Only a few are accessible, where even fewer provide a pre-trained model or a documentation on how to rebuild results presented in the respective publication. Furthermore, no prior work on transfer learning for named-entity linking with deep learning could be found. This fact both represents a motivation as well as a limitation for this work. Subsequently, the approach by Ganea et al. [23] was not created for reusing it for transfer learning. It required a lot of code archeology to be able to recreate the results and to include new datasets.

- Time.
  Time was the dominant reason for not considering to manually create or adapt the generated EUR-Lex dataset. The achieved results in this thesis imply that the automatically created dataset was good enough to trigger the possible benefits of transfer learning. We could observe a better start, a steeper slope and a higher skill in some of the transfer learning approaches. Also, time prevented to create a second kind of automatically extracted dataset from the EUR-Lex corpus that could provide better generalization and thereby might outperform the achieved results. Additionally, a dataset for the legal domain on named-entity linking would have been available, this work could have focused on comparing different named-entity linking approaches by employing deep learning. Unfortunately, the remaining time did not leave room for evaluating a second approach.

	trained with	tested on	F1	Accuracy
Single Training	AIDA-CoNLL	Aida-train	92.36%	91.91%
		Aida-testA	90.1%	91.4%
		Eurlex-train 1k	95.14%	95.2%
		Eurlex-train 20k	95.25%	95.26%
		Eurlex-test 1k	85.46%	89.79%
		Eurlex-test 20k	95.94%	97.19%
	EUR-Lex 1k	Aida-train	77.8%	78.1%
		Aida-testA	74.0%	75.43%
		Eurlex-train 1k	99.02%	99.01%
		Eurlex-train 20k	98.0%	97.9%
		Eurlex-test 1k	98.29%	98.2%
		Eurlex-test 20k	97.7%	97.52%
	EUR-Lex 20k	Aida-train	76.36%	77.81%
		Aida-testA	72.85%	76.15%
		Eurlex-train 1k	98.43%	98.54%
		Eurlex-train 20k	98.34%	98.44%
		Eurlex-test 1k	98.20%	98.79%
		Eurlex-test 20k	98.01%	99.29%
Transfer Learning	AIDA-CoNLL –>EUR-Lex 1k	Aida-train	82.63%	84.20%
		Aida-testA	79.93%	83.54%
		Eurlex-train 1k	99.73%	99.78%
		Eurlex-train 20k	98.23%	98.25%
		Eurlex-test 1k	98.90%	98.80%
		Eurlex-test 20k	97.60%	98.86%
	AIDA-CoNLL –>EUR-Lex 20k	Aida-train	80.70%	77.7%
		Aida-testA	76.80%	75.56%
		Eurlex-train 1k	98.65%	98.49%
		Eurlex-train 20k	98.49%	98.29%
		Eurlex-test 1k	99.09%	98.8%
		Eurlex-test 20k	98.01%	99.28%
	EUR-Lex –>AIDA-CoNLL	Aida-train	93.41%	93.2%
		Aida-testA	88.80%	88.83%
		Eurlex-train 1k	93.92%	93.94%
		Eurlex-train 20k	94.84%	94.8%
		Eurlex-test 1k	86.43%	86.5%
		Eurlex-test 20k	95.21%	95.24%
Joint Training	Merged Dataset	Aida-train	87.50%	89.17%
		Aida-testA	85.29%	89.16%
		Eurlex-train 1k	97.14%	97.2%
		Eurlex-train 20k	97.36%	97.37%
		Eurlex-test 1k	90.41%	90.39%
		Eurlex-test 20k	97.19%	98.45%

Table 5.2: F1 scores and accuracies on all ED datasets

6 Conclusions and Outlook

This chapter presents the conclusions drawn from the outcome of this work. First, the results are put into the context of the research of named-entity linking. Afterwards, potential future tasks ensuing this work are presented.

6.1 Conclusion

The combination of all of the conducted experiments, the interpretation of the respective results and the limitations involved in the execution serve as a basis for drawing the final conclusions. The scope of this thesis to apply transfer learning to named-entity linking system that employs deep learning for resolving entity disambiguation. This section provides a summarization of the answers to the research questions that motivated the implementation of this thesis.

1. *What kind of existing approach should be used for transfer learning?*
   Since named-entity linking itself is a complicated task, not many approaches use deep learning to solve it. The majority in the state of the art uses highly engineered features for entity linking. To our best knowledge we couldn't find any related work coping with the exact same problem. Resulting from that, we can not entirely compare our approach or our results to any prior work. The definition of success of this work thereby is concluded from the achieved accuracies and F1 scores only. By employing the F1 score criteria as a performance metric, the following conclusions can be drawn.
   First of all, choosing the named-entity linking system developed by Ganea et al. [23] for the use of transfer learning is considered the right choice. It fulfilled the fixed requirement of employing deep learning techniques for the entity linking task and outperformed most, if not all state of the art systems. Although the complexity of the entire model was very high, it was possible to include a new dataset and to easily save and reuse prior results. One of the authors, Octavian Ganea, provided support via E-mail whenever needed. Also, their GitHub repository included instructions on how to rebuild their results. Despite the fact that this instruction was partly incomplete, it provided big help in the reproduction of the original model. In a nutshell, the model from [23] did not only fulfill the objective performance criteria, but also provided some help to implement the necessary steps for the answer of this thesis.

2. *Which technique of transfer learning suits best?*
   The different scenarios where transfer learning can and their respective optimal

technique were presented in subsection 2.1.2.2. As stated in subsection 5.2.3, the two involved datasets, AIDA-CoNLL and EUR-Lex, are both rich in data and have high similarity with respect to the high frequent entities. Prior work [11] suggests to employ the technique of fine tuning a network with the new dataset without exchanging any layer. This is justified by indicating that if the datasets are similar, the network only needs to adapt its weights slightly towards the new dataset. For this reason, we decided that the fine tuning technique is the most promising in this work.

3. *Is the use of transfer learning with named-entity linking beneficial in the legal domain?* Table 5.2 presents the final results achieved in all experiments conducted in the scope of this work. Subsection 5.2.3 gives detailed explanations for the improvements that could be observed when applying transfer learning on a dataset from the legal domain. By having a closer look on the final results, transfer learning for the EUR-Lex resulted in an improved F1 score and accuracy compared to the single training experiment. Unexpectedly, we even observed an improvement on the AIDA-CoNLL training dataset when employing transfer learning. Although in this case the test F1 score dropped compared to the single training, we are confident that with a polished legal dataset there is even more room for improvement. Finally, we can conclude that transfer learning was beneficial in the legal domain. Furthermore, the observations even imply that the performance of named-entity linking systems can improve generally through the use transfer learning.

6.2 Outlook

The outcome of this work gives inspiration for future work in natural language processing in the legal domain. Three ensuing tasks with respect to this work could be of interest for the legal domain. First of all, the dataset created in the scope of this thesis should be made publicly available. Although it is automatically created, the quality and quantity of the dataset suffice to result in added value for research in natural language processing. Before the dataset can be made publicly available, it needs to be polished. Despite the fact that an improvement in accuracy could be achieved by employing this dataset for transfer learning in the named-entity linking task, it still holds more potential. In its current state, almost one-third of the entries refer to the same mention-entity pair. With more generalization introduced by manual adaption, the performance of the network especially when applying transfer learning could outperform the results of this work.
Furthermore, it is possible to extract a second kind of dataset from the EUR-Lex corpus [18]. Most of the documents in the corpus contain a *definitions* section, where certain words are defined in the scope of the document. These words that have a definition could serve as entities in the dataset. Per document, every occurrence of that entity would represent a mention. The ambiguity would be simulated by different definitions in different documents for the same word. In that case, a named-entity linking system

would have to choose the correct definition for a mention depending on the context of that mention. This dataset would also be extracted automatically. Though, since it is based on domain expert knowledge, the quality of the dataset can be compared to a hand-crafted dataset. This dataset has already been created in the scope of this thesis, but due to time issues we weren't able to include it in any kind of experiments. Both datasets will be published subsequent to this thesis.
Since the developed pipeline in this work accepts every form of raw text, one more concrete use case could be developed in future work. Iteratec GmbH, who provided great support throughout the development of the thesis, have access to a huge database with technological skills of every employee working at the company. It would be an interesting implementation to use such information in a named-entity linking scenario by connecting the different skills to their respective named-entities. A success of such an approach would build a basis for supplementary tools to automatically analyze and extract valuable information from CVs or LinkedIn[1] profiles. The biggest stakeholders in that use case would be recruiters in general or human resource departments of different firms.
Finally yet importantly, an actual application for a named-entity linking system can be implemented following the results of this work. Several steps are needed if the deep learning architecture used in this work should be transformed into a real-world ready application. First, before a named-entity linking system can be of any use, a named-entity recognizer has to be included in the pipeline. That named-entity recognizer would need to accept raw text only as an input and output a potential mention with information about its context. This mention could then be passed to the linker, putting the mention and its context into the needed input format. The named-entity linking system then would need to output the entity that the model predicted for the respective mention. With this output, different benefits for stakeholders could be implemented. One example would to include meta information in texts from the legal domain, i.e. a tool-tip for a domain specific word, indicating its definition in the current context.
Two of these potential future tasks, namely the publication of the datasets, are planned to be implemented shortly after the end of thesis. The implementation of an actual named-entity linking application is an entire new project and could probably serve enough content to be the basis for another thesis.

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

List of Figures

List of Tables

Bibliography

[1] D. E. Appelt and D. J. Israel. *Introduction to Information Extraction Technology*.

[2] Auriga. *Digital Transformation: History, Present, and Future Trends | Software Development News, Events, Thoughts from Auriga*. 2016.

[3] *Beautiful soup documentation*. https://www.crummy.com/software/BeautifulSoup/.

[4] J. Brownlee. *A Gentle Introduction to Transfer Learning for Deep Learning - Machine Learning Mastery*. 2017.

[5] C. Cardellino, M. Teruel, L. A. Alemany, and S. Villata. "A low-cost, high-coverage legal named entity recognizer, classifier and linker." In: *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*. ACM. 2017.

[6] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. "Dexter: an open source framework for entity linking." In: *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*. ACM. 2013, pp. 17–20.

[7] A. Chisholm and B. Hachey. "Entity disambiguation with web links." In: *Transactions of the Association of Computational Linguistics* 3.1 (2015), pp. 145–156.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural language processing (almost) from scratch." In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537.

[9] S. Cucerzan. "Large-scale named entity disambiguation based on Wikipedia data." In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007.

[10] *Dandelion API - Semantic Text Analytics as a service*. https://dandelion.eu/.

[11] Data Flair. *Transfer Learning for Deep Learning with CNN*. https://data-flair.training/blogs/transfer-learning/. 2018.

[12] L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva. "Analysis of named entity recognition and linking for tweets." In: *Information Processing & Management* 51.2 (2015).

[13] C. B. Do and A. Y. Ng. "Transfer learning for text classification." In: *Advances in Neural Information Processing Systems*. 2006, pp. 299–306.

[14] G. Durrett and D. Klein. "A joint model for entity analysis: Coreference, typing, and linking." In: *Transactions of the Association for Computational Linguistics* 2 (2014), pp. 477–490.

[15] Y. Eshel, N. Cohen, K. Radinsky, S. Markovitch, I. Yamada, and O. Levy. "Named Entity Disambiguation for Noisy Text." In: *arXiv preprint arXiv:1706.09147* (2017).

[16] D. Etherington. *Udacity open sources its self-driving car simulator for anyone to use | TechCrunch*. 2017.

[17] EURNEWS. *Named Entity Recognition for digitised newspapers*. 2014.

[18] European Union. *EUR-Lex*. http://eur-lex.europa.eu/content/welcome/about.html. 2018.

[19] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. "Pathnet: Evolution channels gradient descent in super neural networks." In: *arXiv preprint arXiv:1701.08734* (2017).

[20] J. R. Finkel, T. Grenager, and C. Manning. "Incorporating non-local information into information extraction systems by gibbs sampling." In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2005, pp. 363–370.

[21] M. Francis-Landau, G. Durrett, and D. Klein. "Capturing semantic similarity for entity linking with convolutional neural networks." In: *arXiv preprint arXiv:1604.00734* (2016).

[22] O.-E. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann. "Probabilistic bag-of-hyperlinks model for entity linking." In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 927–938.

[23] O.-E. Ganea and T. Hofmann. "Deep Joint Entity Disambiguation with Local Neural Attention." In: *arXiv preprint arXiv:1704.04920* (2017).

[24] *Getting started with Torch*. http://torch.ch/docs/getting-started.html.

[25] K. Gidney. *What deep learning is doing for the legal sector | VentureBeat*. https://venturebeat.com/2017/03/28/what-deep-learning-is-doing-for-the-legal-sector/. 2017.

[26] N. S. Gill. *Overview of Artificial Intelligence and Natural Language Processing*. https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/. 2017.

[27] I. Glaser. "Semantic Analysis and Structuring of German Legal Documents using Named Entity Recognition and Disambiguation." PhD thesis. TUM, 2017.

[28] I. Glaser, B. Waltl, and F. Matthes. "Named Entity Recognition, Extraction, and Linking in German Legal Contracts." In: (2017).

[29] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[30] Z. Guo and D. Barbosa. "Robust named entity disambiguation with random walks." In: *Semantic Web* Preprint (2016), pp. 1–21.

[31] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran. "Evaluating entity linking with Wikipedia." In: *Artificial intelligence* 194 (2013), pp. 130–150.

[32] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang. "Learning entity representation for entity disambiguation." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2. 2013, pp. 30–34.

[33] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. "Robust disambiguation of named entities in text." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011.

[34] J. Hu, J. Lu, and Y.-P. Tan. "Deep transfer metric learning." In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE. 2015, pp. 325–333.

[35] IBM. *IBM Watson and scaling expertise*. https://www.youtube.com/watch?v=lYxJlFL2opo&feature=share. 2015.

[36] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al. "Google's multilingual neural machine translation system: enabling zero-shot translation." In: *arXiv preprint arXiv:1611.04558* (2016).

[37] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A convolutional neural network for modelling sentences." In: *arXiv preprint arXiv:1404.2188* (2014).

[38] M. Kiser. *Introduction to Natural Language Processing (NLP) - Algorithmia Blog*. 2016.

[39] Kritschmar, Charlie. *Datamodel in Wikidata*. https://en.wikipedia.org/wiki/Wikidata. 2016.

[40] A. Krizhevsky and G. Hinton. "Learning multiple layers of features from tiny images." In: (2009).

[41] J. Lafferty, A. McCallum, and F. C. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." In: (2001).

[42] N. Lazic, A. Subramanya, M. Ringgaard, and F. Pereira. "Plato: A selective context model for entity resolution." In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 503–515.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[44] J. Y. Lee, F. Dernoncourt, and P. Szolovits. "Transfer Learning for Named-Entity Recognition with Neural Networks." In: *arXiv preprint arXiv:1705.06273* (2017).

[45] N. Lintz. *Exploring Computer Vision (Part II): Transfer Learning*. https://indico.io/blog/exploring-computer-vision-transfer-learning/. 2016.

[46] D. Liu, W. Lin, S. Zhang, S. Wei, and H. Jiang. "The USTC NELSLIP Systems for Trilingual Entity Detection and Linking Tasks at TAC KBP 2016." In: ().

[47] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60.

[48] E. Meij, K. Balog, and D. Odijk. *Entity linking and retrieval tutorial*.

[49] D. Milne and I. H. Witten. "Learning to link with wikipedia." In: *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM. 2008, pp. 509–518.

[50] A.-C. N. Ngomo, M. Röder, D. Moussallem, R. Usbeck, and R. Speck. "Automatic Generation of Benchmarks for Entity Recognition and Linking." In: *arXiv preprint arXiv:1710.08691* (2017).

[51] R. Nordquist. *Definition and Examples of Syntactic Ambiguity*. 2017.

[52] E. S. Olivas. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.

[53] S. J. Pan and Q. Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[54] J. Pennington, R. Socher, and C. Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[55] L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin. "Named entity recognition for novel types by transfer learning." In: *arXiv preprint arXiv:1610.09914* (2016).

[56] D. Rao, P. McNamee, and M. Dredze. "Entity linking: Finding extracted entities in a knowledge base." In: *Multi-source, multilingual information extraction and summarization*. Springer, 2013.

[57] L. Ratinov, D. Roth, D. Downey, and M. Anderson. "Local and global algorithms for disambiguation to wikipedia." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 1375–1384.

[58] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, and A. Both. "N^3-A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format." In: *LREC*. 2014, pp. 3529–3533.

[59] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. "To transfer or not to transfer." In: *NIPS 2005 workshop on transfer learning*. Vol. 898. 2005, pp. 1–4.

[60] Ruder, Sebastian. *Transfer Learning - Machine Learning's Next Frontier*. http://ruder.io/transfer-learning/. 2017.

[61] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. "Sim-to-real robot learning from pixels with progressive nets." In: *arXiv preprint arXiv:1610.04286* (2016).

[62] Y. Sawada, Y. Sato, T. Nakada, K. Ujimoto, and N. Hayashi. "All-transfer learning for deep neural networks and its application to sepsis classification." In: *arXiv preprint arXiv:1711.04450* (2017).

[63] *Scrapy at a glance*. https://docs.scrapy.org/en/latest/intro/overview.html.

[64] A. Sil, E. Cronin, P. Nie, Y. Yang, A.-M. Popescu, and A. Yates. "Linking named entities to any database." In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics. 2012, pp. 116–127.

[65] A. Sil and R. Florian. "One for all: Towards language independent named entity linking." In: *arXiv preprint arXiv:1712.01797* (2017).

[66] M. Sokolova, N. Japkowicz, and S. Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." In: *Australasian joint conference on artificial intelligence*. Springer. 2006, pp. 1015–1021.

[67] M. Sokolova and G. Lapalme. "A systematic analysis of performance measures for classification tasks." In: *Information Processing & Management* 45.4 (2009).

[68] R. Stern, B. Sagot, and F. Béchet. "A joint named entity recognition and entity linking system." In: *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*. Association for Computational Linguistics. 2012.

[69] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang. "Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation." In: *IJCAI*. 2015, pp. 1333–1339.

[70] C. Taylor. *Structured vs. Unstructured Data - Datamation*. 2017.

[71] R. Tromans. *Legal AI A beginner's guide About this White Paper*. https://blogs.thomsonreuters.com/legal-uk/wp-content/uploads/sites/14/2017/02/Legal-AI-a-beginners-guide-web.pdf. 2017.

[72] J. Twomey and A. Smith. "Performance measures, consistency, and power for artificial neural network models." In: *Mathematical and computer modelling* 21.1-2 (1995), pp. 243–258.

[73] C. Urmson. *Official Google Blog: The latest chapter for the self-driving car: mastering city street driving*. 2014.

[74] R. Usbeck, A.-C. N. Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer, and A. Both. "AGDISTIS-graph-based disambiguation of named entities using linked data." In: *International Semantic Web Conference*. Springer. 2014, pp. 457–471.

[75] S. Van Hooland, M. De Wilde, R. Verborgh, T. Steiner, and R. Van de Walle. "Exploring entity recognition and disambiguation for cultural heritage collections." In: *Digital Scholarship in the Humanities* 30.2 (2013), pp. 262–279.

[76] S. Wiseman, A. M. Rush, and S. M. Shieber. "Learning global features for coreference resolution." In: *arXiv preprint arXiv:1604.03035* (2016).

[77] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. "Joint learning of the embedding of words and entities for named entity disambiguation." In: *arXiv preprint arXiv:1601.01343* (2016).

[78] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. "Supervised Representation Learning: Transfer Learning with Deep Autoencoders." In: *IJCAI*. 2015, pp. 4119–4125.