UNIVERSITY
OF TWENTE.

deepset

Effects of Inserting Domain Vocabulary and Fine-tuning BERT for German Legal Language

Master's Thesis

Faculty of Electrical Engineering, Mathematics and Computer Science

Masters in Interaction Technology

Specialization in Intelligent Systems

University of Twente

submitted by

Chin Man Yeung Tai

Supervisors:	Mariët Theune
	Christin Seifert
External Supervisor (deepset):	Timo Möller

November 26, 2019

Abstract

We explore in this study the effects of domain adaptation in NLP using the state-of-the-art pre-trained language model BERT. Using its German pre-trained version and a dataset from OpenLegalData containing over 100,000 German court decisions, we fine-tuned the language model and inserted legal domain vocabulary to create a German Legal BERT model. We evaluate the performance of this model on downstream tasks including classification, regression and similarity. For each task, we compare simple yet robust machine learning methods such as TFIDF and FastText against different BERT models, mainly the Multilingual BERT, the German BERT and our fine-tuned German Legal BERT. For the classification task, the reported results reveal that all models were equally performant. For the regression task, our German Legal BERT model was able to slightly improve over FastText and the other BERT models but it is still considerably outperformed by TFIDF. In a within-subject study (N=16), we asked subjects to evaluate the relevancy of documents retrieved by similarity compared to a reference case law. Our findings indicate that the German Legal BERT, to a small degree, was able to capture better legal information for comparison. We observed that further fine-tuning a BERT model in the legal domain when the pre-trained language model already included legal data yields marginal gains in performance.

Acknowledgement

Researching for this thesis has been a long and intense journey, It felt like diving into a new field for me. To be able to work with Deep Learning applied to NLP was equal parts exciting and daunting, but it was definitely eased thanks to the precious guidance and support from my supervisors Mariët Theune and Christin Seifert.

I would like to thank you both for all the ideas and suggestions you provided me. Thank you Mariët, not only for the myriad of feedbacks on how to conduct and document research but also for ensuring that the progress was on track. Thank you, Christin for sharing your expertise in data science and pointing out the countless difficulties that are not so easy to recognize. After every talk we had, I found myself renewed with energy, new ideas and motivation to continue with this research. For this, I am very grateful.

Special thanks to my external supervisor, Timo Möller, for being always so helpful, teaching me new concepts and sharing your knowledge of the AI industry with me. I really appreciated that you always found some time to follow up with my research and guide me through the next steps. Thank you deepset for making me feel part of the team and letting me contribute to your amazing open-source project. As well, thanks for enabling this research project by allowing me access to your cloud resources.

Finally, I would like to extend my gratitude to my family and friends, especially my peers from the EIT studies, who were always there to encourage me when I needed it the most.

Abbreviations and Acronyms

AP Average Precision

AF Activation Function

BERT Bi-directional Encoder Representations using Transformers

BoW Bag of Words

CNN Convolutional Neural Network

FARM Framework for Applicable Representation Models

IR Information Retrieval

LM Language Model

ML Machine Learning

NER Named Entity Recognition

NLP Natural Language Processing

OOV Out of vocabulary

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

TFIDF Term Frequency–Inverse Document Frequency

VSM Vector Space Model

List of Figures

List of Tables

Contents

1 Introduction

Language is the scaffold of our minds. We build our thoughts through language and it conditions how we experience and interact with the world. However, the social nature of the human being makes us dependent on each other for our most crucial needs. In order to achieve fluent interaction, natural language is the principal communication tool to express our intents and expectations. From its primitive form including vocal and body cues to digital text representations, language has enabled but also evolved together with the technological progress.

Natural Language Processing (NLP) is the discipline within the field of Artificial Intelligence (AI) that intends to equip machines with the same comprehension capability of *natural language* as humans do. This field has the goal of extracting knowledge from a text corpus and processing it for a wide array of tasks that provide valuable insights on the analyzed data. Commonly, computers are well suited to process *formal language*. This entails structured data, organized rules and commands without ambiguity. Examples of such are programming languages or mathematical expressions.

Natural language comes with its own set of challenges. Not only the content is unstructured, but the language itself is ambiguous and inconsistent. Metaphors, polysemy, rhetoric such as sarcasm or irony and a vast collection of ambiguities are even hard to grasp for humans when reading. These nuances and sources of difficulties to proper understanding are exacerbated by the variety of national languages (English, German, Dutch, etc.). At the same time, the technical domains where it is being used (scientific, administrative, legal language to name a few) play an essential role defining the meaning of the words. Finally, the context and the implied information from world knowledge are important to the correct interpretation. So, how does NLP deal with these barriers?

Traditionally, methods employed by NLP practitioners have been based on complex sets of hand-written rules. The design and implementation of rules that try to model the complexity of a language needed to take into account all the linguistic elements and nuances. Needless to say, these systems are hard to implement, maintain, scale and transfer. They are generally not flexible enough as they cannot be extended to unknown words and infer their lexical na-

ture. The linguist Noam Chomsky gave another excellent example of the challenge with his sentence: "Colorless green ideas sleep furiously." [10]. Despite of the correct syntax, the sentence is incoherent due to the inherent properties of the entities and their possible attributes. Moreover, considering language as an ever evolving instrument that mutates with the time, adapting these rules would be infeasible. Rule based systems were the norm until late 80s. Then, research increasingly turned to machine learning and statistical methods.

The machine learning approaches have ever since been gaining traction. This is because of their capability to produce probability based predictions that can reliably solve multiple tasks and sub-tasks. These methods have attained remarkable results and have proven themselves robust when extrapolated to new data. Another factor that pushed forward the trend is the continuous progress of hardware performance. Deep neural networks are computationally expensive and it is only with the nowadays wide availability of GPUs that the processing power meets the required demand.

1.1 Motivation

A New Milestone in NLP

In the late 2018, the research community in Artificial Intelligence saw a significant advance in the development of deep learning based NLP techniques. This is due to the publication of the paper *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* by the Google AI team [18]. As the title suggests, the work takes a twist on the recent *Transformer* architecture [71] which is solely based on the attention mechanism and defines a novel type of deep neural network arrangement. Their bidirectional learning approach managed to achieve unprecedented performance and pushed the state-of-the-art in 11 downstream tasks such as classification, question-answering, language inference among others. Followed by the open sourcing of their model, academics working with deep learning methods for NLP [67, 75, 44] were able to reproduce such results, as well as fine-tuning the model for their own research tasks.

BERT is an extremely large neural network model pre-trained over a 3.3 billion words English corpus extracted from Wikipedia and the BookCorpus [78] as training dataset. The model has been influenced by the new movement in NLP initiated by *ELMo* [50] and *ULMFiT* [26], that is *transfer learning*. The main idea of this technique is to allow the reuse of existing deep learning models that have been trained from scratch, saving costly computation power by adapting them across different domains, languages and/or tasks [60]. Research data scientist at Deepmind Sebastian Ruder, compares the impact of BERT for the NLP community with

the acceleration that pre-trained models for images ImageNet brought to the computer vision field[1].

Transfer learning in the industry

For businesses specialized in providing technical solutions based in text mining, the introduction of transfer learning in NLP represents a major paradigm shift in the development and training of deep learning models for NLP. *Deepset GmbH*, the machine learning consultancy that supports this current thesis, is highly interested in evaluating the viability and cost-opportunity derived from this approach. Transfer learning and, in particular, domain adaptation would in theory reduce drastically the time required for producing a new model. With the means of adapting a general model to different industry domains in a time and cost optimized manner, transfer learning would reshape the way deep learning solutions are delivered to clients.

1.2 Why German?

Since deepset is based in Berlin, German is a language of interest because of their portfolio of clients. If we consider the linguistic diversity on the Internet, German has been estimated to be the third most common online language after English and Russian[2]. Despite of this, German would represent, in relative value, just 5.9% of the global content. According to W3Techs, this is almost 10 times less than English, which is the international vehicular language sitting in the first position covering 54% of all online content.

The situation is analog in the field of NLP research, primarily due to the fact that German corpora collections suitable for NLP are far less abundant than in English. Secondly, the Internet has become one of the main sources of data for many studies because of its accessibility as well as its exponentially increasing volume. Additionally, English being the lingua franca in academia, the most renowned benchmarks for NLP tasks are, therefore, also aimed to evaluate language models and tasks using text corpora in English. German, despite of being widespread, can be considered a relatively *low resource* language in task-specific datasets and this turns it into an ideal candidate for the application of transfer learning.

[1] `http://ruder.io/nlp-imagenet/`
[2] `https://w3techs.com/technologies/overview/content_language/all`

1.3 Law and NLP

Numerous disciplines generate an extremely high volume of natural language content, but the ones belonging to humanities are definitely the most prominent. From the fields dealing with human culture and society, law and politics are outstanding in complexity. They constitute a great challenge and are therefore a good choices as domains for knowledge extraction. Bringing insight and structure to data that is otherwise highly verbose and contentious is one of the main goals of NLP. This motivated the choice of the legal domain for conducting the current research using the latest NLP models.

Following an interview with Tom Brägelmann[3], lawyer at BBL Bernau Brosloff, we are going to describe in this section the insights about the organization of the German legal system and its entities, the characteristics that mark a difference compared to other legal systems, the current situation of the workforce in law, the available data that could be used for NLP in the legal field and how all these factors represent a great opportunity and motivation for the current research.

German Jurisdiction

In the German legal system, the comprehensive set of legal codes is divided in two major categories: the Public and the Civil law [20]. The Public law comprises four different types of law: the Constitutional, the Administrative, the Administrative civil and the Criminal law. These codes dictate the relationship between a private person and an official entity or between two official entities. On the other hand, the laws that rule the relationship between two private persons are filed under Civil law or also known as Private law. Then, the organization of the German judiciary structure is composed of seven different kinds of courts: Constitutional courts, Ordinary courts, consisting of civil and penal courts, Social courts, Administration courts, Financial courts and Labor courts.

Subjected to centuries of updates to societal changes and influences from other European legal systems, the German justice presents many unique traits. One particular feature that distinguishes the German legal system from the Anglo-Saxon one is, for example, the active role and participation of the judge in the investigation of a case, instead of acting as a mere referee judging the arguments provided by the two opposing parties in a litigation. Another important trait is the importance of law cases. In Germany, there is, in theory, no system of binding precedents, the law cases are therefore referenced for persuasion as an alternative to strictly applying a previous principle. This proceeding fits the decision to each specific case and avoids the generalization of a previous court decision that might, in fact, be erroneous.

[3] https://www.bbl-law.de/de/rechtsanwaelte/tom-braegelmann-llm/

Overview on the German legal job market

After the financial crisis of 2008, the job market for lawyers in Germany was over-saturated as demand dropped drastically [72]. Now, more than ten years later, a decline in the training of new law practitioners is currently being registered, but the situation turned over and this decrease happens in a historical moment when there is actually an increasing demand for lawyers[4]. Germany was among the first European economies to recover from the crisis and re-enter the growth phase. This societal welfare has many consequences and one of them is the increasing capacity for the population to commit time and money to bringing a case to court.

Legal Tech in Germany

During the past years, the technology industry took a great interest in the so-called Legal Tech [14], a field where technology such as Machine Learning and NLP would provide value by assisting in the common tasks that are carried out by lawyers and judges. Machine Learning requires a considerable amount of data to train and be able to output results with accuracy. However, due to confidentiality and privacy issues, legal text corpora such as court decisions and decrees need the consent of the judge to be openly published. This heavily impacts the amount of publicly available documents. The lack of digitization in this field also limits the accessibility of legal documents. Fortunately, projects from the Open Data movement that are concerned about data transparency, with the support of the Open Knowledge Foundation resulted in open legal databases such as OffeneGesetze and Open Legal Data[5]. These sites and other governmental portals are precious sources of labeled data that can be used to train models to carry out relevant text mining for stakeholders in the legal context.

1.4 Research question

Inspired by these latest developments, the goal of this research project consists in determining whether transfer learning, domain adaptation in particular, is a promising technique ready to be adopted by NLP professionals or not. The chosen method to evaluate this is by measuring the effects of inserting domain vocabulary and fine-tuning of a pre-trained model on downstream tasks. The current language domain being considered is the legal field in German. As BERT has been pre-trained using Wikipedia, a multilingual model "BERT$_{Base}$, Multilingual Cased" supporting 104 languages is available. Nonetheless, a multilingual model presents possible

[4] https://www.faz.net/aktuell/wirtschaft/recht-steuern/juristen-erstmals-seit-jahrzehnten-weniger-anwaelte-15038068.html

[5] http://openlegaldata.io/

shortcomings in performance since the number of articles on Wikipedia varies greatly per language. We will therefore operate with our own BERT model pre-trained in German to ensure more robust representations and avoid interference from other languages.

The configuration of different types of laws and courts in Germany is an opportunity for the implementation of several downstream tasks. For example, a classifier: given an extract from a court resolution, the model should be able to classify to which court the decision belongs to. A regression task to predict the litigation cost and amount in dispute is equally viable. A recommendation system of related cases through similarity analysis would be a useful solution for lawyers to research material that could be cited as an argument for their case.

The project aims to answer the main research question:

"What are the effects of domain adaptation in the performance of a pre-trained German BERT model on German legal downstream tasks?".

This main question can be subsequently divided into sub-questions to help us underpin the different aspects that leads to a complete and thorough answer:

1. What are the requirements for domain adaptation using BERT as a model?
2. How does the vocabulary impact the domain adaptation of the model?
3. What improvements can fine-tuning the language model yield for the selected tasks?

1.5 Thesis Outline

The remainder of the thesis is organized as follows: Chapter 2 reviews the background theories that set the foundational knowledge for this research. Chapter 3 analyses the existing related work. Chapter 4 gives an overview of the FARM framework for NLP transfer learning followed by the methodology in Chapter 5. The experiments implemented using FARM and their results are presented in Chapter 6. Finally, the thesis closes with the conclusion and a discussion on further work in Chapter 7.

2 Background

This chapter provides the essential background knowledge for the subsequent chapters. We introduce basic ML concepts. Then, we focus on neural networks which are the specific type of ML models used in this thesis. Finally, the BERT model and the transfer learning technique are fully reviewed for the understanding of the ensuing methodology. The latest deep learning methods incorporated into BERT such as transformers, self-attention mechanisms, are presented to the reader.

2.1 Machine Learning

Machine Learning is a term coined in the late 50's by Arthur Samuel [61], a researcher in the field of Artificial Intelligence, to describe the techniques based in statistical models and algorithms to learn from sample data. When correctly trained, the mathematical model is capable of inferring classification, prediction or decision when given new data that doesn't belong to the training data. From these outputs, higher level tasks, for example anomaly detection, can be derived. The learning of such systems can be mainly conducted in three different ways, *supervised* and *unsupervised learning* [6] and *reinforcement learning*. We will focus on the first two paradigms. The spectrum is far from binary and there are numerous methods that sit in between these two classes of Machine Learning. In our case, the alternative called *self-supervised learning* will be specially interesting for the current research.

Supervised learning

The supervised learning approach requires the training data to be labeled and a variety of machine learning algorithms are based on this type of training: Linear Regression, Logistic Regression, Naive Bayes, Decision Trees, K-Nearest Neighbors and Support Vector Machines, to name a few, but they are mainly aimed at regression and classification. The working principle of these algorithms is the learning of a mapping function:

$$y = f(x) \tag{2.1}$$

For each input *x*, an output *y* is mapped. The annotated data (labeled data) allows the algorithms to derive and optimize the parameters of the mapping function by minimizing the cost function which expresses the total prediction error of the learning system.

Unsupervised learning
On the other hand, unsupervised learning produces models that are able to extract the underlying structure of data without the need of labeling. Generally, considerable time is saved by not having to annotate the input for the algorithm to learn. This kind of algorithm learns without a corresponding target of the output with the help of labels and is therefore more relevant for different purposes than supervised learning. The algorithms under this category are generally aimed towards clustering, density estimation and projections.

Self-supervised learning
A recent form of unsupervised learning that is catching the research community's interest is the self-supervised variant [53]. This method overcomes one of the major obstacles in machine learning, which is the need for large amounts of labeled data. Self-supervised learning leverages unlabeled data by systematically holding back existing information, thus providing surrogate supervision and the model is tasked to train on it. Different patterns of data concealing allow the training of a model on multiple sub-tasks that would comprise together the target task.

2.1.1 Loss Functions

The *loss function* is a method to assess how well a learning system models the data by quantifying the resulting error. It basically outputs the difference between the model's predictions and the ground truth, also known as *loss*. Hence, a lower loss is always desirable as it correlates to higher performance of the algorithm. There are multiple loss functions and selecting the right one is important for the correct evaluation of a model. *Cost functions* are loss functions applied to a set of observations then averaged across them, although these two different terms are often interchangeable. When used for maximization or minimization problems, they can also be referred as *Objective functions*.

Considering *y* the target value, $\hat{y}$ the predicted value, a sample of size *n*, examples of common cost functions for regression include:

L1 Loss or *Mean Absolute Error*:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

L2 Loss or *Mean Squared Error*:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.3}$$

These are two simple ways of quantifying the total distance from the target and predicted value.

For classification tasks (CLS), the functions above do not capture the probabilities of the classes, we need therefore cost functions such as the *Logistic Loss*, *Hinge Loss* or *Kullback Leibler Divergence Loss*. Here, we give the example of Logistic Loss, also known as *Cross-Entropy Loss*, for binary and multi-class classification, where *p* is the predicted probability of a class label *c*, *M* is the number of classes, *o* is a given observation and *y* is a binary value that indicates if a class label *c* is the correct classification for an observation *o*:

Binary Cross-Entropy Loss:

$$CrossEntropyLoss = -(ylog(p) + (1-y)log(1-p)) \tag{2.4}$$

Multilabel Cross-Entropy Loss:

$$MultiCrossEntropyLoss = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{2.5}$$

These loss functions are key to the training of supervised machine learning models. In conjunction with an optimization algorithm, a procedure that we will introduce in the next subsection, they allow the rectification of the parameters of the original mapping function. This leads to the gradual increment of the model's quality after each batch of processed data.

2.1.2 Optimization Algorithms

Optimization in mathematics is the broad family of methods concerning the selection of the best element from a set considering a defined criterion. In Machine Learning, the optimization generally focuses on the minimization of the loss though iterative evaluations using the cost function. One of the simplest and widely used algorithms is the *Gradient Descent*.

Gradient Descent

The gradient descent algorithm [59] is a iterative method that uses the gradient or derivative of the cost function at a given point to determine the next step to consider in order to reach

a minimum. The original algorithm is also known as *Batch gradient descent*, however this version is deemed inefficient due to the calculation of gradients for the whole dataset in order to determine just one update. A formal definition of the algorithm can be expressed as:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.6}$$

where *J* is the objective function to minimize, θ the parameters to update and η denotes the *learning rate*, a hyper parameter that regulates the size of the update step. The equation expresses the decrease of the parameters θ with regard to the gradient $\nabla_\theta J(\theta)$ in proportion to the established learning rate η.

Stochastic Gradient Descent

Numerous optimizations of the Gradient Descent has been developed. For Machine Learning applications, the *Stochastic Gradient Descent* (SGD) solves the deficiencies of the Batch Gradient Descent by performing updates for each training example. Additionally, it allows *online* updates, that are performed freely with new examples without revisiting the whole dataset. When applying the correct learning rate, the convergence to a global or local minimum depending on the convexity of the parameters θ can match the original gradient descent and even avoid local minima thanks to its more granular or noisy update.

The main difference in the formal expression of SGD lies in the training example $x^{(i)}$ and the corresponding label $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.7}$$

Mini-batch Gradient Descent

The *Mini-batch Gradient Descent* is a variation that sits between the Batch and Stochastic Gradient Descent. It updates the parameters not after each training example, but after a batch of examples of a given size, hence the name of this gradient descent. This method proves itself less computationally intensive than SGD due to grouped updates but still preserves the main advantages of the stochastic variant. However, this introduces a new hyper-parameter to be tuned which is the batch size *n*:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.8}$$

The role of the learning rate and its importance to the proper convergence for both Batch and Stochastic Gradient Descent can be seen in Figure 2.1, where 4 different scenarios are presented concerning the relation of η and an arbitrary constant *C* that represents the optimal convergence condition of a given gradient.

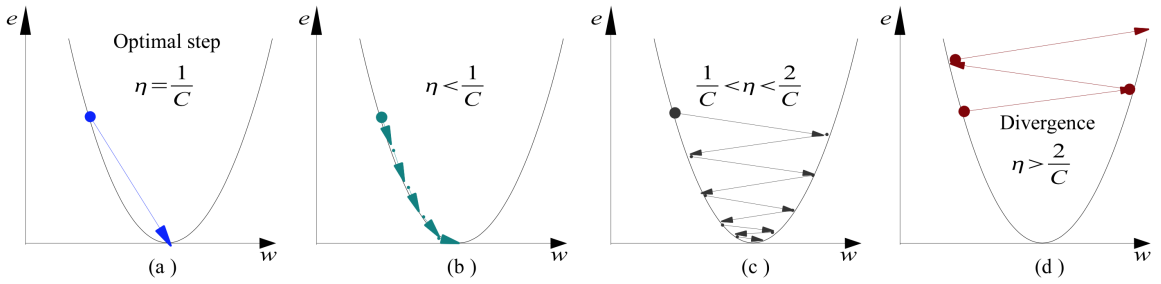

Figure 2.1: Gradient Descent Convergences (Taken from [36])

Adam Optimization

As shown in the scenario *(b)* and *(c)* of Figure 2.1, using a fixed learning rate requires many steps before converging to a minimum, this number may be unacceptably large if the learning rate is too distant from the ideal scenario *(a)*. Research aiming to reduce the number of converging steps found effective approaches that compute adaptive learning rates for each parameter of the objective function. The gradient descent method presents many analogies to the effects of a ball rolling down a slope. The Newtonian mechanics inspired researchers to borrow concepts such as *momentum*[1] and *moment*[2] and apply them to optimization problems.

Adam, short for Adaptive Moment Estimation (Kingma, 2015) [29], is an optimization algorithm specifically designed for multi-layer neural networks. Kingma improves on the findings of Adadelta [77] and the unpublished RMSprop [70]. Adam applies an adaptive learning rate strategy using two moment estimates.

The first moment is the mean m_t and it calculates the decaying average of previous gradients. The second moment v_t is the uncentered variance and it also computes the decaying average of past gradients but squared. They are expressed in the following Equation 2.9 and Equation 2.10 where *t* is the time step, β_1 and β_2 are the exponential decay rates $\beta_1, \beta_2 \in [0, 1)$ for the first and second moment respectively. Then g_t represents the gradient at a given time step and each moment is computed based on their respective past values m_{t-1} and v_{t-1}.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.9}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.10}$$

[1] the quantity of motion of a moving body, measured as a product of its mass and velocity
[2] a combination of a physical quantity and a distance

The authors of the Adam paper noticed that there was a bias towards 0 at the initial steps due to the fact that estimates were initialized as vectors of 0's as well. They decided to apply a correction to circumvent this issue and the resulting moments were modified as following:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.11}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.12}$$

Thus the resulting parameter update step for the Adam algorithm, which adapts from Adadelta including a small number ϵ to prevent any division by zero, is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.13}$$

Kingma suggests that the default values for the newly introduced hyperparameters of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ work favorably.

2.2 Deep Learning

In the history of AI, the field knew two major periods named *AI Winters*. These periods describe a time when the general interest in and support for AI vanished due to the combination of several factors. The reasons for disillusion were such as a low in the hype, technological blockers and the attention of scientists shifting towards other problems. This eventually led to a general stagnation in the research.

2006 marked the end of the second AI winter, when Hinton, Osindero and Teh [24] published their paper about an accelerated learning algorithm for densely-connected multi-layer neural networks. Their work received a great acknowledgment from peers and was considered a major breakthrough. Hinton et al. inspired the research community to retake neural networks seriously by following their approach with deeper networks. Hence the term Deep Learning was coined.

So, Deep Learning is based on neural networks and is a category of Machine Learning methods. Deng and Yu [17] define deep learning as a:

> "Class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification."

Another definition suggested by LeCun [33], creator of the Convolutional Neural Networks (CNN), describes deep learning models as hierarchical probabilistic models that can learn representations with multiple layers of abstraction, and they are generally implemented as deep neural networks.

Given enough data, these multi-layer neural nets are capable of automatically decomposing a problem into smaller and more manageable abstractions. When compared to rule-based methods, Deep Learning tend to generalize better but will still require a rigorous procedure to achieve high performance. A significant number of researchers are now devoted to this fairly novel approach and focusing on this field of AI, mainly due to the interest it sparked by the variety of high-level tasks that it can achieve and by its improved performance. Numerous areas such as speech recognition, computer vision, NLP are already benefiting from the advances in deep learning and deploying systems for commercial use.

2.2.1 Neural Networks

The main approach for Deep Learning, the *deep neural network*, distances itself from the *shallow neural network* by the larger number of hidden layers that form the network. Neural networks are models that can be trained either with supervised or unsupervised learning. They are composed of nodes analog in a certain way to the behavior of biological neurons and their interaction. The manner a neuron would pass along a signal depending on its input, inspired Frank Rosenblatt [58] to conceive the simplified mathematical model of a neuron called the *perceptron*. From that point, researchers derived many models by building more complex artificial neural networks with more nodes, more layers, different architectures and mechanisms to achieve higher performance in specific tasks with specific inputs.

Structure

A typical artificial neural network is composed by an *input layer*, *hidden layers* and an *output layer* of neurons. The number of hidden layers and the amount of neurons per layer can vary depending on the design and purpose of the network. Figure 2.2 is an example of a basic *feed-forward* neural net with 2 hidden layers.

Neuron Output

Each neuron receives one or multiple numeric values as inputs. Each input has an associated *weight* that expresses the importance of the given input to the output that the node will compute (see Figure 2.3). The output y is expressed as the result of the activation function f (section 2.2.1), the example uses the sigmoid function σ (Equation 2.15) to transform the sum of weighted inputs $w^\intercal x$ and biases b (the matrix product $w^\intercal x$ of the transposed vector of

weights by the vector of inputs is a shorthand for the sum $\sum_{x_i w_i}$). This function can be written as:

$$y = f(w^T x + b) \tag{2.14}$$

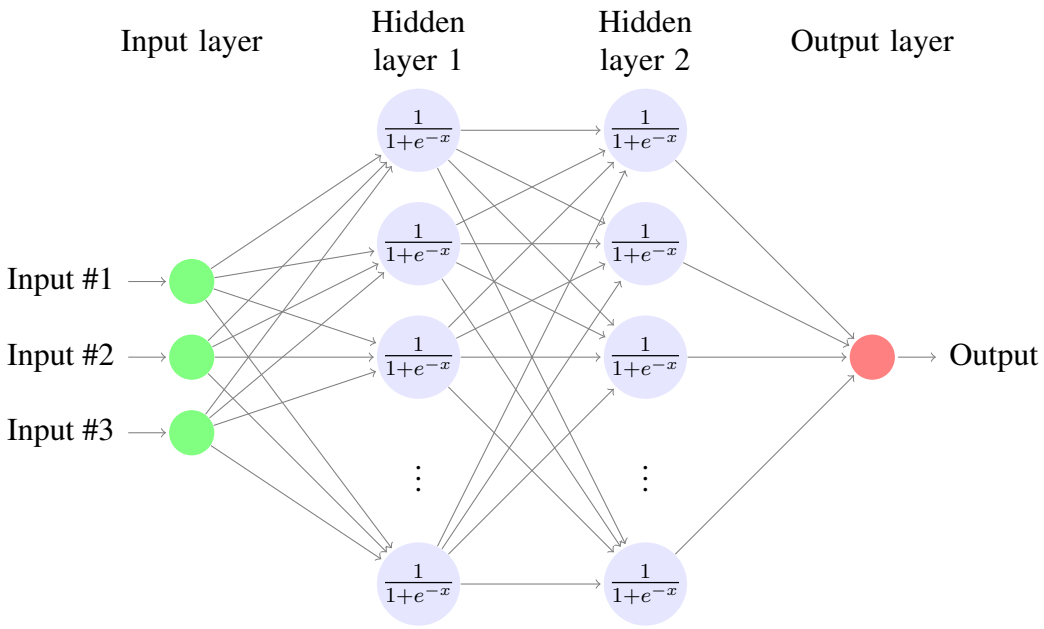

Figure 2.2: Neural network example with 2 hidden layers

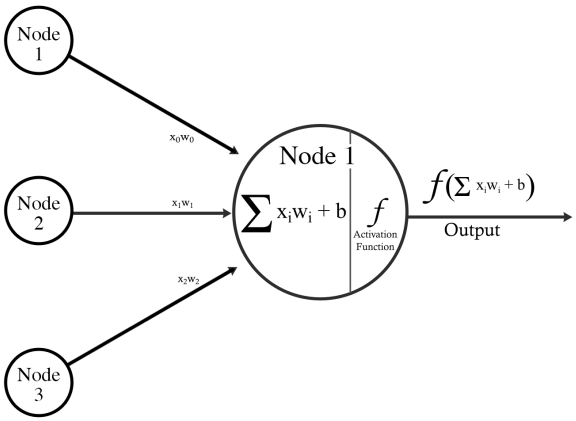

Figure 2.3: Neuron output

Activation Functions

The above mentioned neuron inputs are transformed using an *activation function* (AF). These have a mathematical and biological foundation, since they model the neuronal signal propagation through an action potential also known as *spike* or *nerve impulse*.

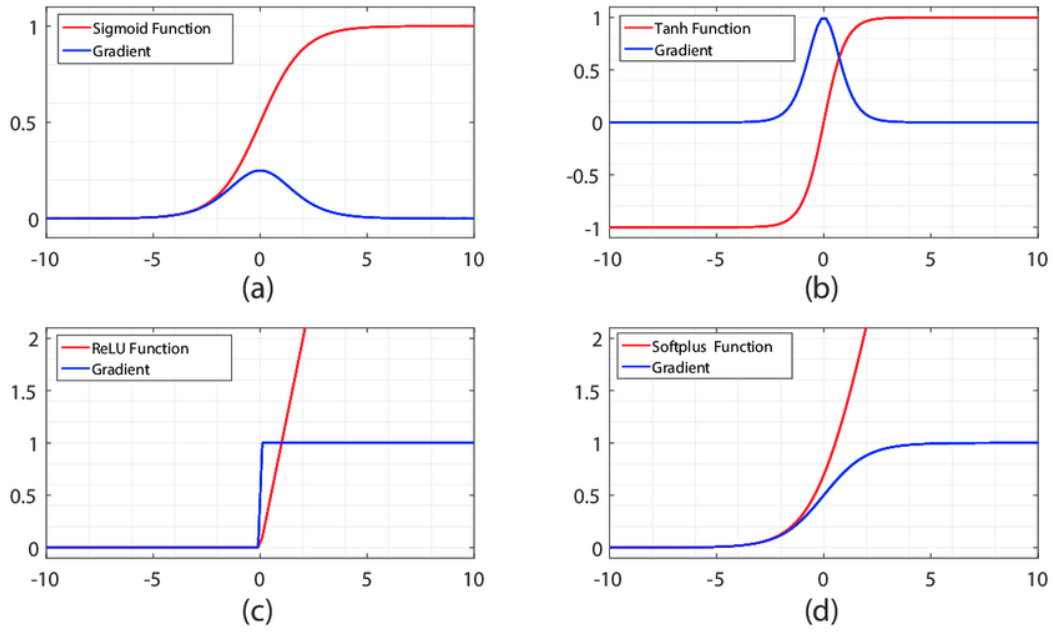

Figure 2.4: Activation Functions Plots (Taken from [57])

The choice of an AF is not trivial and depends on the nature of the considered problem. Deep Learning deals mainly with *non-linear* functions because the expected output is a value ranged between 0 and 1, indicating its degree of activation, whereas *linear* functions would yield unrestricted outputs tending towards infinities. This non-linearity is at the core of the neural network mechanism to model complex problems by abstracting down meaningful features. Typical examples of AF include the *Sigmoid* function (σ) and the *Hyperbolic Tangent* function ($tanh$), as shown in Figure 2.4. The non-linearity is achieved by using the *Euler* constant e and their respective equations and derivatives are:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

$$f'(x) = f(x)(1 - f(x)) \tag{2.16}$$

$$f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.17}$$

$$f'(x) = 1 - f(x)^2 \tag{2.18}$$

The evaluation of subsequent gradient of an AF is key to mitigating certain disadvantages when it comes to applying learning algorithms such as the gradient descent (see subsection 2.1.2). Researchers have incrementally improved the approaches for AF just as they did with the optimizers. Currently, the most popular AF in deep learning is the Rectified Linear

Units (ReLU) [42].

$$f(x) = x^+ = max(0, x) \tag{2.19}$$

$$f'(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & otherwise \end{cases} \tag{2.20}$$

As we can see from the equations and the case *c* in Figure 2.4, the ReLU is much faster to compute than the traditional Sigmoid or Tangent functions because of its linearity for positive values. Two additional benefits make ReLU stand out.

First, its sparsity. This should not be confused with data sparsity, which denotes missing information. Model sparsity refers to displaying fewer features and the ability to differentiate them properly. A model showing the opposite is considered dense. The sparsity of ReLU is observable in the regime $x \leq 0$: the function strictly generates 0 and this helps a faster convergence using the output of ReLU. The Sigmoid and TanH functions on the other hand tend to generate non-zero values resulting in higher density.

Second, when $x > 0$, the gradient of ReLU is constant, contrary to the diminishing gradient of the Sigmoid or Tangent gradients. The stable gradient leads to faster learning and is unaffected by the problem of vanishing gradients that would prevent the weights of a neural network from meaningful readjustments.

This activation function has already successors like the *leaky ReLU* (LReLU) and the *parametrized ReLU* (PReLU) [11] but they come with different advantages as well as downsides, for example expensive computation. ReLU remains therefore as a solid referent.

2.2.2 Error Backpropagation

The crucial mechanism that leads to the enhancement of deep neural networks is the *error back-propagation* that readjusts the weights of the system. The technique has been taking shape since the 80's but it holds its modern form from the work of LeCun [33].

During training, a neural network propagates the input data forward through the layers of neurons, so from the input layer, through the hidden layers until the output layer. This phase is called the *forward pass* and the parameters (weights) of the neurons shape the resulting predicted values at the output layer. This prediction is then compared to the target value and the deviation is measured with a loss function. Then this error is back-propagated through the network informing each neuron about their parameter distance to the ground truth value and allowing the correction of their weights.

The direction of less error is determined by using optimizers like the gradient descent (subsection 2.1.2), although the particular hierarchical setup of the neural network requires a different way of computing the cost function's gradient. The mathematical principle that makes this possible is the reiterative application of the *chain rule* [43]. Doing so, it is possible to decompose the functions comprised in a node and calculate the partial derivative of the error:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta o_j}\frac{\delta o_j}{\delta w_{ij}} = \frac{\delta E}{\delta o_j}\frac{\delta o_j}{\delta net_j}\frac{\delta net_j}{\delta w_{ij}} \tag{2.21}$$

where E is the loss, w_{ij} is the weight parameter between a neuron j and the neuron j from the previous layer, o_j denotes the output of the previous neuron and net_j is the weighted sum of outputs o_j.

2.2.3 Convolutional Neural Networks

In order to solve problems of different natures more efficiently, researchers explored alternatives to the traditional feed-forward networks. Within the Computer Vision field, Convolutional neural networks (CNN) [54] are highly popular, they are a class of neural networks that are primarily used for image processing. Image classification, clustering, object and optical character recognition are some of the applications.

CNNs have characteristic design concepts such as the convolutional and pooling layer that reduce the amount of parameters and the dimensions of the data. The convolutional layer applies a filter that processes sequentially parts of the input matrix that represents an image, for example, and transforms this input into a smaller matrix by the means of dot product operations. Doing so, the features, such as high contrast areas, edges, and contours are extracted. The pooling layer works in a similar way but the objective is to compress the information and turn it computationally more manageable. It operates by applying a filter that calculates the maximum or the average of the submatrices. These architectural elements in combination with the proper tuning of hyperparameters and regularization methods augment considerably their efficacy.

2.2.4 Recurrent Neural Networks

While static visual information can be efficiently processed by CNNs, these are however not the ideal approach to other forms of data that are sequential or time-dependent. Learning from sequential data is better handled by a category of specialized neural networks called Recurrent

Neural Networks (RNN) [37]. This architecture is especially relevant for NLP since a text corpus is made of sequences of words and therefore sentences.

Sequential data is commonly divided by time and RNNs accept inputs that correlate with data at a given time step. Their most prominent feature is the incorporation of a feedback loop. Every time step's output is fed back to the network, this provides a record of the previous state that will affect the output of future steps, hence the name "recurrent". The persisting information lets the network process upcoming inputs taking the previous ones into consideration. The basic recurrence can be expressed as:

$$h_t = f_w(h_{t-1}, x_t) \tag{2.22}$$

where h_t is the new hidden state is computed with some function f_w with parameters w, h_{t-1} is the previous hidden state and x_t is the input at time step t.

The recurrent connections of an RNN can be visualized as *unfolded* or *unrolled*, see Figure 2.5. Here, the original layer is replicated as many times as necessary to cover all the time steps to process the whole sequence. Every replica shares the same parameters and the backpropagation is now called *backpropagation through time* (BPTT) because the gradients are cumulative through the time steps.

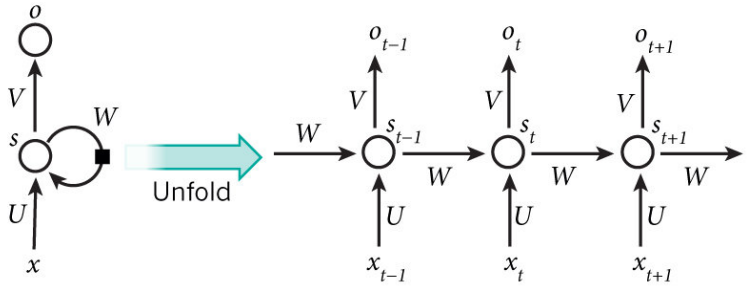

Figure 2.5: Unfolded Recurrent Neural Network (Taken from [33])

However, the classical RNN present an important caveat. Because the gradients are accumulated, thus multiplied with the same shared parameter the same amount of times as the sequence length. When this becomes excessively long, the gradients have a tendency to either *explode* (reaching incoherent large values) or *vanish* (values tending to zero).

Long-Short Term Memory

The Long-Short Term Memory (LSTM) [25] is a RNN that solves the above mentioned gradient problems by introducing the concept of *gates*. These elements help regulating the flow

of information inside the LSTM unit. The usual gates that are included are an *input gate* i_t, an *output gate* o_t and a *forget gate* f_t. On top of that, LSTM maintains two hidden states at every time step. First, the *hidden state* h_t which is already present in traditional RNNs. Second, the *cell state* c_t which behaves as a memory that interacts with the gates. A LSTM can be described as:

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ g_t &= tanh(W_g[h_{t-1}, x_t] + b_x) \\ c_t &= i_t \odot g_t + f_t \odot c_{t-1} \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{2.23}$$

where g_t can be seen as a supportive gate that computes how much to write to the cell state, $\odot$ indicates the element-wise product and W and b are respectively the weight matrices and bias vector parameters which need to be learned during training.

The main idea from the LSTM is not only to assess the impact on the hidden state of each word in the sequence, but also the words that are not meaningful enough and are thus safe to "forget". In addition to these mechanisms, the way the units are connected through the internal cell states carries the gradient forward and backwards in a cleaner flow reducing the likelihood of gradient deterioration.

2.3 Natural Language Processing

2.3.1 Language Modeling

A language model [22] exploit through observations the characteristics of a language and how the words relate, instead of describing it with rules, which would grow too complex. It is a probabilistic model that is able to predict the word that will follow given a sequence of words. In more elaborated models, more context will be taken into account, from sequences of previous words, to sentences, paragraphs or entire documents. One can use a language model to predict the continuity of a sentence but also to generate sentences.

N-Gram Models

An example of Language Model is the *N-gram* model. N-grams are simple models that are defined by word sequences of length N. When $N = 1$, known as unigram, each word is taken

as a unit and its probability is calculated by counting its occurrence in the document and dividing by the total amount of words. For $N = 2$, a bigram, the probability calculation becomes conditional taking into account the previous word and thus applies the same assumption as the Markov condition. Finally for the rest of N-gram models, the calculation can be generalized by considering all the $N - 1$ precedent words. The effectiveness of the different N-gram models depends on the length of the targeted corpus data as well as the vocabulary that the model can recognize.

2.3.2 Encoder-Decoder Model

An essential building block for NLP using deep neural networks is the *Encoder-Decoder* architecture [68]. This design is composed of two blocks. The encoder block is responsible for encoding an input sequence into a fixed dimensional representation vector, also known as the *context vector*, which acts as the final hidden state of the encoder. This representation should encode enough information that the input can be recreated. Then, it gets fed to the decoder block which will then produce the output using only this internal representation. The encoder and vector blocks are commonly implemented as LSTMs and this architecture is used for sequence-to-sequence tasks such as machine translation. Advantages of the encoder-decoder include the capacity to process sequences of arbitrary length into a fixed vector representation and connect encoders to different decoders for training by passing the intermediate encoded representation.

2.3.3 Word Embeddings

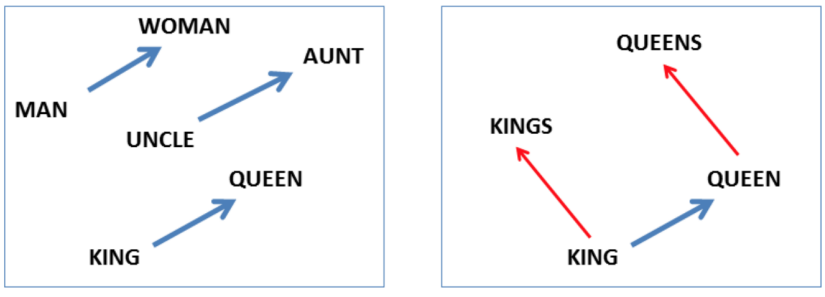

Figure 2.6: Projected relationships between word embeddings. (Taken from [39])

So, language models are statistical approaches, they require therefore quantifiable and continuous representations. But words, on the other hand, are discrete units. A straightforward

approach to represent them in a quantifiable way is to encode the features with a vector and for every word's feature identify its location in a binary way, this is known as the *one-hot vector*. It's simplicity is shadowed by the high dimension of these vectors and the complexity that it carries. To escape this limitation, the encoder-decoder model comes into play and is used to encode a representation with a reduced dimensionality. A text corpus can then be encoded into numerical vectors, also known as *word embeddings*.

Once words are encoded, a subsequent vector space model (VSM) is modeled. Then, simple linear algebra can be applied, enabling the calculation of the relationship between words. Figure 2.6 showcases the famous example of King and Queen's words association, the blue arrow represents the vector projection modeling gender whereas the red one models the plurality. Tomas Mikolov demonstrates that the same equation *vec("King) - vector("Man) + vector("Woman") = vector("Queen")*. This kind of composition can be extended to other entities and their attributes, such as countries and their languages or currencies. This leads to the retrieval of basic inherent properties such as similarity and weighting. From this point, advanced applications such as document similarity, term frequency and matching can be derived.

Different word embedding approaches have been implemented and pre-trained models have been published, well known examples are *Word2Vec* [40], *Glove* [49] and *FastText* [23]. Word2Vec is the referential distributed word vector model that encodes words into embeddings using two different language modeling methods: the *continuous bag-of-words* (CBOW) or the *Continuous Skipgram* models. CBOW as its name states, is based on the BOW model but it predicts the probability of a target word considering the surrounding context words within a window of a given size instead of the whole document. The Continuous Skipgram model is the opposite and it tries to predict the context words given the target word.

The word embedding models present numerous limitations concerning the length of the used corpus, the order and independence of words, but the major downside of the vector approach is the inability to represent multiple meanings of a word. This is due to the association of a single representation per word. When a word appears simultaneously in different contexts, Word2Vec is unable to accurately learn its semantic nor syntactic nature, for example: *"a bank of fish"* or *"a bank holiday"*.

2.3.4 (Downstream) NLP Tasks

NLP provides nowadays a wide array of tasks to tackle problems of different scales from part-of-speech tagging (POS) to Dialog systems. It is common for complex NLP tasks to be broken down into multiple sub-tasks to attain the desired goal. When applying transfer

learning (section 2.4), it is typical to use the term *downstream task*. There is no consensus in the definition for it but Jay Alammar[3], former Deep Learning content developer at Udacity[4], provides a concise one: *"downstream tasks is what the field calls those supervised-learning tasks that utilize a pre-trained model or component"*.

No matter the approach, –rule-based, machine learning or deep learning– NLP tasks can be divided in the following categories:

Text Classification Tasks
Text classification generally doesn't need to preserve the word order. The methods for this task usually process the corpus as a whole with an approach similar to the bag of words. It is used to predict labels and categories based on the dominant content, but it is also frequent to see sentiment analysis. It is applied for offensive language and spam detection as well as supporting the proper taxonomy of documents.

Word Sequence Tasks
Contrary to text classification, the word order is important for this kind of task as it deals with sequences. The word order is especially relevant for language modeling (subsection 2.3.1), therefore the derived tasks include prediction of previous and next words. Some models are capable of extending the prediction to complete sentences. Another general capability is the generation of text recursively inferred from the next sentence prediction. Notable applications of this kind of tasks are *Named Entity Recognition* (NER), Part-of-Speech tagging ()POS), language translation and text completion.

Text Meaning Tasks
Extracting the word embeddings (subsection 2.3.3) of a corpus, text meaning focuses on semantics. This is generally used for tasks such as search, topic modeling, question answering. The association of meaning to a word is well achieved in NLP, however, capturing the meaning for sentences or documents presents challenges that current studies are still looking into.

Sequence to Sequence Tasks
This category could be considered an extension of the word sequence tasks. Also known as seq2seq, these tasks take a sequence as an input and output a transformed one. For this purpose, encoder-decoder methods and hidden representations are used. Common applications are translation, summarization and Question Answering (QA) among others.

Dialog Systems
NLP is fundamental to power conversational agents. These systems require high performance

[3] `http://jalammar.github.io/illustrated-bert/`
[4] `https://www.udacity.com/`

in natural language understanding to correctly detect users' intent. Moreover, the agent is expected to provide an answer. For this, the system can combine tasks from the different categories above mentioned to achieve both understanding and answer generation tasks. Depending on the scope, integrating world knowledge is necessary.

Dialog systems can be split into two types, goal-oriented and conversational. The first one aims to fulfill the intents of the user in a defined context and usually replaces the graphical user interface where the desired transactions would be communicated. Many enterprises integrate goal or task-oriented dialog as an interface for their services, a clear application can be found in the hospitality industry, where concierge services for reservations and bookings are increasingly being supported by goal-oriented dialog systems. The second system is broader and without a specific end. Purely conversational agents have no other purpose than keeping up a dialog flow as human as possible. They present challenges beyond NLU and answer generation, that include maintaining the state of the conversation, logical reasoning of the input through world knowledge or paying the adequate attention to the different topics that are being discussed. The conversational agents require, in other words, a certain capacity of memory and active learning in order to emulate a human dialog. Nowadays, conversational bots such as Mitsuku[5] are highly performant in unrestricted Turing tests.

2.4 Transfer Learning

Transfer learning [30] is a sub-field within ML concerning the relation between the applied datasets used for training and evaluation, and the overall underlying distribution. Pan and Yang [47] define transfer learning as:

> Given a source domain $\mathcal{D}_S$ and its learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and its learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function f_T in $\mathcal{D}_T$ using the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$.

Thus, transfer learning defines an approach in which a base model from a certain source domain aimed for a task *A* can be repurposed to solve a different target task *B* possibly belonging to a different target domain. Instead of training an entire model for each specific task from scratch, the main idea is to only further train a base model with additional data which is better suited to the target task, as seen in Figure 2.7. In certain cases, removing the interference of the original domain would be desirable.

[5] https://www.pandorabots.com/mitsuku/

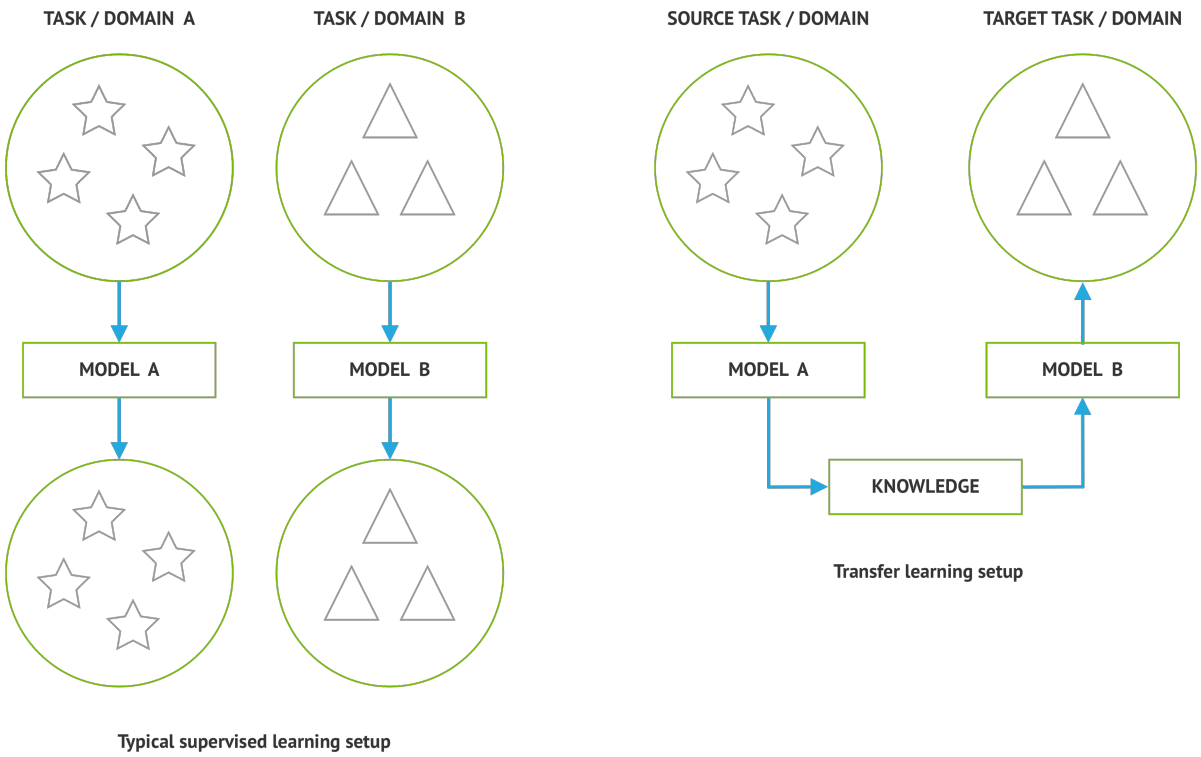

Figure 2.7: Traditional ML setup vs. Transfer learning setup

The different categories of transfer learning are classified according to several situations. First, whether the data is labeled in the original and the target domain, and second, the difference between original task and the target task. Figure 2.8 summarizes concisely the existing taxonomy.

Ruder [60] proposes a scenario called *positive forward transfer*, when the transfer learning is successful, in other words, when the performance of the target task using the fine-tuned model increases. In contrast, the opposite scenario, the *negative forward transfer* can be observed when fine-tuning harms the target task's performance. The degradation of the pre-trained model's performance after the supplementary training is commonly due to the dissimilarity of the new input. When the datasets for pre-training and fine-tuning are too distant, for example two completely different languages, the weights of the model can revert to a random state and lead to the observation of the phenomena called *catastrophic forgetting*.

The motivation that supports transfer learning includes numerous advantages. The lessened amount of data required to adapt the base model to another domain and/or task, the subsequent time and cost efficient training, and the overall good results are the main benefits that allowed deep learning practitioners to quickly derive new models to handle different tasks.

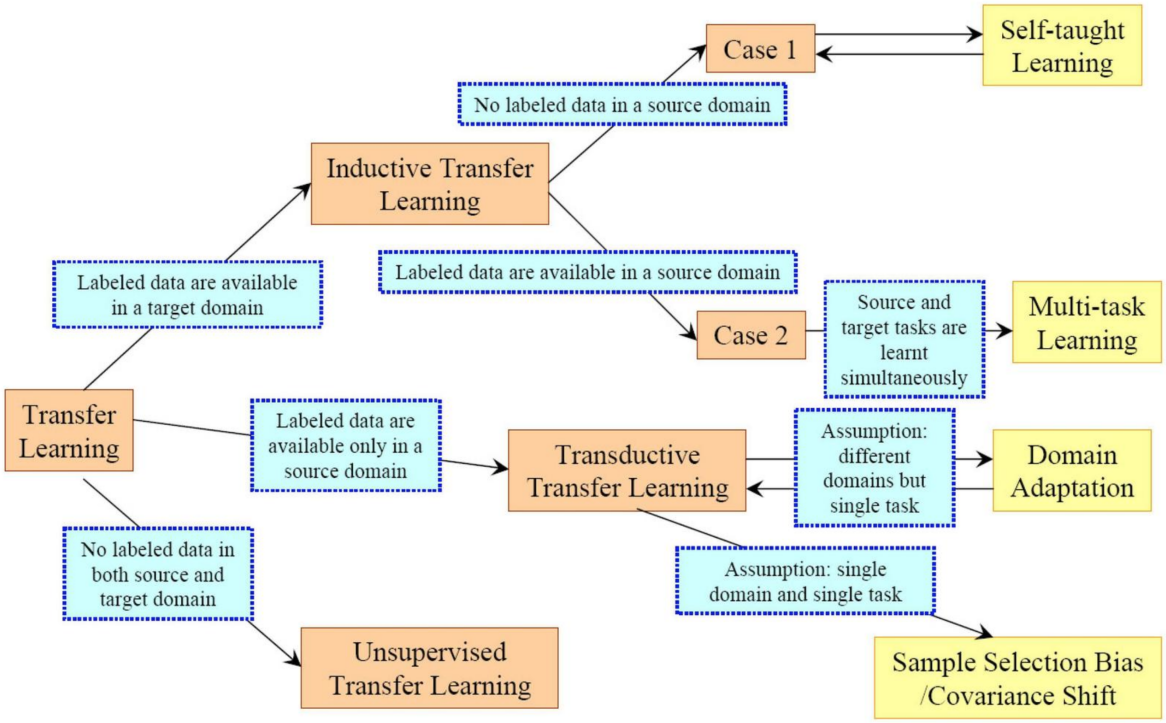

Figure 2.8: An overview of different settings of transfer learning (Taken from [47])

2.4.1 Fine-tuning

A well known approach to apply transfer learning to neural networks is copying and training the first n layers, n being a variable that can be selected depending on the required specificity to be retained for a certain target task. Two distinct approaches exist. On one hand, the *fine-tuning* method, where the error on a specific task will be back-propagated and the original weights will be readjusted. On the other hand, the *frozen layers* approach, in which only the last layers will learn from the new data, the rest of the copied layer weights will remain unchanged. The rationale for these approaches correlates with the findings of Yosinki about the transferability of features in deep neural networks [76]. The study shows that the first (or lower) layers of a deep neural network commonly encode general information and the last (or higher) ones become increasingly specific. The transfer learning approach tends to hold the first layers more valuable considering their capability to generalize and serve for a broader range of domains and tasks.

2.4.2 Domain Adaptation

The notion of domain adaptation is a class of transductive transfer learning. The definition of domain differs in each field of ML and the main adaptations encompassed in NLP [35] are the following ones:

- Adaptation between different corpora
- Adaptation from a general dataset to a specific dataset
- Adaptation between subtopics in the same corpus
- Cross-lingual adaptation

The current thesis focuses on the adaptation of a model trained with general language to the legal language, the domain adaptation corresponds therefore to the second category listed above which is the adaptation from a general to a specific dataset. The corpus data from the specific domain is more likely to display a particular vocabulary belonging to the given field, but could also attribute different semantics to words that are common to the general domain language. Furthermore, the distribution and the frequency of the terms are possibly skewed.

A relevant vocabulary is the cornerstone of many NLP applications. A representative vocabulary is essential to produce meaningful word embeddings and, consequently, a body of research has established that the low performance of NLP models on unfamiliar domain data is due to the effects of out-of-vocabulary (OOV) words [7] [15]. Other studies show that the insertion of domain specific vocabulary as an adaptation strategy leads to better performance of their language models. [45][19][9]

2.5 BERT

The BERT [18] model's architecture is composed of 12 bidirectional Transformer encoder blocks [71], 768 hidden layers, and 110M parameters and is heavily based on Attention mechanisms [38]. Another important design characteristic of this model is its bidirectionality that makes BERT different from traditional left-to-right trained language models. This would be the case of the GPT Transformer model from OpenAI [52] that processes sequences in the same fashion as reading a sentence in English. ELMo [50], on the other hand, concatenates an LSTM trained left-to-right and another one right-to-left. It can be therefore considered bi-directional, but ELMo is not a single neural network trained simultaneously in both directions.

In this section, we introduce the concepts of Attention and Transformers that are essential to the understanding of BERT's architecture. To complete the overview on BERT, we provide details on the pre-training and fine-tuning of the model.

2.5.1 Attention

Neural networks implementing the encoder-decoder model (subsection 2.3.2), which is common for solving sequence-to-sequence (seq2seq) [68] problems, use a fixed-length context vector for internal representation. The fixed size of this vector makes this method ineffective when dealing with longer sequences since it can't retain all the information and tends to "forget" the initial inputs. Attention mechanism solves this problem by adding an additional layer that normally sits between the encoder and the decoder operating on the context vector to help the decoder capture global information from the input sequence. This layer doesn't handle the original reduced representation context vector, but weighs all the outputs of the encoder, calculates the weighted sum and feeds the result to the decoder. It provides the hidden state of all the encoder nodes to the decoder, acting as a memory.

In his work on machine translation using deep neural networks, Bahdanu [2] proposes an alignment model to train neural networks to produce accurate translations with the help of attention. Given an input sequence in English and another one in French, the model tries to score the best matching words between the two inputs. For this purpose, the author uses a bidirectional RNN encoder and an alignment function that assigns the score for each pair of words, in his proposed solution the function in this case is a non-linear $tanh$ activation function.

The example above describes a common scenario of how attention can be applied to improve seq2seq tasks. However, the concept of attention spawned into a family of attention mechanisms that differ in alignment score functions and other properties. We will briefly explain the categories of *Self-Attention* and *Multi-Headed Self-Attention* that are relevant to the Transformer architecture.

Self-Attention

Self-attention, also known as intra-attention, is a mechanism that applies attention not between two different sequences but to the same sequence instead. Each word of the input sequence is paired with previous words with the highest attention score in order to compute a related representation. This is particularly useful to the disambiguation and resolution of coreferences and pronouns. Applied to NLP, it has a positive impact on tasks such as machine reading comprehension or summarization.

2.5.2 Transformers

A Transformer [71] is a neural network architecture proposed by Waswani et al. in 2017. The Transformer implements the encoder-decoder pair but has a completely different structure from RNN and CNN in design (see Figure Figure 2.9). Instead it relies completely on multi-head self-attention mechanisms in each encoder (left block) and decoder (right block). This kind of network is proven highly performant for solving language-oriented problems such as syntactic parsing and language translation through sequence transduction.

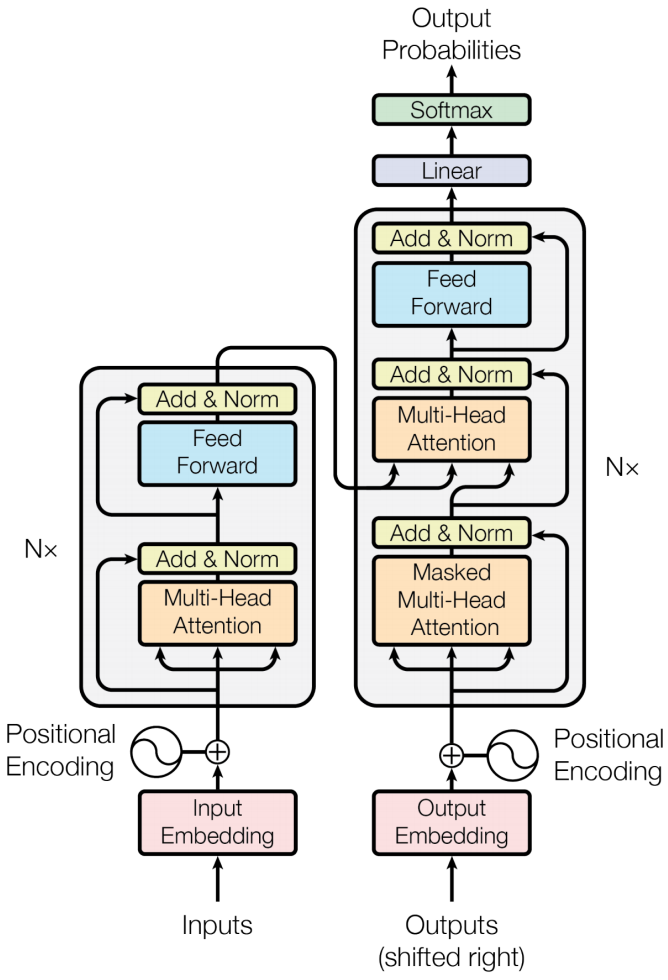

Figure 2.9: Transformer Architecture (Taken from [71])

The following paragraphs will proceed to explain each component of the Transformer, from bottom to top.

Positional Encoding

The positional encoding appends necessary information about the position of the tokens in

the sequence. This is required because the Transformer model doesn't rely on recurrence nor convolutions. The authors propose using sinusoidal functions sinus and cosine to encode the position of the token and the dimensions of the vector.

Multi-Headed Self-Attention

In the Transformer structure, we can see how the encoder processes input embeddings into the attention sub-layer, which is in fact a multi-headed self-attention (section 2.5.1). Before going into details with the Multi-Head attention proposed by Waswani, we need to explain the different view adopted by the author. Their approach suggests to consider the attention function as a mapping of a *query* (Q) and a *key-value* pair (K, V), where the output is the weighted sum of the values and the weight assigned to each value is computed by a alignment function of the query and the corresponding key.

This alignment or compatibility function as they name it is described as follow:

$$Attention(Q, K, V) = softmax(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{2.24}$$

where Q, K and V are respectively the query, key and value. d_k refers to the dimension of the queries and keys.

The *softmax* function is a normalization function that accepts a vector z of K real numbers and transforms the components of the vector into a probability distribution, that means after applying softmax, each component z_i of the vector will be bound by (0,1) and the sum of all of them will yield 1. The formula is given as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad for \quad i = 1, ..., K \tag{2.25}$$

The multi-head mechanism runs through the scaled dot-product attention multiple times in parallel. The resulting scaled dot-product attentions are then concatenated and linearly transformed into the expected dimensions. The authors claim that this approach permits the attention take into account *"different subspace representations at different positions"* improving the performance of single head attention mechanisms. Figure 2.10 illustrates the processes described above.

The output of the attention is normalized and passed to the feed forward network. Consequently, the output of the encoder block is fed to the decoder, directly to the block which as an identical structure as the encoder, except this block is preceded by a Masked Multi-Head Attention that processes the output embeddings. This masking and the shifting of output

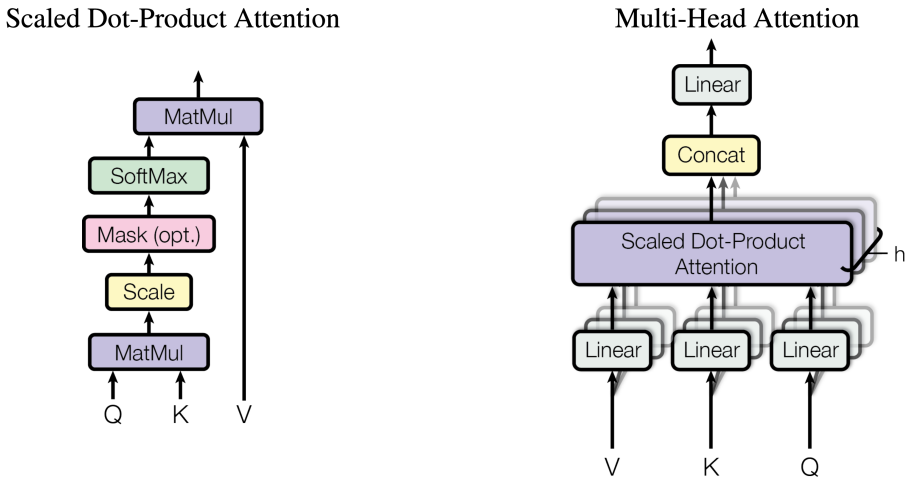

Figure 2.10: Multi-headed scaled dot-product self attention (Taken from [71])

embeddings to the right are measures to make sure that the predictions are based on known outputs for previous positions.

Residual Connection and Normalization

We can observe in figure Figure 2.9 *residual connections* (also known as *skip connections*) short-cutting every sub-layer to the normalization layers. These skip connections serve for both forward and backward passes and are mechanisms to avoid the problem of vanishing and exploding gradients. We've seen similar workarounds in LSTM (section 2.2.4) by preventing the repeated flow of gradients through non-linear activation functions such as sigmoid and tanh. The normalization is applied to mitigate the problem known as *covariate shift*, which refers to the distributions of the training and test sets being different. This disparity is reduced by fixing the mean and the variance of the summed inputs of the layer.

Feed Forward Networks

The sub-layers that transform the output of the encoder and decoder are fully connected feed-forward networks. According to the authors of the Transformer paper [71], they implement two linear transformations with a ReLU activation in between.

2.5.3 Model Pre-training

The pre-training of BERT has been performed on 40 epochs over a 3.3 billion words English corpus using the English Wikipedia dump and the BookCorpus [78] as training dataset. The language modeling performed by BERT follows two different strategies: first, the masked language model (MLM) and the next sentence prediction.

Masked Language Model

In order to train a model that takes into account the surrounding context of a given word and not just the sequence that preceded it, BERT relies on bi-directionality. BERT's language modeling masks randomly 15% of the input tokens. This produces a self-supervised training setup in which the language model is tasked to predict only the masked tokens. By doing so, the model can learn representations accurately in both directions simultaneously without the risk for the model knowing before-hand what token is coming next.

Next Sentence Prediction

The second training method that BERT adopts to model its language is the next sentence prediction. The purpose is to enhance the understanding of the relationship between sentences. The algorithm selects sentence pairs A and B in two different ways: i) B is indeed the next sentence of A and ii) B is randomly chosen and therefore not related to A. The model learns the correlation of the next sentence by training 50% on case i) and another 50% on case ii).

2.5.4 Model Fine-Tuning

BERT's input sequence is an array of tokens formatted with the special [CLS] hidden state token at the beginning and the [SEP] separator token at the end. [CLS] is a token that encodes the whole input for classification tasks. Note that when the sequence includes two sentences, these should also be split with [SEP] to mark the transition between segments. The figure Figure 2.11 shows how an input is broken down into token, segment and position embeddings.

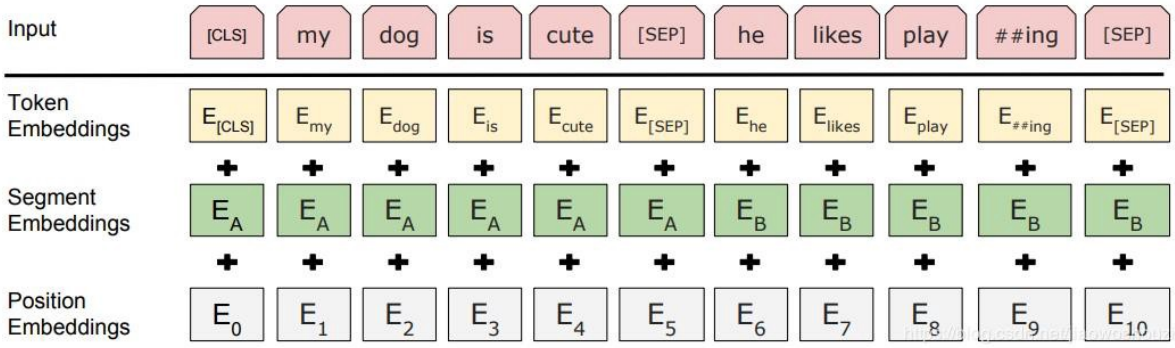

Figure 2.11: BERT Input Example (Taken from [18])

The tokenization applied by BERT relies on the *WordPiece* tokenizer which allows the splitting of a word into more than two tokens. The main idea of dividing words into sub-units allows the model to identify more words that belong to the existing vocabulary. In the ex-

ample provided by Devon, "playing" is split into "play" and the token "##ing", the prefixing of "##" denotes that this unit is a suffix of another token. This technique helps reducing the instances of tokens that are out of vocabulary (OOV).

For fine-tuning, task-specific inputs should be fed following the format described above. The provided example input is optimized for the next sentence prediction task, but it can be extrapolated to other pairwise downstream tasks such as QA, paraphrasing and so on. However, we must consider the limitation that BERT presents for the maximum sequence length which is 512 tokens, including [CLS] and [SEP] tokens.

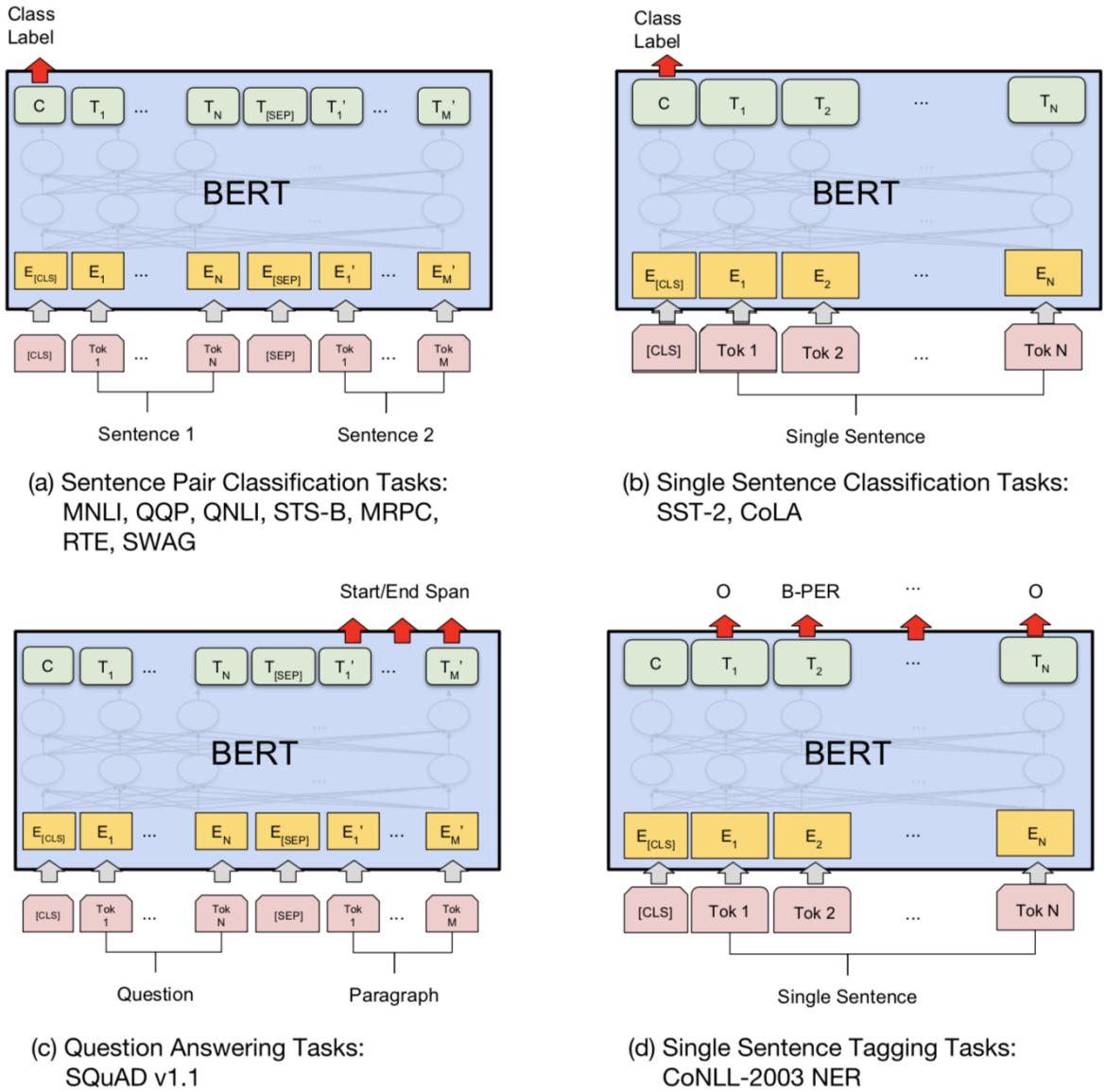

Figure 2.12: Downstream tasks fine-tuning using BERT (Taken from [18])

Then, the input sequence will be transformed into token representations that will be subsequently inputted into the output layer. We adapt the output layer and use the pertinent representations according to the task as shown in the figure Figure 2.12. The first downstream task example a) provided by the authors is a Sentence Pair Classification Task and the binary prediction is inferred from the Class Label C, that indicates if Sentence 2 follows Sentence 1. The task b) represents a single sentence classification task, and similarly we use the hidden state derived from the [CLS] token to show us the predicted label for the inputted sentence. For these both classification tasks, the prediction is extracted by applying softmax on the hidden state of [CLS]. The Question Answering and Sentence Tagging or NER are token based tasks so the [CLS] token won't be use anymore, instead probabilities will be calculated per token.

2.5.5 Feature Extraction

Contextualized word embeddings, in other words, embeddings that are trained bidirectionally trained by BERT, can be extracted and applied to custom models in a similar way to how the other word embeddings can be used. This approach is called *feature-based* and the embeddings diverge greatly depending on the layer where they are extracted. The authors of the BERT paper indicate that the best results were found concatenating the last four hidden layers of the model.

3 Related Work

This chapter reviews the most relevant work related to transfer learning methods using context vectors. Ensuing the release of BERT a body of research has studied the adaptation of this particular model to different scientific fields. We then conclude the review by examining the NLP research applied to the legal domain.

3.1 Transfer learning in NLP

The papers introducing *ULM-FiT* and *ELMo*, both published in 2018, received high accolades from the NLP research community. They furthered the possibilities of transfer learning with novel approaches and opened the path to the research direction leading to BERT.

3.1.1 ULM-FiT

ULM-FiT stands for Universal Language Model Fine-Tuning and it is the research on transfer learning presented by Howard and Ruder [26]. The authors proposed in their paper the concept of pre-trained language models. Their method is to train on a very large corpus of text to obtain a general language model, then fine-tune it for any classification task. In the study, a variant of LSTM called *AWD-LSTM* without any short-cut connections nor attention was used. For the fine-tuning, Howard and Ruder suggest to use the frozen layers approach (subsection 2.4.1) and gradually unfreeze and train the deeper layers to the network.

The researchers claim that ULM-FiT works across tasks varying in document size, number and label type. They point out that the approach requires no custom feature engineering and doesn't require additional in-domain documents or labels. Fulfilling the goal of transfer learning, they managed to achieve the same error rate of a model trained from scratch on 20.000 examples by fine-tuning a pre-trained model with only 100 examples.

3.1.2 ELMo

Embeddings from Language Models (ELMo) [50] is the approach suggested by Peters et.al which was published almost simultaneously together with ULM-Fit. The novelty of their model is the capacity to produce contextual embeddings and the bi-directionality of the language model (biLM). However, the latter is not trained simultaneously, but rather a concatenation of *Forwards* (predicts the following word given previous words) and *Backward* (predicts precedent words given posterior words) LMs. ELMo also uses a pre-trained LSTM-based LM and extracts each layer's hidden state for the input sequence. The final embedding for each word is calculated as the weighted sum of all those hidden states.

The obtained word embeddings are compatible with other existing embeddings such as Glove, Word2Vec or FastText (subsection 2.3.3). In fact, the authors recommend to concatenate them together. ELMo reported state-of-the-art results in 6 different downstream tasks including classification, QA and NER.

3.2 Domain Specific BERT Models

Soon after the publication of BERT, several researchers tested the domain adaptation capabilities of BERT by fine-tuning the language model to specific fields. It has been applied to scientific, biological and clinical data yielding respectively the *BioBERT*, *SciBERT* and *ClinicalBERT* pre-trained BERT language models. We will focus on the first two as their approaches are significantly different from each other but relevant for our current study.

3.2.1 BioBERT

The first domain-specific pre-trained BERT is BioBERT [34], which is optimized for the biomedical field. The authors Lee and Yoon use 4.5B words from the PubMed corpus and 13.5B words from PMC, both archives of biomedical and life science literature. The training of several models was reported to last between 10 days and 23 days depending on the amount of data used and the dataset combinations. The hardware that served the training was composed of 8 NVIDIA V100 GPUs with 32GB memory.

They initialize BioBERT with BERT$_{Base}$, that means, the language model is built on top of the learned weights from English Wikipedia and BookCorpus. Using different combinations of datasets for language model adaptation, the researchers found that the best results were found when combining both PubMed and PMC corpus through 470K pre-training steps, they

labeled this BioBERT v1.0. Later, they release a v1.1 using only PubMed but pre-trained with 1M steps and this yielded general performance improvements over the v1.0.

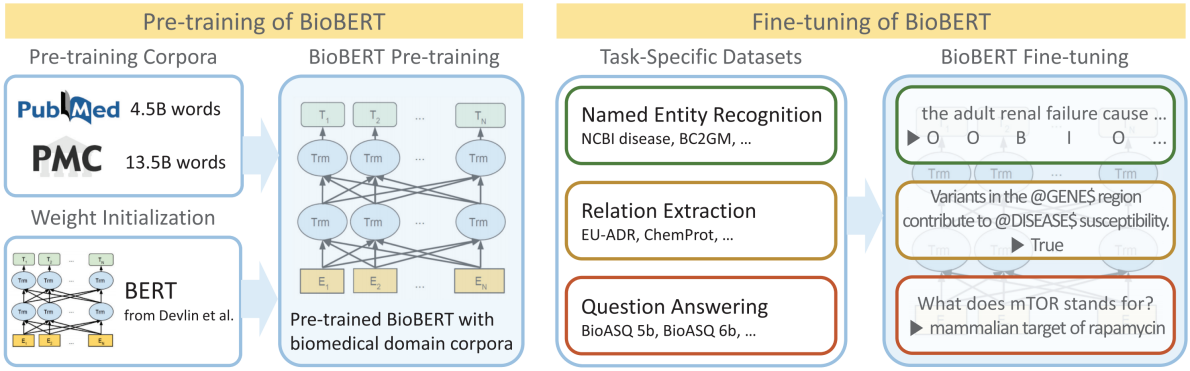

Figure 3.1: Overview of the pre-training and fine-tuning of BioBERT (Taken from [34])

The biomedical language model was assessed with an array of 14 task-specific datasets compatible with NER, relation extraction (RE) and QA. To sum up, BioBERT improved the state-of-the-art in 4 out of 8 datasets for NER and 1 out of 3 for RE, always considering F1 score as a metric. For QA, measuring the Mean Reciprocal Rank (MRR), BioBERT exceeded the state-of-the-art in all 3 datasets. BioBERT notably attains higher F1 scores in biomedical NER (0.62) and biomedical RE (2.80), and a higher MRR score (12.24) in biomedical QA.

Lee and Yoon opted for not creating a customized vocabulary, but instead relied on the WordPiece tokenizer to split OOV words into known sub-words or tokens. The main rationale behind this decision is the compatibility and interchangeability between BERT$_{Base}$ and BioBERT models.

3.2.2 SciBERT

Opposed to BioBERT's domain adaptation atop of BERT$_{Base}$, Beltagy, Lo and Cohan [3] pre-trained from scratch a BERT model using exclusively data from the Semantic Scholar corpus. This corpus includes 1.14M biomedical and computer science papers equivalent to 3.1B tokens and the training process lasted for about a week using a single TPU v33 with 8 cores. The SciBERT model was then thoroughly tested against a collection of 12 task and their corresponding domain specific datasets. These tasks, tackled using embeddings from frozen layers and fine-tuning for posterior comparison, spawned from classification, different sequence labeling and dependency parsing.

SciBERT obtained positive results and achieved state-of-the-art performance for 8 out of 12 tasks. In table 3.1, we can see how SciBERT substantially outperforms BioBERT on two

datasets, one NER specific and another one RE specific. The rest of the results are comparable but Beltagy underlines that the training has been done with much less data. It is important to note that all highest results are obtained through the fine-tuning approach. Surprisingly, BERT$_{Base}$ without domain adaptation also managed to achieve for a couple of tasks a performance between the previous state-of-the-art and the new one set by SciBERT.

Table 3.1: Comparing SciBERT with the reported BioBERT results on biomedical datasets

Task	Dataset	BioBERT	SciBERT
NER	BC5CDR	88.85	90.01
	JNLPBA	77.59	77.28
	NCBI-disease	89.36	88.57
REL	ChemProt	76.68	83.64

The authors of SciBERT complemented the domain adaptation with the insertion of custom vocabulary they named *SciVocab* that matches the scientific domain. The author measured the difference in performance with and without custom vocabulary and reported an average increase of 0.60% on F1 score.

3.3 NLP research in the Legal Domain

AI is changing the way work is done in the legal profession. NLP helps attorneys better organize their knowledge by automatizing the processing of documents. It also makes information easier to find and can even assist lawyers to be more persuasive. The methods powering these tools are constantly improving, although research applied to the legal domain is a small fraction when compared to other domains such as the biomedical domain.

3.3.1 Classification of Legal Documents

Sulea et al. [46] published their work on classification of legal case documents in French using more traditional ML techniques. The researchers apply in particular an ensemble system based on Support Vector Machine (SVM) classifiers. The classification was performed on predicting the law area, the ruling, but also on estimating the date of the ruling. The authors report results of 98% average F1 score in predicting a case ruling, 96% F1 score for predicting the law area of a case, and 87.07% F1 score on estimating the date of a ruling.

Another comparative study of legal text classifiers has been led by Howe et el. [27]. This paper evaluates a wide range of classification methods including ULM-FiT, Glove and Latent Semantic Analysis (LSA), but it is the first study of this kind to also include the general BERT$_{L}arge$ and BERT$_{B}ase$. The study was done using different training data constraint scenarios, feeding first only 10%, then 50% and finally 100% of the available corpus to estimate the performance variation of the models. By doing so, Howe revealed that deep transfer learning approaches, mainly ULM-FiT and both BERT variants, outperform the other models under the scarcest data scenario (the performance being measured using F1 score). Whereas Glove performed the best with 50% and LSA stands out completely when the training data is made fully available.

However, the authors note that the recall of the BERT models was superior when the data had low and medium availability. Also, they underlined a critical aspect of BERT: these pretrained models were competing in disadvantage due to the fact that their architecture is limited to a maximum sequence length of 512 tokens. BERT can only process a selected part of each document, meaning that if crucial information would appear after the clipping, this wouldn't be taken into consideration for the classification fine-tuning.

3.3.2 NER, semantic matching and linking of Legal Documents

Recent research involving legal data has a major interest in NER believing that this would carry a positive impact in more sophisticated applications such as information retrieval or structured summarization. Examples of studies focusing in this NLP task are the one from Glaser [21] and the MIREL project team [69]. Apart from that, a few other articles offer their take on semantic matching and linking between legal documents, such as Landthaler and Glaser [31] and Schaffer [66].

In their research on providing relevant legal information based on an arbitrary user selected text, Landthaler and Glaser [31] obtained superior results using a word embedding approach compared to TFIDF. They extracted the Word2Vec embeddings from segments of rental contracts and legal comments, then used them to calculate sentence vectors and these vectors were then compared with cosine similarity to finally rank them. It is important to note that the above mentioned studies where Glaser — Research Assistant at the Technical University of Munich — contributed were primarily conducted in German language. The datasets were nonetheless manually created and relatively small.

Schaffer [66], on the other hand, compares a rule-based model, a logistic regression model and a deep neural network model to link documents. The deep learning approach yielded the

best performance while exhibiting an interesting setup. It was built with *Doc2Vec* [32] on a bi-directional LSTM, nevertheless, BERT features were used for the word embeddings.

3.4 Information Retrieval with BERT

As confirmed by the authors of the previously described research papers, NLP plays a very important role in enhancing related fields such as Information Retrieval (IR). The ranking task is an elemental problem that IR solves. This can be summarized as: given a query q and a collection D of documents that match the query, the system has to sort the documents and return the results ordered by relevancy. Whether a document is relevant or not has to be defined by a established criterion.

With the arrival of BERT, researchers in the field of IT tested the impact of contextual modeling and reported positive findings. Dai and Callan [13] compared a variety of models on Robust04 and ClueWeb09 datasets for search tasks and found that BERT outperformed traditional word embeddings like word2vec. Qiao et al. [51] studied the behaviors of BERT in ranking with the MS MARCO dataset evaluating a passage reranking task and then the TREC Web Track ad hoc tasks with ClueWeb documents. Their experiments were performed by implementing 4 different BERT ranking models aimed at finding the highest contextual similarity between query and document.

The first model, *BERT (Rep)*, extracts the representations of both query q and document d, uses the special hidden state token [CLS] of the last layer and directly applies cosine similarity to measure the distance of the representations of both q and d. This method proved to be very ineffective. The second model, *BERT (Last-Int)*, concatenates instead those [CLS] tokens and uses BERT to predict the pairwise probability of q and d. The third one, *BERT (Mult-Int)*, is a variation of the Last-Int model and it sums the [CLS] representations of each layer. The last model, *BERT (Term-Trans)*, is an elaborated architecture that uses translation matrices, mean-pooling and linear combination with an extra neural network.

The researchers found the best results with their second approach, concatenation of [CLS] tokens, BERT (Last-Int). This correlates with the precedent work of Nogueira on ranking with BERT [44].

4 FARM Framework

4.1 Introduction

Machine learning in general requires streamlined workflows to carry out research efficiently. This facet is even more relevant in the practice of deep learning, where the internals of deep neural networks obfuscate the process and render debugging cumbersome. Without structure, the process can be error prone, the consequent study would lack robustness and become hard to replicate. The large amount of steps such as data collection, data cleaning, data preprocessing, training, evaluating, monitoring, loading and saving increase in complexity when adapting to different tasks, models and datasets. To tackle these issues, the community develops and constantly improves a variety of tools to ease the processes. Popular NLP frameworks and libraries for machine learning and deep learning include TensorFlow, Pytorch, Pytorch-transformers [74], Gensim, spaCy, Scikit-learn, and the number keeps soaring.

With the surge of the transfer learning paradigm in NLP, the space for the development of tools to apply this technique opened up. For startup ventures, this becomes an opportunity to pioneer in the front of bringing research into industry. An open source contribution is the preferred way for innovative business to reach the community of NLP specialists, ML practitioners and enthusiasts.

Deepset decided therefore to seize the opportunity and create a framework with the goal of making transfer learning straightforward for NLP experts to adopt but also for computer scientists from different backgrounds to try out state-of-the-art NLP methods. The company conceived *FARM, Framework for Adapting Representation Models*. This tool is core to the experiments set up in this current research and provides a convenient way to streamline the fine-tuning of language models and for downstream tasks. The author of this thesis contributed mainly in the set up of the evaluation routines, the regression prediction head and the user interface for inference.

4.2 Components

The framework has been designed in a modular way with a clear separation of concerns. The provided components allow the user to easily run fine-tuning for downstream tasks, monitor the training and evaluate the resulting model at the end. The configuration for each module is made as explicit as possible and the building blocks abstract away most of the usual implementation overhead.

4.2.1 Data Handling

Proper data handling is essential to any deep learning environment. The framework supports reading input files and creates processing ready datasets that can be used for other downstream tasks including training, fine-tuning or running inference. The processor is the component responsible for interfacing the input. A broad set of processors is made available covering classification, regression, NER and SQuad among others. In case the user needs a custom processor, they can follow the pattern of the existing ones and create an efficient handler in a short time. Subsequently, the Data Silo manages the created dataset and feeds it into the Dataloader. This includes parallelization to accelerate the data processing. Once the data is read and formatted, the Datasilo forwards it to the pertinent module according to the specified task. The flow of data is visualized in the Figure 4.1.

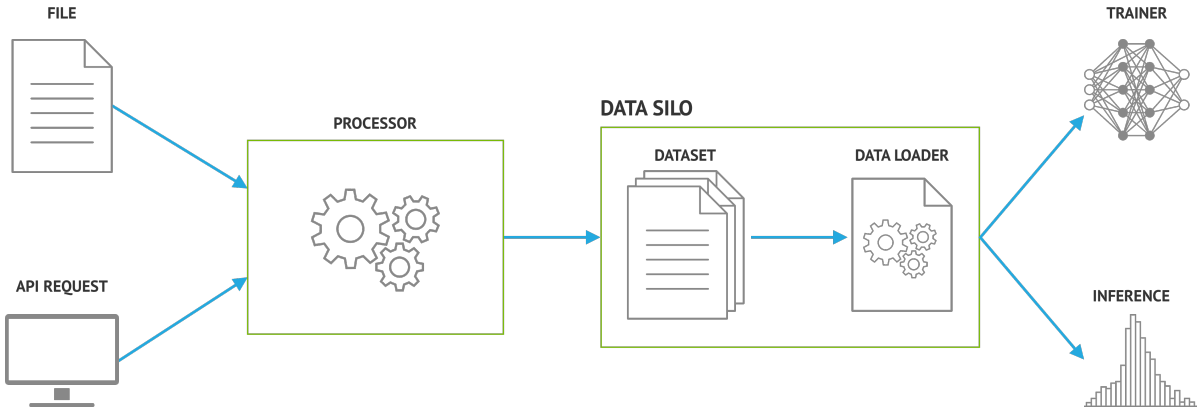

Figure 4.1: FARM Data Silo

4.2.2 Modeling

The FARM framework's modeling revolves around the concept of Adaptive Models as we can see in the Figure 4.2. This parent class is based on a pre-trained Language Model and

it appends one or multiple Prediction Heads for a specific downstream task. It conducts the forward passes, calculates the total loss of the neural network and back-propagates the result to readjust the weights of the whole Adaptive Model itself. With this building block approach, many combinations can be quickly implemented with minimum effort.

In a similar way to the Processor component, if a user needs a custom Prediction Head for a specific task, they can implement one following the existing pattern. If relevant enough, both new Processor and Prediction Head can be submitted through a pull request to the code repository and be integrated as a contribution for everyone else to use.

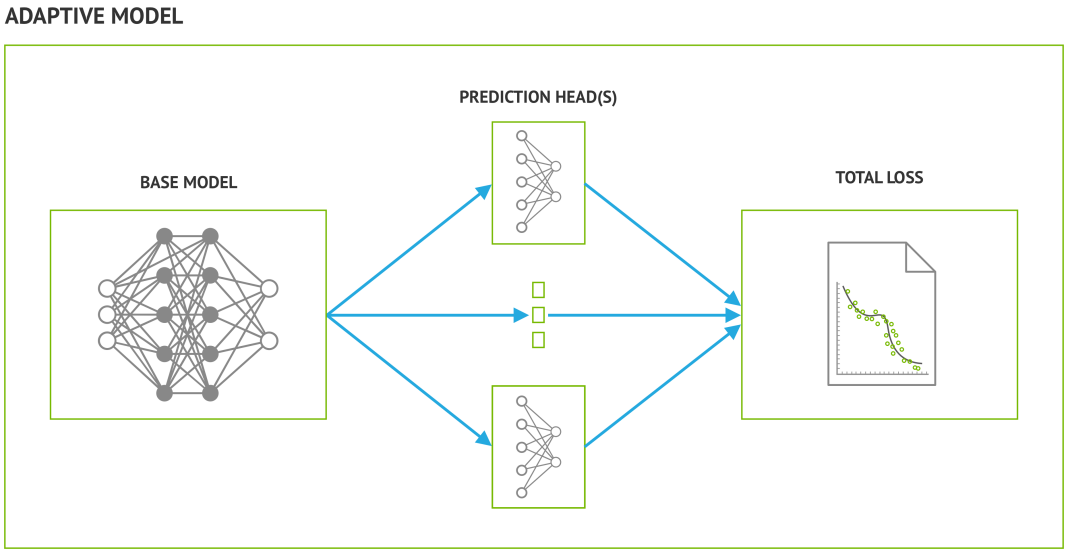

Figure 4.2: FARM Adaptive Model

4.2.3 Running and Tracking

To proceed with the training of a new model, the user needs to set up the data handlers and the Adaptive Model, configure the hyper-parameters and execute the training routine that's made available as a method of the Adaptive Model class. During the training, regular evaluations on the development set will be performed. At the end, a final assessment of the performance on the validation (or test) set will conclude the process and a new Adaptive Model object with fine-tuned weights will be returned.

From the start of the run until the final evaluation, the performance on the downstream task will be logged and there is also the option to track the training in real-time. The tracking relies on another well known open source platform named MLFlow [1]. The platform allows

[1] `https://mlflow.org/`

the collection of the details of the run as well as monitoring the evolution of the training. On top of that, the user interface includes the side-by-side comparison of multiple runs.

4.2.4 User Interface

Included in the tool set of FARM is the user interface for inference and testing. At the moment of this study, several BERT models are deployed, mainly English, Multilingual and German BERT. The available tasks that the user can try out include Question Answering (QA), Named Entity Recognition (NER) and document classification. The interface is currently hosted on DockherHub[2] as a ready to use container, but future development will open source the code. Users will be then able to deploy their own models, tasks, layout and inference visualizations. This feature has two purposes: the first one is to help NLP enthusiasts understand the capabilities of nowadays technology through a user friendly interface (see figure 4.3). Second, to make it easy for software engineers to showcase their own developments and models.

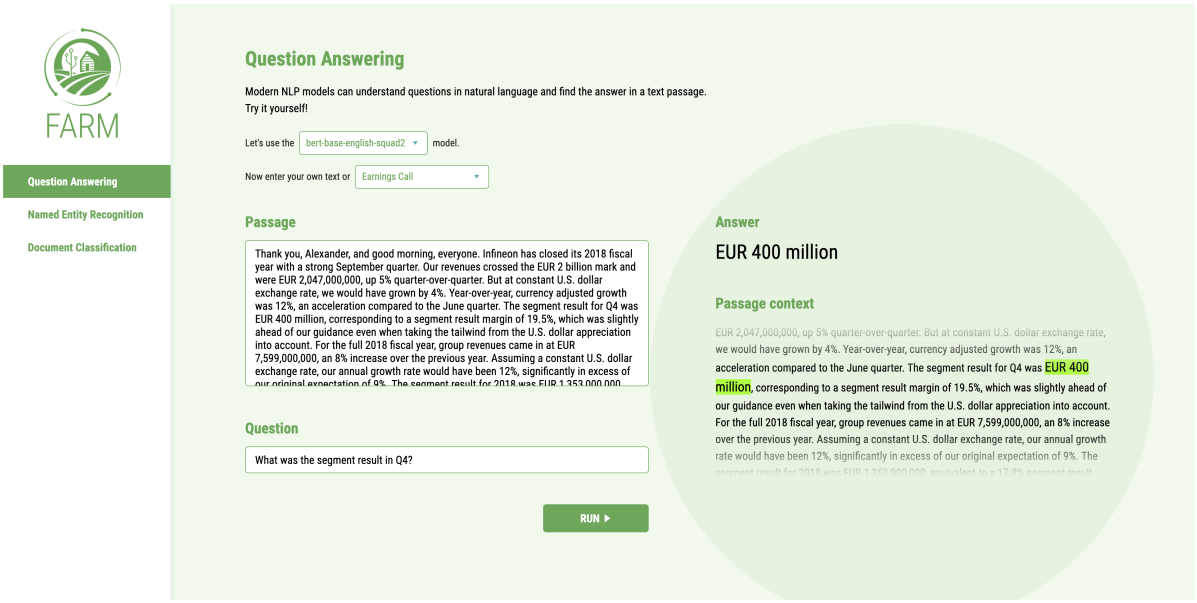

Figure 4.3: FARM Inference UI

4.2.5 German BERT

Finally, the FARM framework integrates a separate contribution of deepset which is the pre-trained German BERT model. The researchers behind the original BERT paper underline the advantage of single language pre-trained BERT models by estimating that they perform

[2] `https://hub.docker.com/`

about 3% better than the Multilingual BERT[3] on translation tasks using the XNLI dataset [12]. This German BERT model will serve as the base model for the future development of fine-tuning experiments in this research. Here we outline the principal facts about its pre-training approach and performance results. More details can be found in the post published by deepset[4].

Pre-training

To pre-train a single language model that can be compared to the BERT$_Base$ model, a corpus matching the size of the English data dump from Wikipedia (2,500M words) and BookCorpus (800M words) would be preferable. The 3,300M words used for the BERT$_Base$ pre-training totals about 16 GB of uncompressed text data.

For the pre-training of the German BERT, the dump from German Wikipedia (6 GB), news articles from different sources (3.6 GB) and the data from OpenLegalData (2.4 GB) has been utilized. This total of 12 GB of data is less than the amount used in the original BERT due to the lack of available quality text datasets. The data has been cleaned with custom scripts and sentences were extracted with spacy v2.1. The SentencePiece library has been used to create the WordPiece vocabulary and the suggested Tensorflow tools were applied to convert the text into an acceptable input for the pre-training of BERT.

The training of the monolingual model using the above-mentioned data was performed using Google's original Tensorflow code on a single Google Cloud TPU v2. Following the recommendations from Devlin et al [18], 810K steps were trained with batches of 1024 short sequences of 128 tokens length, then longer sequences of 512 tokens were trained through 30K steps. The BERT$_Base$ pre-training reported 1M steps but this is hardly comparable, as the hardware that has been used for BERT$_Base$ is much more optimized, mainly 4 Cloud TPUs in Pod configuration (16 TPU chips total).

Evaluation and results

The resulting pre-trained German BERT has been evaluated with 5 shared tasks. These encompass a sentiment classification task divided in two levels of granularity from germEval18 [73] (measured with macro F1 score), 2 NER tasks respectively from germEval14 [4] and CoNLL03 [62] (measured with F1 score for sequence labeling evaluation) and a document classification task 10kGNAD [63] (measured with accuracy). The results were compared with the Multilingual BERT. As shown the Table 4.1, the German BERT outperforms Multilingual in 4 out 5 tasks. The only task that Multilingual BERT performs better is CoNLL03. This shared task includes a dataset in English. By running the NER task on it using BERT$_Base$, the

[3] https://github.com/google-research/bert/blob/master/multilingual.md
[4] https://deepset.ai/german-bert

results were in the same range of Multilingual and German BERT confirming that the obtained metric for German is correct.

Table 4.1: Pre-trained BERT model multi-task performance comparison

BERT Models			
Shared Taks	Multilingual cased	Multilingual uncased	German cased **(ours)**
germEval18Fine	0.441	0.461	**0.488**
germEval18Coarse	0.710	0.731	**0.747**
germEval14	0.834	0.823	**0.840**
CoNLL03	**0.850**	0.844	0.848
10kGNAD	0.888	0.901	**0.905**

4.3 Open Sourcing

So, are the efforts of creating and maintaining a large-scale project worth the dedication? Studies show that the benefits tend to outbalance drawbacks when it comes to contributing in open source software [41]. A successful open source project brings the attention of industry and academy talents, reinforces the momentum for advancing the technique, promotes the group or institution behind the initiative and captures many adherents. The most valuable outcome resides without doubt in its collaborative aspect: the open source projects are resilient due to the engagement of the supporters who actively participate in the implementation process by either contributing, providing feedback, fixing bugs or raising issues.

Open sourcing is, nevertheless, challenging. The end goal is building a community around the open source project. Reaching out for the target segment of users requires capturing their attention and interest, but first, the authors need to find the proper channel to do so. A recent study showed that social media and blog posts are still important platforms to share the news about a new development [8].

In spite of the incubating state of FARM and the recentness of German BERT, several participants of the GermEval 2019 competition[5] used them for the shared task 2, *Identification of offensive language*. This task is divided into 3 classification sub-tasks. The first one is binary detection of offensive language, the second one is a fine-grained labeling using 3 subcategories and finally the third sub-task requires detecting if the offensive language was implicit or explicit.

[5] `https://2019.konvens.org/germeval`

Using the pre-trained German BERT model in combination with their own data pre-processing and fine-tuning, Paraschiv and Cercel (UPB) [48] obtained good results. This led them to the first position for both the shared task 2.1 and 2.2, binary and multi-label classification respectively. On the other hand, the winning team for the task 2.3, Risch, Stoll, Ziegele and Krestel (HPI) [55] solely used the FARM framework and German BERT.

4.4 Summary

FARM takes away most of the complexity by abstracting the common parts of modern deep learning approaches for NLP. Its focus on transfer learning makes it the ideal framework to work with pre-trained models. The relevance of usefulness of FARM has been validated by the positive results at the GermEval 2019. Finally, the growing support by the open source community will hopefully maintain the code repository at the pace of research and steer the future design decisions to the right port.

5 Methodology

The current chapter presents the considered data resource and how we use it to train a domain specific BERT model as well as creating an adapted vocabulary. We also clarify the fine-tuning methods and evaluation downstream tasks.

5.1 Introduction

In order to find the answers to the research questions, firstly, an adequate methodology has to be defined. This will be heavily inspired by the related work and build upon previous research. In fact, a similar approach to the domain adaptation that has been performed by BioBERT will be adopted. Nonetheless, the available tasks and datasets for the legal domain in German language are extremely different in nature. Therefore, we present in this section the chosen dataset and the considered tasks to evaluate the fine-tuned model.

5.2 Dataset

Data selection

Legal data openly available in German is scarce. There are a number of internet resources, mainly law portals, listing the rules of the German legal system. But few of them are aiming to provide the available data for processing instead of consultation. The two online resources that conform to the requirement of a minimally structured data that will allow data scientists and machine learner engineers to operate with relative ease without recurring to scraping are OpenLegalData and OpenJur. When compared, OpenLegalData is becoming an important player in the scene of legal data mining, offering resources such as Jupyter Notebooks with different analysis on their own data. In February 2019, they published a dump of their database for public use in NLP. OpenLegalData also offers an API to conveniently fetch documents. OpenJur has, on the other hand, more content, about 450.000 court decisions and verdicts compared to the approximately 104.500 cases from OpenLegalData. Unfortunately, at the

date of this research, exports of the database from OpenJur were not available nor the API was ready. Hence, the selected dataset for the research is the one from OpenLegalData.

Data pre-processing

The data provided by the database dump of OpenLegalData is a collection of JSON structured in the same way it would be obtained via API request. Each document is described by fields such as:

```
1  {
2    "id": 318559,
3    "slug": "bgh-2019-03-26-xi-zr-37218",
4    "court": {
5      "id": 4,
6      "name": "Bundesgerichtshof",
7      "slug": "bgh",
8      "city": null,
9      "state": 2,
10     "jurisdiction": null,
11     "level_of_appeal": "Bundesgericht"
12   },
13   "file_number": "XI ZR 372/18",
14   "date": "2019-03-26",
15   "created_date": "2019-04-13T10:00:11Z",
16   "updated_date": "2019-06-11T12:49:20Z",
17   "type": "Beschluss",
18   "ecli": "ECLI:DE:BGH:2019:260319BXIZR372.18.0",
19   "content": "<h2>Tenor</h2>\n\n<div>\n         <dl class=\"RspDL\">\n            <dt/>\n
                <dd>\n               <p>Die Beschwerde der Beklagten gegen die
       Nichtzulassung der Revision in dem Urteil des 6. Zivilsenats des Oberlandesgerichts
       Stuttgart vom 19. Juni 2018 wird zur&#252;ckgewiesen.</p>\n...   "
20 }
```

The court block (lines 4-12) is an important metadata that will serve as a label for classification. Then the text is included in the content field, although coded in HTML. The personal data is already anonymized by OpenLegalData. For the current study, the text of each document has been cleaned by removing all HTML enclosing tags and trimming the trailing spaces and tabs. The resulting data is a TSV file with 104.500 case laws preserving every field but with a ready to process cased corpus.

5.3 Tools and environment

Software

Deep Learning in both research and production has been dominated by the python programming language. This is due to its lean learning curve and eco-system of data science oriented tools. Between the two major open source machine learning framework, we choose PyTorch

(implemented by Facebook) over TensorFlow (by Google) because of its simpler approach and reduced learning overhead. It is worth mentioning that at the moment of this thesis, the latest Pytorch version was 1.2 and it didn't offer support for Google Cloud TPUs until 1.3, although it already included support of tools such as TensorBoard that was exclusive to TensorFlow. Using PyTorch as foundation, we also use FARM (chapter 4) as an abstraction to streamline all the experiments in this current study.

Hardware
All the deep learning computing that train faster on GPU was performed on Telekom Cloud GPU-accelerated instances with 8 vCPUs, 64 GB RAM and NVIDIA V100 with 1 core, more experiments were conducted on a server instance with the same settings on Google Cloud Platform. The remote machines ran Ubuntus 16.04.6 LTS Xenial Xerus as operating system, the local machine used MacOS 10.14 Mojave.

5.4 Language Model Fine-tuning

Following the principle of fine-tuning in transfer learning (section 2.4), we "pre-train" our own domain specific BERT model using the German BERT language model (subsection 4.2.5) as initialization in the same way as BioBERT (subsection 3.2.1) does. If we would pre-train from scratch like SciBERT, we could expect better results, but resource limitations made the expensive required hardware unattainable. Apart from that, it would have entailed a very long computation time, more than a week according to the authors of SciBERT.

The language model fine-tuning process further trains the model on the domain specific data making it faster as it should require less data. The training processes are the Next Sentence Prediction task and the Masked Language Modeling task (subsection 2.5.3), which are the two original self-supervised tasks for BERT pre-training. Using FARM, we set up the processor for the domain data and an adaptive model with prediction heads for the respective tasks. Then we fed a data sample containing 500MB of OpenLegalData documents. The LM fine-tuning lasted about 12 hours on a cloud machine equipped with 4 V100 GPUs and reported positive results included in Table 5.1

Table 5.1: German Legal BERT LM Fine-tuning results

Taks	Test Acc.	Test Loss	Validation Acc.	Validation Loss
Next Sentence Pred.	0.934	0.161	0.937	0.155
Masked LM	0.617	0.025	0.627	0.027

5.5 Domain vocabulary insertion

When fine-tuning a pre-trained language model on domain specific data, the resulting model has updated weights that represent better the characteristics and the vocabulary of the target domain (subsection 2.4.2). Before starting the process, it is required to specify a custom vocabulary file to incorporate into the base model. Therefore, we need to extract the vocabulary relevant for the legal domain. A sample of 1.000 documents has been selected from the OpenLegalData dataset and the *Byte Pair Encoding* algorithm (BPE) has been used to analyze the words and sub-words units [65]. The tokenizer that we utilized is the one from German BERT. This produced a collection of 145.380 words broken down in an array of tokens.

Once the collection of tokens was obtained, these tokens were compared against the segments of a legal dictionary obtained through dict.cc. The new tokens that did not belong yet to the original German BERT vocabulary were added, resulting in an increase from 27.000 to 29.910 tokens. There is a 90% overlap of the original vocabulary with the legal one.

With the insertion of our custom legal vocabulary, the tokenization provides different segmentation of words affecting the input sequence for BERT and we expect to slightly improve the performance in the same way SciBERT did by creating their own SciVocab (subsection 3.2.2).

5.6 Hyperparameters search

The hyperparameters can also be optimized to attain the highest performance [16][5]. These are the parameters that the model cannot learn by itself and need manual adjustment. Examples of hyperparameters include the learning rate, the number of training epochs, the train batch size, the warm-up period, the dropout rate and all the parameters that could be adjusted in the optimizer like the Adam optimization algorithm (section 2.1.2). Architectural configurations such as the number of hidden layers and nodes could also be considered hyperparameters, but we won't alter these factors since we will be evaluating our BERT against existing ones fine-tuned with the default values. For the downstream tasks we perform non exhaustive random search mostly on the learning rate.

5.7 Evaluation Tasks

The legal domain provides a rich context for applying NLP for a variety of goals. Inspired by Dale's review of NLP in the legal domain [14] and taking into account the idiosyncrasy of the data structure from the information provided by OpenLegalData, three different downstream tasks have been devised. The following tasks will serve to evaluate the different models in the legal context: a classification task, a regression task and a semantic similarity task.

5.7.1 Baselines

For the tasks mentioned above, we compare the BERT model with other well-researched and robust methods as baselines. This way, we can judge the impact of BERT and specifically our German Legal BERT on tasks that cover different complexities and goals. The chosen baseline models are the Bag-of-Words (BOW) complemented with term frequency–inverse document frequency (TFIDF) [56] and FastText [23].

Bag-of-words with TFIDF
The bag-of-words is a model similar to the N-gram language model that counts only occurrences of words and disregards the order, but considering $N = 1$. TFIDF recalculates the weight of the words taking the assumption that a term, that is rare throughout the whole corpus and appears frequently in a given sentence, is more important to the meaning of the sentence than words which are generally frequent in the corpus. From a semantic perspective and especially taking into account references and pronouns, this assumption is not always applicable. Taking for example "a judge" and "the judge", the first can refer to any person practicing this profession and the latter to a very specific and unique person. This model is very flexible and can be used in a large variety of tasks.

FastText
FastText is a library designed to address text representation and classification. It is based on word2vec, although it extends the model by treating the characters as the smallest unit and not words as word2vec and Glove would do. FastText uses a character level N-gram model. This makes it capable of constructing words that are OOV by computing word vectors using N-grams from 3 to 6 letters. It is able to produce word embeddings from a corpus that can be used for different downstream tasks but pre-trained word embeddings are made available and the German pre-trained model will be used for the evaluations.

5.7.2 Fine-tuning BERT for downstream task

Neural transfer learning (section 2.4) allows fine-tuning for a specific downstream task. In a similar way to the Language Model Fine-tuning (section 5.4), we use the pre-trained BERT models we aim to compare, copy the weights of each network, append an additional layer for each downstream task and fine-tune the new structures. We included each task specific output layer using the composition that FARM relies on, that is adding a prediction head to the base model forming an adaptive model (subsection 4.2.2). Figure 5.1 displays the process of fine-tuning by copying the model's weight and feeding annotated data for a downstream task.

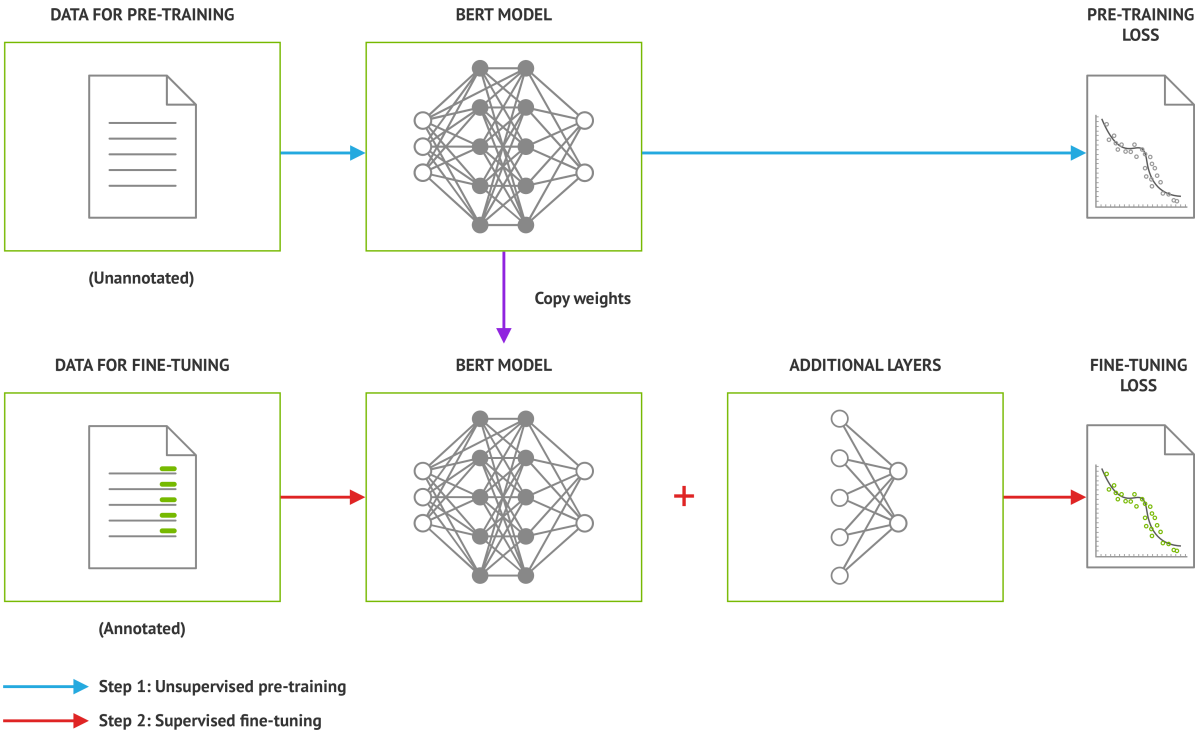

Figure 5.1: Fine-tuning process

5.7.3 Classification

We implemented several classifiers that can take a case law as input and classify it in the correct court and level of appeal. The application of laws and the trials corresponds to specific courts depending on the nature of the case, this segmentation allows the classification. This task is analog to the one researched by Sulea et al (subsection 3.3.1).

In order to implement TFIDF, we use the Scikit-Learn library (sklearn), mainly the TfidfVectorizer and the LogisticRegression methods. The relevant parameters are L2 normalization,

maximum features set to 10.000 and using the word analyzer. For FastText, we use the default classifier that comes included in the FastText library. The hyperparameters are as follow learning rate of 1.0, 5 epochs, using bi-gram, a bucket size of 200.000. The buckets is a concept the authors use to define the dimension of the model in addition to the vocabulary size. The remaining parameters are word vectors of size 50 and loss using hierarchical Softmax. The proportion of the test sample is 20% of the available examples.

For BERT models, we take Multilingual BERT and German BERT as baselines for deep neural network approaches. But the relevant evaluation is centered in the German Legal BERT Model. To fine-tune for this task, we use FARM, create an adaptive model with the classification prediction head. FARM's classifier for BERT deals with imbalanced classes by weighting them with CrossEntropy. The output layer that has been added uses a simple feed forward network and consequently applies a Softmax to obtain the prediction probabilities. The hyperparameters that we applied are: max. sequence length of 512 tokens, 5 epochs, batch size of 32, learning rate set at $2 \cdot 10^{-5}$ and 10% for the warm-up proportion.

5.7.4 Linear Regression

The linear regression task aims to predict the amount in dispute given a court decision. The court resolutions are highly variable in content, length, style and even structure. Hopefully they are all based on a predefined template and certain sections such as the case summary or the reasoning for the judge decision are preserved. One very common element in most litigations is the amount in dispute, where the plaintiff claims a certain amount of monetary value to compensate for the damages they suffered. This current downstream task could be used by the law practitioners to verify the adequateness of the monetary claim.

We use the same models as for the classifier, however, a degree of adaptations is required to perform regression. The linear regression with TFIDF has been implemented again with Tfidf-Vectorizer but this time using the sklearn LinearRegression function for computing the ordinary least square regression. The Standard Scaler, a preprocessing function provided by sklearn, was applied in order to remove the mean and scale to unit variance. We keep the same parameters as in classification, applying L2 regularization (Ridge Regression) yield better results. The FastText library doesn't offer linear regression, so we extract FastText sentence embeddings for the whole document and then feed them to the same LinearRegression function as for TFIDF but in this case, L1 regularization (Lasso Regression) was the best option.

We implemented a prediction head for regression in FARM in order to fine-tune the BERT models. It also features the Standard Scaler, it learns using the Mean Squared Error as cost

function (see Equation 2.3) and outputs continuous predictions. The same hyperparameters used for the classification performed equally well for regression, but we found that 8 epochs performed best.

5.7.5 Semantic Similarity

Searching documents by similarity has many possible applications in the legal profession. Recommendation systems and more efficient search engines would accelerate the legal research for valid arguments based on precedent cases. This would impact positively the efficiency of attorneys' daily work.

We calculate the similarity with TFIDF using the cosine similarity between document pairs and sorting the highest scoring ones. Cosine similarity calculates the cosine of the angle between the two document vector representations and the result is bound by [0,1], 1 indicating that the vectors are identical. The sklearn library comes again handy for this calculation. For FastText, we also apply cosine similarity but it is well known that computing the pairwise distance using this method directly on word embeddings yields poor results. We apply the Smooth Inverse Frequency (SIF) [1] on the FastText embeddings before feeding them to the cosine similarity.

The similarity using BERT models was implemented utilizing the Next Sentence prediction head. We fit the tokens of the two documents to be compared into the expected input format (subsection 2.5.4). The amount of tokens for each sequence is distributed evenly. The documents are then sorted by highest score, in other words the most similar ones, for posterior evaluation.

User evaluation

However, the lack of proper labeling in the legal data renders the automatic evaluation of the similarity task impossible. To be able to do so, a complete citation of related cases should be attached to each document. This is unfeasible, not even manually by a lawyer, and it would also require a much larger dataset of case laws as the existing citation network continuously expands.

To circumvent this issue, we opted to evaluate the performance of the different models on the similarity task by the means of a user evaluation. The experiment consists in emulating a search based on a document, where ten hypothetical results describing supposedly related case laws are presented to the user, and their task is to go through these documents and manually determine if they are relevant or not to the reference document. The criterion for relevancy

here is the similarity. Yet, the degree of required resemblance is arguable and the evaluation should be ideally taken by a professional of the legal area.

We originally considered presenting 5 different case laws to each user to evaluate. But the amount of time required per case law diverged from an average of 7 minutes for lawyers and 15 minutes for non-lawyers. 5 case laws is an unfeasible evaluation due to its excessive length. Therefore, a handpicked case where the results were not so trivial was chosen. This case[1] describes a claim against the tax office of Rheinland-Pfalz. The plaintiffs are a couple. The wife suffers a car accident on the journey between home and workplace and the legal dispute is whether the health treatment costs derived from the accident can be deducted as income-related expenses from her income. The result is that the legal action is dismissed and the claimers are ordered to pay the costs. This case is fairly complex and specific as it involves a public institution, a car accident, treatment costs and taxes for a married couple. There are many cases involving taxes related to the transportation to work, some concerning an accident as well, but not dealing with the tax issue from the costs of the hospital care.

To prepare for this test, we produced the similarity matrix of the selected case with all the different cases in the dataset using all the different models (Table 5.2). We serve the top 10 returned documents to the user to evaluate.

Table 5.2: Table of top ranked similar documents per model, document ids are shown with the similarity score

Rank	BOW + TFIDF + Cosine Similarity	FastText + SIF + Cosine Similarity	German BERT	German Legal BERT
1	119538, 0.6353414	137553, 0.9998200	133656, 0.9924986	133656, 0.9996964
2	133656, 0.6253577	125238, 0.9998052	137163, 0.9648799	89404, 0.9990593
3	140999, 0.5130107	91421, 0.9998046	89404, 0.6492820	90209, 0.9989531
4	132758, 0.5063738	136275, 0.9997901	107185, 0.4611582	140999, 0.9982892
5	167974, 0.5022549	100185, 0.9997861	105995, 0.3971383	162168, 0.9980289
6	66324, 0.4969475	80525, 0.9997857	115690, 0.3920132	104356, 0.9976148
7	66320, 0.4777645	162536, 0.9997842	110306, 0.3693900	133607, 0.9974799
8	112683, 0.4774234	108948, 0.9997798	91923, 0.3505391	92997, 0.9974213
9	137163, 0.4582637	137708, 0.9997791	71363, 0.3448213	98772, 0.9967712
10	135572, 0.4533617	103588, 0.9997768	98886, 0.2499939	105612, 0.9962702

We provide participants with an UI following the recommendations for information retrieval systems with users outlined by Kelly [28]. Aware of the duration and the cognitively de-

[1] https://de.openlegaldata.io/case/fg-rheinland-pfalz-2016-02-23-1-k-207815

manding task that is finding out which legal documents are the most similar ones, the user experience is an aspect that received a lot of attention and the design of the interface has been improved through several design iterations and extensive user testing.

The UI arrangement places the reference case in the left column, the user can browse through ten hypothetical search results in the right column and the content of the selected result would be loaded in the middle column for an easier side by side comparison. A retrieved document can be mark as relevant or not using action buttons placed on the top and on the result preview. The considered case law in the central column has its evaluation state represented using background color (light green for relevant and light red for unrelevant) and it's linked to the preview thumbnail on the right column for easier recognition and action feedback as we can see in Figure 5.2. This evaluation is preceded by a introductory page and the layout and colors are chosen to evoke a professional feel according to the original desired user target.

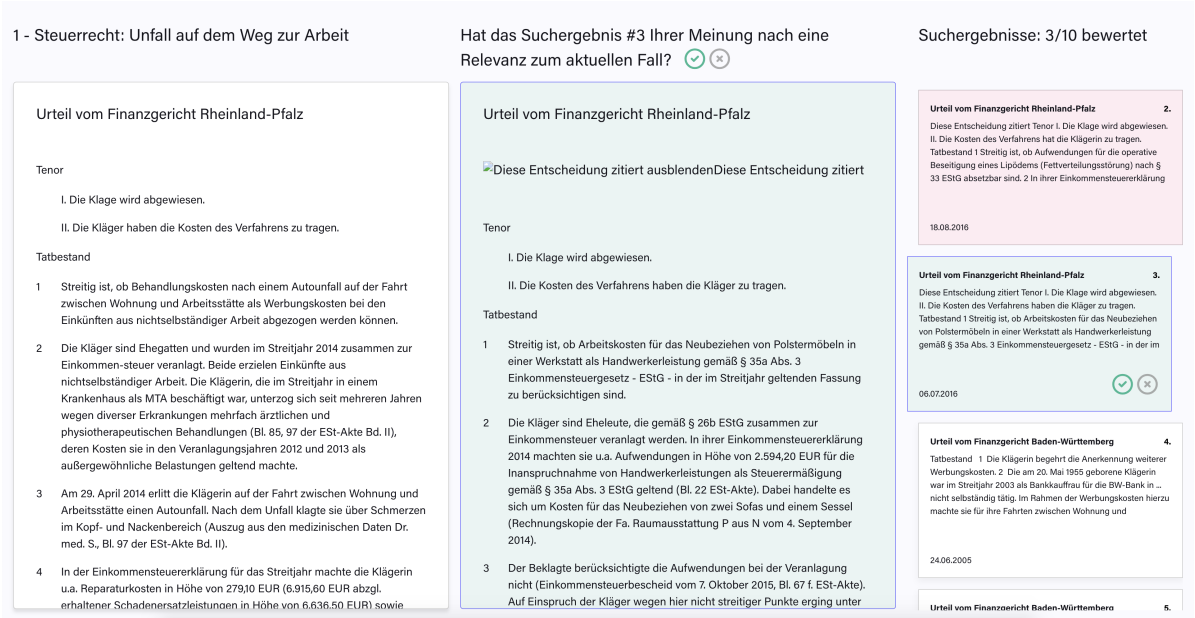

Figure 5.2: User evaluation UI

As a within-subject experiment, the reference case law remains the same one for all the participants, but the language model retrieving the pre-computed results is different for every new user, updating the ranking of similar case laws.

6 Experiments

6.1 Introduction

In the current chapter, the experimental setup will be briefly outlined. We give an in depth account of the obtained results and proceed to analyze them per downstream task. Interpretation and discussion of the experimental results and the factors leading to them will be provided.

6.2 Experimental Setup

6.2.1 Data

We adapt the OpenLegalData dataset section 5.2 for the classification, regression and similarity tasks. For the classification, we preserve the labels about "Jurisdiction" and "Level of Appeal" and the cleaned text resulting from pre-processing the corpus. Cases with no labels were discarded, in the end we obtain 54.654 cases with "Level of Appeal" and 69.298 containing the "Jurisdiction" label.

For the regression task, we discarded the cases where no dispute amount was involved. The relevant documents were reduced to 12.481 case laws. Then, we isolated the numerical value describing the compensation and removed the sentence containing this amount in order to hide it from the model during training.

6.2.2 Metrics

In this section we will give an overview of the different metrics that are used to measure the quality fit of the models and the different downstream tasks. Most of them are common from the field of Information Retrieval as it correlates with the matching quality of the returned documents with the original query.

F1 score

The F1 score considers both the precision P and the recall R of the test to compute the score. The formula is given by Equation 6.1, where TP are true positives, FP are false positives and FN are false negatives. The F1 score is the harmonic mean of the precision and recall, when perfect precision and recall are reached, F1 returns 1.

$$F_1 = \frac{2PR}{P+R} = \frac{2TP}{2TP+FP+FN} \tag{6.1}$$

A macro-average F1 score computes the metric independently for each class and then take the average, hence treating all classes equally:

$$F_1^M = \sum_{i=1}^{n} w_i F_{1i} \tag{6.2}$$

A micro-average F1 score aggregates the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable to mitigate class imbalance:

$$F_1^\mu = \frac{2\sum_{i=1}^{n} TP_i}{2\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i + \sum_{i=1}^{n} FN_i} \tag{6.3}$$

R-Squared, Coefficient of Determination

The coefficient of determination, denoted R^2, provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes predicted by the model.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} \tag{6.4}$$

Equation 6.4 describes R^2 where SS_{res} is the residual sum of squares and SS_{tot} is the total sum of squares.

6.3 Classification

The first task refers to the classification of case laws according to their Jurisdiction and Level of Appeal. Similar to the GermEval tasks for offensive language detection, there are two different classification levels and the distribution of classes is unbalanced as we can see in Figure 6.1 and Figure 6.2. We measure the performance of the classifiers with the weighted-averaged F1 score, averaging the support-weighted mean per label. The results are shown

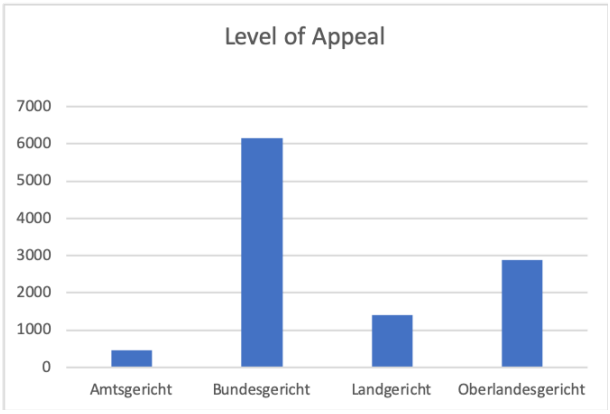

Figure 6.1: Distribution of Level of Appeal labels

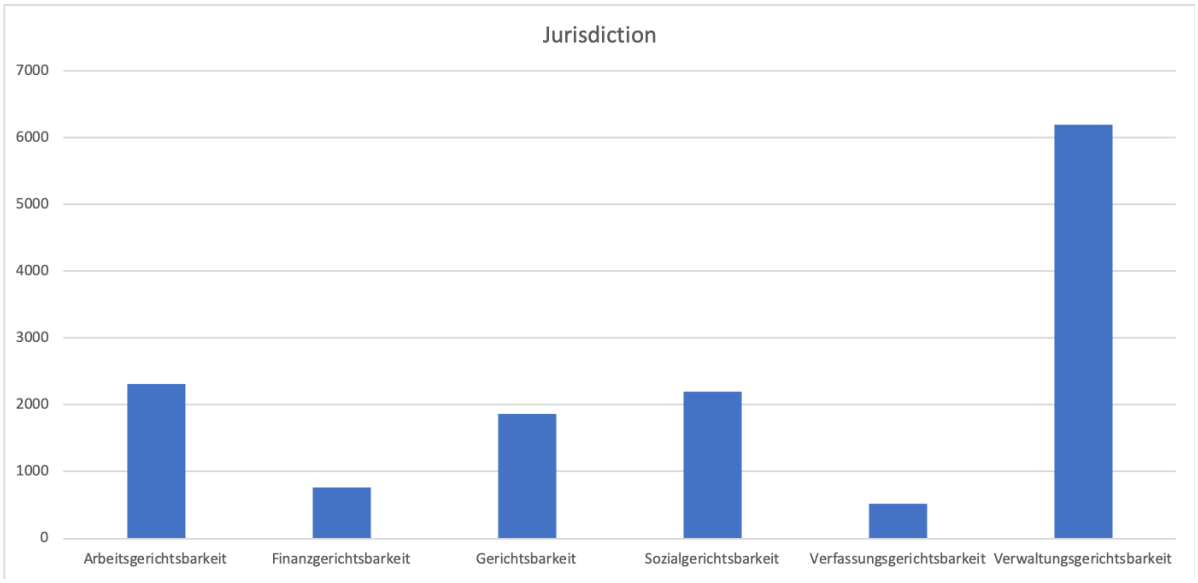

Figure 6.2: Distribution of Jurisdiction labels

on Table 6.1. The classification was highly effective across all the different models. The Jurisdiction that describes 6 different major courts, and therefore labels, was almost forecasted perfectly. We believe that the nature and diversity of topics covered per jurisdiction -for example constitutional, tax, social- permitted the classifiers cluster the case laws in well defined areas of the vector space. The Level of Appeal is also very performant but not reaching the accuracy levels of the Jurisdiction labels. The slightly lower F1 score is mostly due to the low recall on one label, the Amtsgericht. In the test set, only 471 documents are tagged with this court, resulting in a lower recall of 0.65 in the case of TFIDF. When compared to Bundesgericht, this area includes 6.161 documents and yields a recall of 0.99.

Table 6.1: Results for classification task

	Jurisdiction	Level of appeal
TFIDF	0.99	0.95
FastText	0.99	0.97
Multilingual BERT	0.99	0.96
German BERT	0.99	0.97
German Legal BERT	0.99	0.96

Using the same hyperparameters for all the BERT models, we notice that German Legal BERT did not outperform German BERT but yields close results. However, we would have expected a bigger gap between the performance of Multilingual BERT and German BERT too, but the results are comparable. This leads us to believe that the BERT models are reaching their full potential for the classification task on this dataset, improving over TFIDF and matching FastText.

6.4 Linear Regression

The linear regression task aims to predict the amount in dispute of a case. This amount is estimated by the judge and it could vary considerably due to smaller details. We expect a loose correlation with the semantics of the corpus and that it can be better reflected with models that capture the context better.

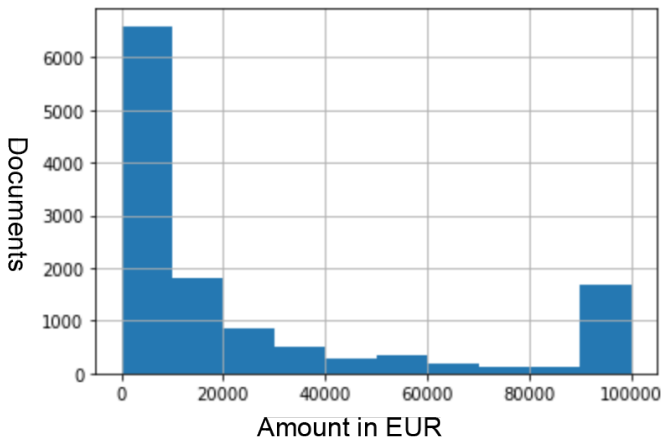

Figure 6.3: Distribution of compensation values

Upon first exploration of the data, we noticed that the distribution of the compensation amounts follows the slope of a multiplicative inverse function ($\frac{1}{x}$) indicating that most cases were smaller than 10.000 EUR in disputed amount. However, we applied a ceiling to values greater than 100.000 EUR to define a manageable range of predictions and this produced an accumulation towards that end as we can see in Figure 6.3.

Table 6.2: Results for regression task

	R^2
TFIDF	0.47
FastText	0.35
Multilingual BERT	0.24
German BERT	0.37
German Legal BERT	0.38

In Table 6.2 we can observe the evaluation results of the linear regression task using the TFIDF, the FastText embeddings and the fine-tuned BERT models. The closer the R-squared is to 1, the better. In the current evaluation the best model corresponds to TFIDF which predicts better the amount in dispute, the coefficient of determination is relatively low and doesn't indicate a strong correlation, however the Figure 6.4, which depicts the linear regression prediction and the observations, subtly hints in favor of an existing correlation but as expressed by the coefficient of determination, it is not significant.

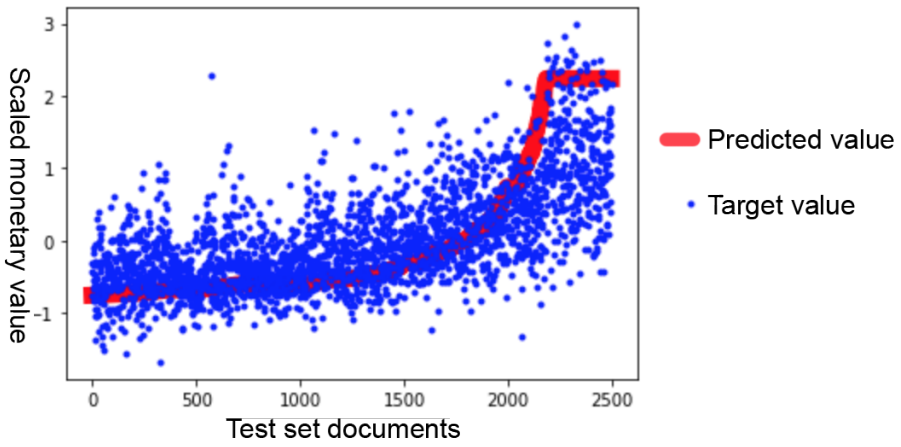

Figure 6.4: Plot of linear regression on monetary values using the test set.

The mediocre performance of the BERT models can be explained by the restriction on the amount of data they can process, When traditional ML models like TFIDF and FastText can be fed with entire documents independently of their length, BERT can only take a maximum sequence length of 512. This is an issue already mentioned in many related literature, pointing out that BERT competes in disadvantage of amount of processed data. The German BERT model shows only a marginal improvements over FastText for this regression task, the German Legal BERT presents a very small improvement, but they both still perform better than Multilingual BERT which is the model that scores the lowest in this case.

6.5 Similarity

Before deploying the user evaluation for the similarity task, we manually performed a sanity check and verify that the given results contained relevant ones. To our surprise, the Multilingual BERT scored high similarity for all the documents and wasn't capturing the case's specifics. The top similar cases returned were all unrelated. We tested Multilingual BERT with other corpora from Wikipedia and found out that it is capable of distinguishing general context such as history, politics, science, but not so well between subcategories. This is most likely due to the sampling of German data used to train the model that is also compatible with other 103 languages. We decided therefore to exclude Multilingual BERT from the Similarity task.

The similarity is measured by the Mean Average Precision (mAP) calculated from the answers given by the participants of the evaluation. The formula for the Average Precision of an individual evaluation response is:

$$AP@10 = \frac{1}{10} \sum_{i=1}^{10} \frac{TP_{Count}}{i} \tag{6.5}$$

where TP_{count} is the number of true positives at the position i. With all the collected answers N, we can average them as:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i@10 \tag{6.6}$$

We sought out for participants with the ideal profile in mind. Considering the technicalities of the field, the subjects should be native German speakers, understand legal language and have experience in legal research. This profile corresponds to the segment of law practitioners which includes attorneys and judges. However, engaging professionals from this field turned

out to be more challenging than expected, so the conditions were extended to German law students and German native speakers with an interest in law. By the end of the evaluation, we collected 16 answers, 4 per model. The results are turned into mAP per model is listed on the table 6.3:

Table 6.3: Results for similarity task

	mAP
TFIDF	0.2141
FastText	0
German BERT	0.2025
German Legal BERT	0.2565

Checking manually the set of returned documents, we noticed that it contains on average 2 case laws that are easily identifiable as similar. TFIDF and German BERT are clearly indicating this tendency. In certain cases the participants consider that another document ranked lower is relevant to the case but there is a general consensus that the top 2 retrieved documents are very similar. Nonetheless, when observing Table 5.2, we see that the TFIDF, German BERT and German Legal BERT don't share all the same top ranked documents, but the legal version still yields a higher mAP. This indicates that other returned documents were found similar enough to be relevant to the case. It is interesting to note that the decisions are more variable. This is most likely due to the fact that some aspects of the reference case were shared in the retrieved case but not all of them, making the user hesitant.

The pilot tests with different cases as reference showed that FastText with SIF was able to retrieved relevant documents, however for this specific case, it was not capable of returning the two documents that are definitely similar nor any other cases the users would have considered relevant.

This similarity experiment shows that the fine-tuning on the legal domain had an effect on the Next Sentence prediction head which is the fine-tuning setup we use to produce similarity scores. The results are however not significant, and one can discuss about the usefulness of retrieving documents that cover critical information partially.

7 Conclusion and future work

7.1 Review

We presented an example of transfer learning of the recent BERT model to the legal domain. Based on related literature, we proceeded with an approach that doesn't require the pre-training of the whole model from scratch, but instead initializes the weights of the model with an existing one, German BERT in our case. It is a convenient approach for several reasons: First, when the volume of available text corpora is not large enough to match the specifications for pre-training an entire new model. Second, the computational time, still considerably long and dependent on the amount of additional training data, is much shorter than the traditional pre-training that would expand over a week.

As suggested by Sebastian Ruder [60], the adaptation requires the source domain and the target domain to share a common base. In our case, it is the understanding of the German language. And following this principle, the adaptation was performed using the German BERT instead of the Multilingual one, as the evaluations during the pre-training of the German model obtained results that were superior to the Multilingual.

The creation of a custom vocabulary should help German Legal BERT model better the language by weighting the additional embeddings more accurately. Nonetheless, the original vocabulary already includes 90% of the legal vocabulary. This is one of the characteristics that renders the legal domain especially hard to grasp for both humans and machines, many of the day-to-day words are imbued with a different meaning when used in the legal context. SciBERT, the pre-trained BERT model for scientific texts [3], improves around 0.60% on F1 across all the datasets but their original-scientific vocabulary overlap is just 47%. In our case the effects of vocabulary use are rather diluted, specially when we don't have a battery of evaluations to firmly discern the improvement.

Fine-tuning for the classification task gave positive results for any BERT_{Base} models, although the language fine-tuning for the legal domain didn't yield improvements over the base model German BERT. For regression, all the BERT models are performing below the TFIDF and

FastText baselines, and no major differences were present between them. However, when it comes to the similarity task, the are noticeable contrasts. Multilingual BERT wasn't able to grasp the more fine grained differences between legal documents, as this model considers them all similar because they belong to the same domain, whereas German and German Legal BERT models are able to capture the similarities on the different legal topics. German Legal BERT seems to be able to provide documents that are more related to the reference document than German BERT.

7.2 Discussion

The German BERT model has been trained using a corpus from OpenLegalData among others. We believe that the results of fine-tuning the language model would have been much more significant if the model would have been pre-trained on more general information just like BERT_{Base} did with Wikipedia and Book Corpus. We think that German Legal BERT would improve even more in the semantic similarity task, but not necessarily for classification and regression, as fine-tuning for the tasks did yield similar results for Multilingual, German and German Legal BERT. We can therefore recommend the use of TFIDF and Word embeddings methods for common tasks such as classification especially when the task-specific data is composed by long text documents.

One of the biggest weaknesses of BERT is the limitation in the accepted sequence length. 512 tokens is sufficient for many specific tasks – the GermEval shared tasks for instance – but it is very far from enough in order to approach problems from the Information Retrieval field effectively. For this kind of tasks, the model isn't an out-of-the-box solution but rather a building component capable of tackling auxiliary tasks for later assembling more complex systems. Solving properly the length constraint with other methods is crucial in order to fully exploit the potential of this model.

The similarity task could have benefited from a larger sample of participants and mainly from the participation of more legal professionals. They could discriminate the relevancy of the documents much better than just native German speakers. Depending on the user base taking the evaluation, the criterion of the ranking would subtly shift. Attorneys would be able to deem if a case is relevant or not beyond just the semantic or topic similarity. Details that legally untrained users would have omitted would certainly be considered by lawyers.

The fine-tuning process saves considerable amounts of time and yields acceptable results. In the comparative BioBERT (language model fine-tuning) and SciBERT (language model pre-

training), the latter obtains better results but also at the cost of requiring more training data and training for over a week on very expensive machines. If models keep growing, this would lead to a not democratic situation for research, where only more favorable position institutions will be able to perform experiments. Other downside that the research community showed concerns about is the impact of the increasing interest in deep learning and the practice of it on other social aspects such as sustainability. The carbon footprint of these computations is surprisingly high and some researchers like Schwartz [64] already suggest new metrics to tie together energy efficiency and performance of the model. We believe that if this measure would be applied to the comparison between BioBERT and SciBERT, BioBERT would emerge victorious. In our case, the moderate improvements that the German Legal BERT provided would probably not justify the pre-training of the model when taking the efficiency considerations that Schwartz proposes. However, it will very likely produce much better results when adapting to a target domain that is further from the base domain. Transfer learning in NLP is still at its early stage and will very likely become a key driver in the industry and in research.

The low resourcing of German legal data and task-specific dataset makes the research in this area difficult. In contrast, AI applied to the legal domain is growing exponentially in China[1], as the ethics guidelines and data privacy regulations are much more permissive. A clear example is the amount of publicly available criminal cases on one single governmental portal of Wenshu[2] that sums up to almost 80 million documents. While still adhering to the strong data protection laws of German, more efforts should be invested in the collection of quality legal corpora as well as proper labeling for progress to be made in Germany.

7.3 Future work

As discussed above, data scarcity is a major obstacle. If there would be more available data, we could pre-train from scratch German Legal BERT model to compare with the current one. But as for now the other possible improvement using language fine-tuning would be training further with the remaining data of OpenLegalData and evaluating again. Alternatively, the legal language utilizes many terms and expressions from current language to refer to notions proper to the legal domain, being able to remove the ambiguity of these nuances would eventually lead to an performance enhancement of the model. Extending the contextual word

[1] http://cail.cipsc.org.cn/
[2] http://wenshu.court.gov.cn/

representations with specific world knowledge about law and legal language would definitely be an interesting follow-up research.

Studying ways to solve the issue of the maximum sequence length so the BERT model can produce document embeddings is a great challenge but also could produce interesting findings. The document embedding problem has been approached by Mikolov with an extension of Word2Vec called *Doc2Vec* [32], but there is still much room for improvement. The contextualized word embeddings from BERT would be an interesting approach to produce better document embeddings.

The author of this work believes that in domains such as legal, the only way to attract the professionals to participate in an experiment is in case the experiment itself provides them direct value. Therefore, it is believed that the similarity task, would have a better participation rate from attorneys if it would be a useful tool offered to them. This means, a full system with a search engine where they can search for documents for the current cases they are dealing with, would collect much more information about the performance of the model powering the system. This would be the ideal experiment, but comes also with slight ethical issues. Once the evaluation has been done, the most common thing to do is to stop the system to avoid maintenance and infrastructure costs. If the tool results useful to the attorneys and they integrate it to their daily workflow, it would be unethical to deprive them from this tool. Alternatively, research on the effectiveness of NLP and IR methods could be viable on open knowledge platforms such as OpenLegalData. Many indicators of document relevancy could be anonymously measured through user behaviors and the tool would remain available to the community as long as the project exists.

Bibliography

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. “A Simple but Tough-to-Beat Baseline for Sentence Embeddings”. In: (2017).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473 (2014).

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. “SciBERT: Pretrained Language Model for Scientific Text”. In: *EMNLP*. 2019. eprint: `arXiv:1903.10676`.

[4] Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Pado. “Germeval 2014 named entity recognition shared task: companion paper”. In: (2014).

[5] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. “Algorithms for hyper-parameter optimization”. In: *Advances in neural information processing systems*. 2011, pp. 2546–2554.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[7] John Blitzer, Mark Dredze, and Fernando Pereira. “Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification”. In: *Proceedings of the 45th annual meeting of the Association of Computational Linguistics*. 2007, pp. 440–447.

[8] Hudson Silva Borges and Marco Tulio Valente. “How do developers promote open source projects?” In: *Computer* 52.8 (2019), pp. 27–33.

[9] Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. “How to train good word embeddings for biomedical NLP”. In: *Proceedings of the 15th workshop on biomedical natural language processing*. 2016, pp. 166–174.

[10] Noam Chomsky and David W Lightfoot. *Syntactic structures*. Walter de Gruyter, 2002.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *CoRR* abs/1511.07289 (2015).

[12] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. “XNLI: Evaluating Cross-lingual Sentence Representations”. In: *Proceedings of the 2018 Conference on Empirical Methods*

in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, 2018.

[13] Zhuyun Dai and Jamie Callan. "Deeper Text Understanding for IR with Contextual Neural Language Modeling". In: *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: ACM, 2019, pp. 985–988. ISBN: 978-1-4503-6172-9. DOI: 10.1145/3331184.3331303. URL: http://doi.acm.org/10.1145/3331184.3331303.

[14] Robert Dale. "Law and Word Order: NLP in Legal Tech." In: *Natural Language Engineering* 25.1 (2019), pp. 211–217. URL: http://dblp.uni-trier.de/db/journals/nle/nle25.html#Dale19.

[15] Hal Daumé III and Jagadeesh Jagarlamudi. "Domain adaptation for machine translation by mining unseen words". In: *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies: short papers-volume 2*. Association for Computational Linguistics. 2011, pp. 407–412.

[16] Li Deng, Geoffrey Hinton, and Brian Kingsbury. "New types of deep neural network learning for speech recognition and related applications: An overview". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 8599–8603.

[17] Li Deng, Dong Yu, et al. "Deep learning: methods and applications". In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT*. 2019.

[19] Rosa L Figueroa, Qing Zeng-Treitler, Sergey Goryachev, and Eduardo P Wiechmann. "Tailoring vocabularies for NLP in sub-domains: a method to detect unused word sense". In: *AMIA Annual Symposium Proceedings*. Vol. 2009. American Medical Informatics Association. 2009, p. 188.

[20] N.G. Foster and S. Sule. *German Legal System & Laws*. Oxford University Press, 2002. ISBN: 9780199254835.

[21] Ingo Glaser, Bernhard Waltl, and Florian Matthes. "Named entity recognition, extraction, and linking in German legal contracts". In: *Internationales Rechtsinformatik Symposium*. 2018.

[22] Joshua T Goodman. "A bit of progress in language modeling". In: *Computer Speech & Language* 15.4 (2001), pp. 403–434.

[23] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. "Learning Word Vectors for 157 Languages". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[24] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[25] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[26] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://www.aclweb.org/anthology/P18-1031.

[27] Jerrold Soh Tsin Howe, Lim How Khang, and Ian Ernst Chai. "Legal Area Classification: A Comparative Study of Text Classifiers on Singapore Supreme Court Judgments". In: *arXiv preprint arXiv:1904.06470* (2019).

[28] Diane Kelly. "Methods for Evaluating Interactive Information Retrieval Systems with Users". In: *Found. Trends Inf. Retr.* 3.1–2 (Jan. 2009), pp. 1–224. ISSN: 1554-0669. DOI: 10.1561/1500000012. URL: http://dx.doi.org/10.1561/1500000012.

[29] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[30] Wouter M. Kouw. "An introduction to domain adaptation and transfer learning". In: *ArXiv* abs/1812.11806 (2018).

[31] Jörg Landthaler, Elena Scepankova, Ingo Glaser, Hans Lecker, and Florian Matthes. "Semantic text matching of contract clauses and legal comments in tenancy law". In: *Tagunsband IRIS: Internationales Rechtsinformatik Symposium*. 2018.

[32] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. 2014, pp. 1188–1196.

[33] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), p. 436.

[34] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* (Sept. 2019). ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. URL: https://doi.org/10.1093/bioinformatics/btz682.

[35] Qi Li. "Literature survey: domain adaptation algorithms for natural language processing". In: *Department of Computer Science The Graduate Center, The City University of New York* (2012), pp. 8–10.

[36] Li Jun and T. Duckett. "Some practical aspects on incremental training of RBF network for robot behavior learning". In: *2008 7th World Congress on Intelligent Control and Automation*. 2008, pp. 2001–2006. DOI: 10.1109/WCICA.2008.4593231.

[37] Zachary Chase Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). arXiv: 1506.00019. URL: http://arxiv.org/abs/1506.00019.

[38] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *EMNLP*. 2015.

[39] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 746–751.

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

[41] Frank Nagle. "Learning by contributing: Gaining competitive advantage through contribution to crowdsourced public goods". In: *Organization Science* 29.4 (2018), pp. 569–587.

[42] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[43] Michael A Nielsen. *Neural networks and deep learning*. Determination press San Francisco, CA, USA: 2015.

[44] Rodrigo Nogueira and Kyunghyun Cho. "Passage Re-ranking with BERT". In: *ArXiv* abs/1901.04085 (2019).

[45] Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. "Evaluation of Domain-specific Word Embeddings using Knowledge Resources". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. 2018.

[46] Octavia-Maria, Marcos Zampieri, Shervin Malmasi, Mihaela Vela, Liviu P. Dinu, and Josef van Genabith. "Exploring the Use of Text Classification in the Legal Domain". In: *Proceedings of 2nd Workshop on Automated Semantic Analysis of Information in Legal Texts (ASAIL)*. London, United Kingdom, 2017.

[47] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[48] Andrei Paraschiv and Dumitru-Clementin Cercel. "UPB at GermEval-2019 Task 2: BERT-Based Offensive Language Classification of German Tweets". In: *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*. Erlangen, Germany: German Society for Computational Linguistics & Language Technology, 2019, pp. 396–402.

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

[50] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://www.aclweb.org/anthology/N18-1202.

[51] Yifan Qiao, Chenyan Xiong, Zheng-Hao Liu, and Zhiyuan Liu. "Understanding the Behaviors of BERT in Ranking". In: *ArXiv* abs/1904.07531 (2019).

[52] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8 (2019).

[53] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. "Self-taught learning: transfer learning from unlabeled data". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 759–766.

[54] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". In: *Neural computation* 29.9 (2017), pp. 2352–2449.

[55] Julian Risch, Anke Stoll, Marc Ziegele, and Ralf Krestel. "hpiDEDIS at GermEval 2019: Offensive Language Identification using a German BERT model". In: *Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*. Erlangen, Germany: German Society for Computational Linguistics & Language Technology, 2019, pp. 403–408.

[56] Stephen Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF". In: *Journal of documentation* 60.5 (2004), pp. 503–520.

[57] Giorgio Roffo. “Ranking to learn and learning to rank: On the role of ranking in pattern recognition applications”. In: *arXiv preprint arXiv:1706.05933* (2017).

[58] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[59] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[60] Sebastian Ruder. “Neural Transfer Learning for Natural Language Processing”. PhD thesis. National University of Ireland, Galway, 2019.

[61] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM J. Res. Dev.* 3.3 (July 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: http://dx.doi.org/10.1147/rd.33.0210.

[62] Erik F. Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *ArXiv* cs.CL/0306050 (2003).

[63] Dietmar Schabus, Marcin Skowron, and Martin Trapp. “One Million Posts: A Data Set of German Online Discussions”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Tokyo, Japan, Aug. 2017, pp. 1241–1244. DOI: 10.1145/3077136.3080711.

[64] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. “Green ai”. In: *arXiv preprint arXiv:1907.10597* (2019).

[65] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162.

[66] Robert Shaffer and Stephen Mayhew. “Legal Linking: Citation Resolution and Suggestion in Constitutional Law”. In: *Proceedings of the Natural Legal Language Processing Workshop 2019*. 2019, pp. 39–44.

[67] Asa Cooper Stickland and Iain Murray. “BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning”. In: *ArXiv* abs/1902.02671 (2019).

[68] I Sutskever, O Vinyals, and QV Le. “Sequence to sequence learning with neural networks”. In: *Advances in NIPS* (2014).

[69] Milagro Teruel, Cristian Cardellino, Fernando Cardellino, Laura Alonso Alemany, and Serena Villata. “Legal text processing within the MIREL project”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[70] T. Tieleman and G Hinton. “Lecture 6.5 - RMSProp”. In: *COURSERA: Neural Networks for Machine Learning* Technical report (2012).

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[72] K Werle. *Wohin nur mit all den Anwaälten?* 2013. URL: https://www.spiegel.de/karriere/juristenschwemme-zu-viele-juristen-draengen-auf-den-arbeitsmarkt-a-919819.html (visited on 04/13/2019).

[73] Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. “Overview of the germeval 2018 shared task on the identification of offensive language”. In: (2018).

[74] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. *Transformers: State-of-the-art Natural Language Processing*. 2019. arXiv: 1910.03771 [cs.CL].

[75] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. “BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis”. In: *NAACL-HLT*. 2019.

[76] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[77] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *CoRR* abs/1212.5701 (2012). URL: http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-5701.

[78] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan R. Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 19–27.