University of Magdeburg

School of Computer Science

Master's Thesis

Classification of Multilingual Legal Text Using Deep Learning: Evaluation of General-Purpose Resources for Legal Domain-Specific Task

Author:

Jay Dilipbhai Vala

13. May 2019

Advisors:

Prof. Dr. rer. nat. habil. Gunter Saake

Department of Databases and Software Engineering

Prof. Dr.-Ing. Andreas Nürnberger

Department of Data and Knowledge Engineering

Vala, Jay Dilipbhai:
Classification of Multilingual Legal Text Using Deep Learning: Evaluation of General-Purpose Resources for Legal Domain-Specific Task
Master's Thesis, University of Magdeburg, 2019.

Abstract

The plethora of legal corpora available online can be overwhelming and hard to comprehend for a non-domain expert. Big law firms also need to organize and update legal documents to keep up with changes in regulations and legislation. These law firms employ experts who help them navigate and find useful information for legal documents in a timely manner. Artificial Intelligence (AI) can be helpful in the organization and exploration of these documents [MS97]. Classifying these legal texts into higher level categories using machine learning and deep learning can help in the organization and navigation of these huge corpora for big firms and the non-domain expert. The diversity, the advantages and the drawbacks of these algorithms make choosing the algorithm a time-consuming and challenging process.

This thesis goes into investigating a few popular machine learning and deep learning algorithms with different configurations on EUR-Lex Summaries legal data for text classification. It also examines the performance of general-purpose resources used by deep learning algorithms on legal domain-specific data and the performance benefits of using multilingual data which is widely available for the legal domain. These experiments help us in exploring the viability of these algorithms in domain-specific settings and improve our decision-making process. For the investigation, three different research questions are formulated, first comparing a supervised machine learning algorithm Support Vector Machines (SVM) to a deep learning algorithm Bidirectional Long Short Term Memory (BiLSTM), second comparing the general-purpose word embeddings to the domain-specific word embeddings for training BiLSTM and third comparing performance evaluation of classifiers using multilingual data to monolingual ones in BiLSTM.

Furthermore, with the limited labeled legal domain dataset, and class imbalance, several alternate training, and evaluation strategies were formulated. The training is done on sentences of the document with and without clustering, and the evaluation is done of sentences as well as on documents through the combination of predictions from the sentence. During the assessment, it is observed that adding more languages in case of a BiLSTM indeed increases the performance of the classifier.

Acknowledgements

> "Life is like a box of chocolates. You never know what you're gonna get"
> - Forrest Gump

Last year in November, I embarked on this fascinating journey of undertaking my Master thesis unaware what lies ahead, just like a box of chocolates. It gives me immense pleasure and a sense of satisfaction to finish this thesis. This thesis would not have been completed had there not been a few people to support and guide me all along.

First and foremost, I want to thank my supervisor M.Sc. Sabine Wehnert, without whom this thesis would not have been possible in the first place. Her constant support, discussions, and assistance at every step were very helpful. I want to thank M.Sc. Marcus Thiel, for his assistance and suggestions on validating various concepts, and Stefan Langer for his support and ideas.

I want to thank my Family for their moral and financial support. They stood by me in my thick and thin. I can not even begin to describe the gratitude and appreciation for their support.

I want to thank my friends for listening to me, and at times tolerating me. They were like a family away from home.

Last but not least I would like to thank everyone who has supported me directly or indirectly throughout this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BiLSTM	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
LSTM	Long Short Term Memory
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SVM	Support Vector Machines
TF-IDF	Term Frequency Inverse Document Frequency
TN	True Negative
TP	True Positive

1. Introduction

1.1 Introduction

There is an abundance of legal corpora available online. The publications office of the European Union (EU) offers different EU laws such as Treaties, Legal acts, Consolidated texts, International agreements, Preparatory documents and Summaries of EU Legislation. It also provides EU case laws, national laws, and national case laws. Other governments in the EU also publish legal text such as the *gesetze-im-internet* website, the German Federal Ministry of Justice and Consumer Protection and the Federal Office of Justice offerings of every federal law of Germany, the *Library of Congress* for the French law and many more.

Large legal text corpora can be overwhelming and hard to comprehend for non-domain experts, especially when jargon is used. Also, the complexity and the capricious nature of the legal texts is a challenge in itself. It can also be challenging for domain experts handling legal text written for different demographics, with different structure, and in different languages. Some of the most common problems with the legal texts are that it may be coming from different sources (different department of the government or from different governments altogether), it is difficult to know which laws overturn other laws, understanding different lexicon that changes with contexts, authority and over time [BDCH+12]. These barriers make it challenging to organize and aggregate these legal texts in a coherent manner. Traditional full-text search matches the exact text, and this helps in finding relevant information, but it is still challenging to discover all the relevant information such as in the cases where related words and synonyms are used for search [LWHM16]. To get around this, domain ontologies which map the relationships of synonyms, related words and antonyms are used; however, these are hard to create and difficult to maintain.

The ongoing advancements in Artificial Intelligence (AI) and Machine Learning which is partly due to the availability of computing power can be promising in mitigating some of the difficulties with legal texts mentioned above. Natural Language Processing (NLP) is a subset of AI that focuses on systems that can learn and understand language. Artificial Neural Network (ANN) are a part of AI which

was inspired by the biological neural network of human brain [vGB18]. They are a framework comprising many different machine learning algorithms learning complex data. The application of ANN is suitable for data that has noise which in case of legal text is having no sole representation of a law, that is the perception of law is in accordance with the requirements and needs. Secondly, there is a poor understanding intrinsic structure which means that there is no single entity which knows all the laws and lastly for the data having changing characteristics [MS97].

Various methods involving machine learning and AI have been developed to solve some of the problems with legal texts described above. de Maat et. al (2010) [dMKW+10] have shown the effectiveness of using machine learning algorithms to classify legal text compared to a pattern-based classification task on unseen data. Mozina et. al (2005) [MŽBCB05] showed that argumentation-based machine learning can be made using the justification of the case laws making it suitable for legal domain-specific tasks. Boella et. al (2011) [BDCH11] have shown that sophisticated legal document management systems can be further improved using machine learning, the authors used classification algorithms to classify laws and integrate into an existing system.

1.2 Goals of this Thesis

This thesis focuses on the classification tasks of legal text. There are various classification algorithms with their advantages and disadvantages. The goal of this thesis is to evaluate these algorithms in various configurations,

Specifically, this thesis will investigate three questions,

- Can deep neural network algorithms like Long Short Term Memory (LSTM) networks perform comparatively to the thoroughly studied text classification algorithms like Support Vector Machines (SVM)?

  The Support Vector Machines (SVM) has been thoroughly studied and applied in text classification [CC08, FGP06, For08, Joa98, Seb02, SLB+07]. SVMs have been shown to outperform many classification algorithms, but the sensitivity of the parameters to the classification task makes them difficult to use [CB06]. Also the text representation technique used by the SVM which is Term Frequency Inverse Document Frequency (TF-IDF) does not capture the syntactics and semantics of the text [CJMdS17]. Many deep learning classification algorithms alleviate the problems that SVMs posses, with text representation techniques such as word embeddings capturing the semantics and syntactics of the text to represent its context better.

- General-purpose resources such as pretrained word embeddings are getting better and are available from various sources and trained using different algorithms. As they are trained on large generic web scraped data, are they good enough for the legal domain-specific task?

  Recently, various word embeddings such as BERT (Bidirectional Encoder Representations from Transformers) [DCLT18], Fasttext word vectors in 157 languages [BGJM17], Flair [ABV18] and ELMo [PNI+18] were released. They,

however, use general data scraped from websites like Wikipedia or freely available news and other corpora. These corpora do not cover the highly specialized and specific vocabulary used in the legal text; hence the effectiveness of these word embeddings for a legal domain-specific task has to be investigated.

- Deep Neural Network algorithms like Long Short Term Memory (LSTM) can take advantage of multilingual data, as the text representation technique of word embeddings has been shown to work on multilingual data. Thus, the performance of LSTMs needs to be investigated in case of monolingual and multilingual data to see if there is any advantage of using multilingual data.

Conventionally, multilingual text classification is done by employing a language detector, which detects the language to be classified and then initiating a classifier that is trained on a specific language. This naive approach does not take advantage of the polylinguality of data for classification for a predefined category which may benefit from this data [WYL+14]. Recent techniques of learning bilingual word embeddings [ZSCM13, CAPLL+14] have shown to be improving, for this reason, the effect of multilingual embeddings is to be explored.

1.3 Structure of the Thesis

The thesis is divided into seven chapters. Chapter 2 provides details of the various machine learning and deep learning methods, algorithms and information about document categorization. Chapter 3 details the conceptual aspects for all the research questions and the techniques such as data cleaning, data collections that will be used to answer the research questions. Chapter 4 gives the detailed narrative of the tools used in implementing the concepts, the architecture and the hyperparameters of the algorithms provided in the Chapter 3 and the specifics of how they are implemented. Chapter 5 provides details of the evaluation approaches and the results of all the research questions. Chapter 6 shows the similar work done on the legal dataset, and similar techniques used in training and evaluation of the algorithms. Chapter 7 presents the conclusion of the thesis, discusses the limitations of some approaches and future works.

2. Background

This chapter covers the fundamentals of document classification, how machine learning is applied for document classification. Furthermore, this chapter will provide an overview of various machine learning, deep learning, and evaluation concepts. This chapter will also describe the concepts of different classification algorithms employed.

2.1 Document Categorization

Document categorization also known as document classification is the analysis and assignment of documents into some predefined classes. Categorization is an essential part of document retrieval as without categorization it is impossible to label the document and know when to present it to the user in response to a search query. Manually assigning these documents (also referred to as indexing) is not feasible due to the continuous addition of new documents every day.

Luhn first introduced the classification of documents for creating abstracts in technical resources. Luhn proposed using statistical analysis of word occurrences in the abstract and title of the literature to assign it to a predefined category [Luh58]. Many early adaptations of document categorization techniques relied upon carefully hand-crafted features [HW91, BFL+88]

2.2 Machine Learning and Document Categorization

This section briefly describes concepts of machine learning and its use in document and text categorization. Machine learning is a branch of AI that gives a computer the capability to learn by itself and improve based on seen examples. Mitchell formally defines it as, "*A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance measure P for the task T improves with experience E*" [Mit97]. Machine learning algorithms are broadly divided into two categories, *supervised* and *unsupervised* learning algorithms. Supervised learning algorithms learn from a given set of data and try to predict an

unseen target set. Unsupervised learning algorithms learn the relationships between elements of the provided data.

Text or document categorization is a supervised learning problem. We have a set of training documents which are labeled and used to train the machine learning algorithm. This trained classifier is then used on the target or test set to predict the label.

2.2.1 Supervised and Unsupervised Learning

In supervised learning, a machine learning algorithm learns the mapping of a given set of examples to some specified, predefined categories. Formally, let Z be the training set given by $\{(a_1, b_1), (a_2, b_2), (a_3, b_3), ..., (a_n, b_n)\}$ where a_i are the input vectors to the algorithm and b_i are the corresponding labels, then a supervised learning algorithm will try to learn the mapping function,

$$f(a) = b \tag{2.1}$$

The goal of this function is to approximate the mapping well enough that when a new data point (a) is passed to the algorithm it can predict label (b) from the data.

Contrary to the supervised way of learning, in unsupervised learning there are no corresponding labels; the algorithm has to model and learn the relationship of the underlying , for example using a similarity function to determine how similarity between two data points. It is similar to learning without a teacher. [HSP99].

2.3 Multi-class vs Multi-label Classification

Classification or categorization is the identification of an instance of data into a predefined category or class. Classification can be broadly divided into three types, *Binary Multi-class* and *Multi-label* classification. Multi-class classification is the process of identifying an instance into one of the many classes. In binary classification the instances are classified into two classes hence the term *binary*. Many algorithms solve the binary classification task. Several methods have been proposed to extend these binary classification problems into multi-class ones, as a multi-class problem can be seen as a set of several binary class problems [Aly05]. Some of these binary classification problems extend naturally to a multi-class problem, and some require special transformations to do so. An examples of this natural extension are Neural Networks as instead of having a single neuron at the output as in binary classification, it has N neurons for N classes [B$^+$95]. Other algorithms convert multi-class problems into a set of binary classification problems for example Support Vector Machines [CV95]

In the Multi-label classification problem, instead of identifying an instance of data into a single category or class, it is classified into multiple classes. Multi-label classification is common in case of document classification as a document might contain aspects of different topics, so it is attributed to multiple classes. The most common approach to tackle a multi-label problem is to transform it into n different binary or multi-class problems, and then the predictions from these n binary or multi-class classifiers are transformed into multi-label predictions [RPHF11].

2.4 K-means Clustering

It is often required to divide the data into groups to better understand it. Clustering is the method of grouping the data in such a manner that similar instances are grouped together (in the same cluster) and dissimilar objects are grouped separately (in different clusters). Clustering is extensively used in data mining for statistical analysis in the earlier stages of exploratory analysis. The k-means clustering algorithm is one of many unsupervised clustering algorithms.

The k-means algorithm separates data into k clusters. These clusters are such that they are as far away from each other as possible. Each cluster has a central data point (which is randomly initialized) called **centroid**, and an instance of the data is assigned to a cluster if it is close to this center. k-means iteratively minimizes the distance between every example of the data and the center to find the optimal number of clusters.

Steps involved in K-means clustering is as follows,

1. Initialize k data points as centers of the clusters.
2. Calculate the distance between every data point and the centroid and based on this distance allocate each point to the nearest cluster.
3. The cluster centers are updated by calculating the average of all the points of that cluster.
4. If the position of the centroids changes, then the process is repeated from step 2 until there is no further change in the position of the centroid.

The process stops when the average distance between the centroids and the distance between the points of a cluster is lowest. The minimal distance between the instances of a cluster ensures that the cluster is compact with the least variance between the points of a cluster.

2.4.1 Pairwise constrained K-means

Domain knowledge about the instances can be helpful in the better assignment of the instances to the clusters. It can be used to assess which instances should be grouped together [WCR+01]. There are two types of pairwise constraints.

- **Must-link**: This set of constraints specifies which of the instances of the data should be placed in the same cluster.
- **Cannot-link**: This set of constraints specifies which of the instances of the data should not be placed in the same cluster.

This is a modified version of k-means clustering with the constraints as mentioned above. The algorithm takes data D and a set of *must-link* constraints and a set of *cannot-link* constraints. As a result, the data is divided into k groups satisfying all the constraints. When a data point is assigned to a cluster, the algorithm ensures that it does not violate any of the specified constraints.

2.4.2 Choosing k

To produce high quality clusters, the number of clusters (k) that the data needs to be divided into should be known. *Silhouette Score* [Rou87] and *Elbow Analysis* [Tho53, KS96] are two methods to find out the number of clusters.

2.4.2.1 Silhouette Score

Silhouette score is a measure of the quality of a cluster. It determines how well an object fits the cluster assignment.There are two requirements for the creation of silhouettes, the first thing we need is the clusters, and second, we need the proximity between each object of a cluster.

Let $s(x)$ be the silhouette score for element x which is placed in cluster A as shown in Figure 2.1. We can calculate the dissimilarity (distance) $a(x)$ between elements x of cluster A and other elements of cluster A,

$$a(x) = \textit{average dissimilarity between } x \textit{ and other elements of cluster } A \tag{2.2}$$

In Figure 2.1 $a(x)$ is represented by the average length of all the lines of cluster A. Now we calculate the dissimilarity of element x with elements from any cluster other than A. For example, cluster B which is different from A.

$$d(x, B) = \textit{average dissimilarity between } x \textit{ and other elements of cluster } B \tag{2.3}$$

In the Figure 2.1, $d(x, B)$ is represented by the average of lines stretching from element x in the cluster A to all the elements in the cluster B. Once this is calculated for all the clusters, the minimum of these numbers is selected and denoted by

$$b(x) = minimum(d(x, B)) \tag{2.4}$$

Here, for cluster B we have the minimum of $b(x)$, so cluster B is called the *neighbour* of x, this is the second best option for x, if it cannot be accommodated in cluster A.

To calculate the silhouette score of $s(x)$, we use $a(x)$ and $b(x)$ obtained in Equation 2.2 and Equation 2.4

$$s(x) = \frac{b(x) - a(x)}{max\{a(x), b(x)\}} \tag{2.5}$$

Hence, the higher the value of $s(x)$, the better the assignment of the clusters.

2.4.2.2 Elbow Analysis

Elbow analysis [Tho53, KS96] is another method of finding the number of clusters (k). The k-means algorithm works by finding the clusters which minimizes the intra-cluster variance. The sum of squares of all the points in a cluster explains the variance within a cluster and hence it is used in elbow analysis to find the optimal number of clusters.

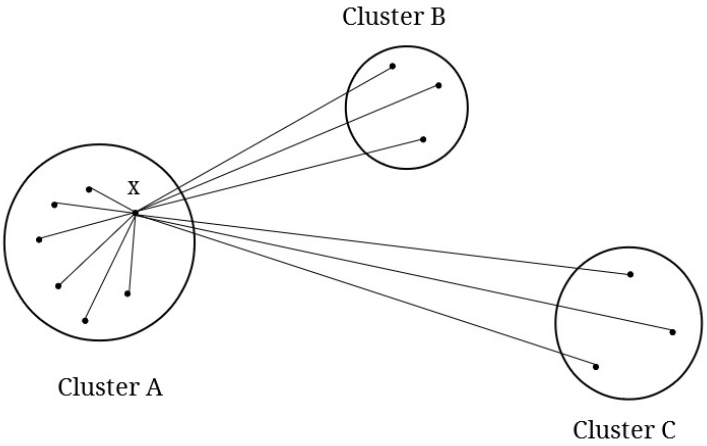


Figure 2.1: Computation of silhouette score $s(x)$ with three clusters for point x in cluster A.

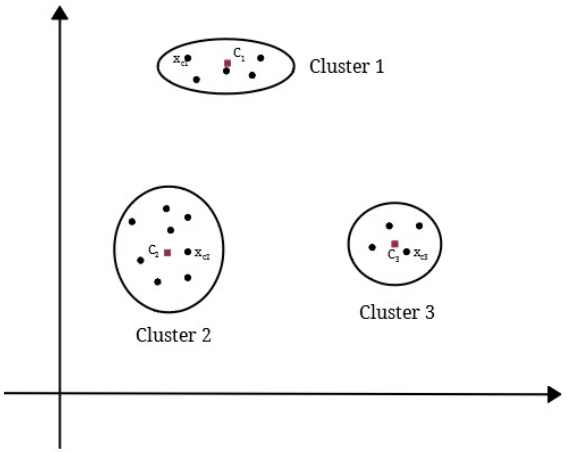


Figure 2.2: An illustration of clusters and centroids, with three clusters and their respective centroids C_1, C_2, C_3

Elbow analysis uses the Within Cluster Sum of Errors (WCSS). WCSS is calculated as the sum of distances of each point of a cluster and the centroid of that cluster.

For the clusters in the Figure 2.2, WCSS will be calculated as follows.

$$WCSS = \sum_{i=1}^{n} \sum_{k=1}^{n} distance(x_{i,j}, C_k) \tag{2.6}$$

where:

$$x_{i,k} = \text{Data point } i \text{ in cluster } k$$
$$C_k = \text{Centroid for cluster } k$$

This WCSS distance is plotted against the number of clusters which show an elbow shape; adding more clusters indicated by the elbow shape will decrease the WCSS further and will not model the data significantly better.

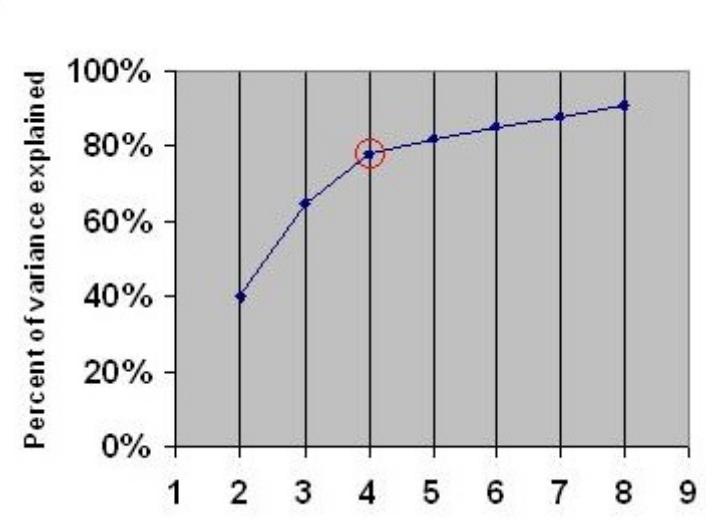


Figure 2.3: Example illustrating elbow at $k = 4$

In the Figure 2.3 we can see the elbow shape at $k = 4$ on the x-axis. This suggests that we divide the data into four clusters.

2.5 Support Vector Machine

An SVM is a supervised learning algorithm. It tries to find boundaries or hyperplanes in the given feature space which are then used for classification and regression. An SVM tries to maximize the distance between the two points of a decision surface [MRS10].

SVMs are based on *structural risk minimization*, which aims to find out a hypothesis h for which we have the lowest true error. This true error is the probability of making an error on unseen data. The ability of SVMs to learn from data is not dependent on the dimension of input features which makes them ideal for text categorization [Joa98].

To understand how an SVM works, consider the equation of a hyperplane which is given as [FHT01],

$$w \cdot x + b = 0 \tag{2.7}$$

where w represents a non-zero vector which is normal to the hyperplane, x is any point on the same space as the hyperplane and b is a scalar. Now, let us consider additional two hyperplanes which are normal:

$$w \cdot x + b = \pm 1 \tag{2.8}$$

Distance of any given point x_i is given by

$$d\left(w, b; x_i\right) = \frac{|w \cdot x_i + b|}{\|w\|} \tag{2.9}$$

The points on the hyperplanes defined by equation Equation 2.8 are known as support vectors. The other points have no influence on the classification process and

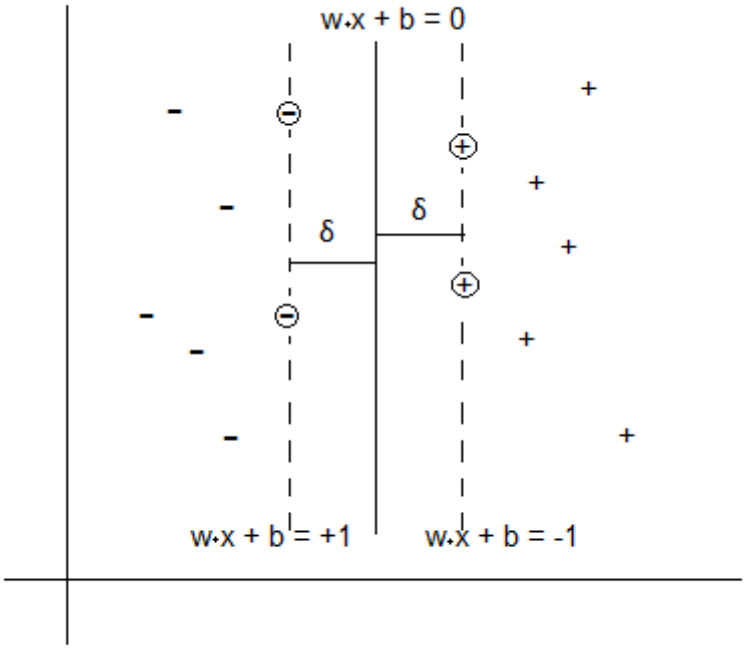


Figure 2.4: Support Vector Machines

we can further simplify the equation Equation 2.9 by replacing the numerator with 1:

$$d(w, b; |w \cdot x + b| = 1) = \frac{1}{\|w\|} \tag{2.10}$$

To optimize Equation 2.10 requires us to simply maximize $||w||^{-1}$, which is equal to minimizing $||w||^2$. Hence, we need to solve the optimiztion problem

$$\min_{w,b} \left(\frac{1}{2} \|w\|^2 \right) \tag{2.11}$$

The Equation 2.11 is subject to the constraint that the distance from a given point x_i must be equal to or greater than one. Otherwise, $w \to 0$ will always be the optimal solution.

$$y\,(w \cdot w_i + b) \geq 1, \forall i \in [1, m] \tag{2.12}$$

A change in support vectors of the SVM will only affect the margins and changes in any other points will not have any impact on the margins and the hyperplane found by the algorithm. For training the SVM we need to find C which is a regularization parameter. For large value of C the algorithm will choose a small margin hyperplane and for small values of C the algorithm will look for larger margin hyperplane separating the data.

2.6 Deep Learning

The rise in digitization brought in the problem of the creation of vast and diverse unstructured data. The increase in computation power and the need to analyze the data to get a better insight of it has given rise to deep learning. Deep learning is a subcategory of AI which is inspired by the human brain and replicates the way a human learns. ANN is a deep learning algorithm which imitates the human brain. A ANN has an input layer, an output layer along with hidden layers comprising of units that process the given input. ANNs are effective tools for recognizing patterns which are way too complicated for programmers to extract and teach the machine to understand. ANNs exist since the last century; however, in the last decade they have become a significant part of AI due to a technique called backpropagation and due to the availability of computational resources.

An artificial neuron is the basic component of an ANN and is shown below in Figure 2.5.

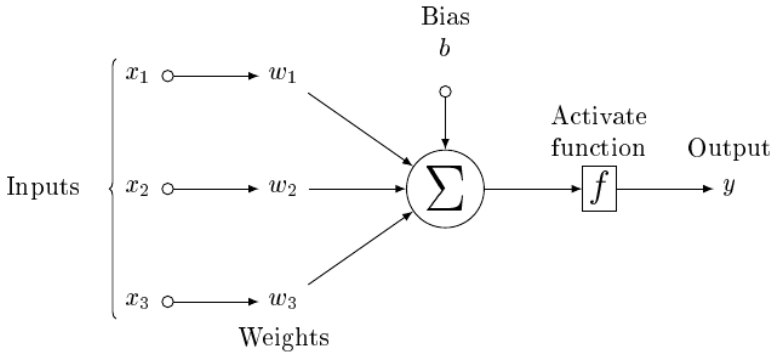


Figure 2.5: An Artificial Neuron

Each input x_i is multiplied with random weights w_i. Then they are summed up with biases b_i, and then this total input is passed through the activation function.

The output of the neuron in the Figure 2.5 would be as follows:

$$y = f(x_1w_1 + x_2w_2 + x_3w_3 + b) \tag{2.13}$$

Although simple in structure and computation, the full potential of these artificial neurons is realized when they are connected to form the ANN as shown in Figure 2.6.

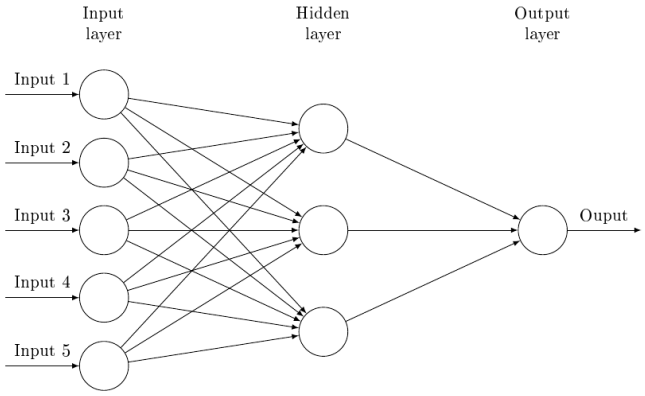
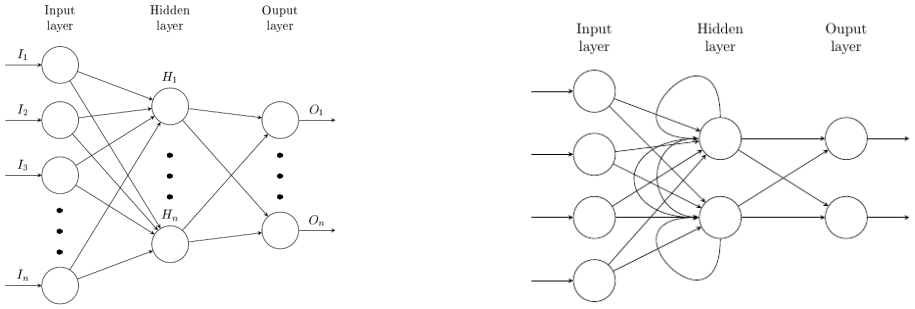


Figure 2.6: Artificial Neural Network

There are two main topologies in which these neurons are connected to form ANN as shown below.



(a) Feed forward neural network

(b) Recurrent neural network

Figure 2.7: Topologies of ANN, (a) shows feed forward neural network and (b) shows recurrent neural network

As we can see from the 2.7(a) that in a feed-forward networks the the flow of information is only in one direction, that is from the input layer to the output layer wherein in the recurrent neural network the flow of the information is in the forward as well as in the backward direction as shown in the 2.7(b)

Activation functions

Activation functions are important features of ANNs. Activation functions are described as a function which converts the input into outputs. Consider the example of the flow of current in an electric circuit similar to the flow of information in a neural network, switch in the circuit as the activation function for that circuit. Hence, an activation function is responsible for *ON* and *OFF* state of the circuit depending on the input. As the biological neuron inspires the artificial neurons, the activation function in biological from determines whether a neuron is firing.

The activation function is a non-linear transformation applied to the input. When the activation function is not applied to the inputs, the output that we get would be a linear transformation, which is rather easy to solve but will not model many real-world complex problems. A neural network without the activation function is a simple linear regression model. The use of an activation function is highly dependent on the problem at hand. Some of the most common activation functions are listed below.

Sigmoid activation function

Sigmoid is one of the most common activation function used in neural networks. The wide use of this activation function is a result of its ease to calculate its derivative, which makes calculating weights easier.The sigmoid activation can suffer from the vanishing gradient problem [BSF+94]. Vanishing gradient occurs when layers of a neural network have gradients of 0 because layers above them have saturated between 0 and 1. [MHN13]

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

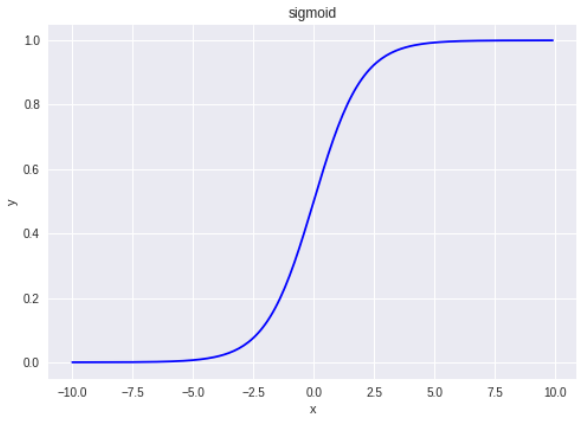


Figure 2.8: Sigmoid activation function

Rectified Linear Unit (ReLU) activation function

The Rectified Linear Unit function is one of the most popular activation functions from deep neural networks. The Rectified Linear Unit offers alternative nonlinearities compared to sigmoid functions, which mitigates the problem of the vanishing gradient, however during the optimization procedure when these units are inactive the gradient is 0 which leads to the problem of these units never getting activated as gradient based optimization will never adjust the weights of units which did not activate initially [MHN13]

$$f(z) = \begin{Bmatrix} 0 & z < 0 \\ z & z \geq 0 \end{Bmatrix} \tag{2.15}$$

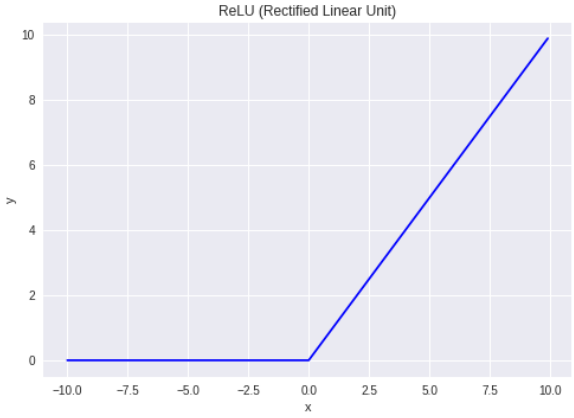


Figure 2.9: Rectified Linear Unit Activation Function

Hyperbolic Tangent

Hyperbolic Tangent activation function is the ratio of the hyperbolic sine function to the hyperbolic cosine function. This function is similar to the sigmoid function but outputs values between -1 and 1 as it can be seen in the Figure 2.10:

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.16}$$

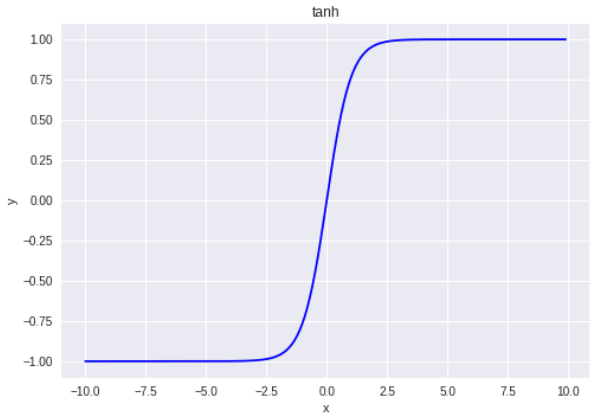


Figure 2.10: TanH Activation Function

Softmax activation function

Th softmax activation function calculates the probability distribution of a class over all the possible classes. Hence, the softmax activation function is used in various multi-class classification tasks in artificial neural networks [WLKF16].

$$f(z_i) = \frac{e^{z_i}}{\sum_{N=1}^{K} e^{z_i}} \tag{2.17}$$

Where:

$$f(z_i) = \text{Probability of } i_{th} \text{ class}$$
$$z_i = i_{th} \text{ dimension output}$$
$$N = \text{Total number of classes}$$

2.6.1 Backpropagation

Backpropagation is shorthand for "backward propagation of errors". In feed-forward neural networks information flows from the input layer to each hidden layer to produce the output which is referred to as **forward propagation** or **forward pass**. This forward pass continues to produce a scalar cost that is simply the difference between the target (what the network should have produced, true target) and the output (what the network produced). The **backpropagation** or **backward propagation** algorithm allows this cost to flow backward in the network to calculate the new weights of the network [GBC16]. It is the learning procedure which fine-tunes the weights of the connections of the network to minimize the error between the input the network is given and the output vector which is desired [RHW+88].

Consider a simple neural network with one input layer with 3 inputs namely x_1, x_2 and x_3, one hidden layer consisting of 2 neurons h_1 and h_2, and one output neuron, o_1. The activation function is a sigmoid activation and the cost function is simple Euclidean distance.

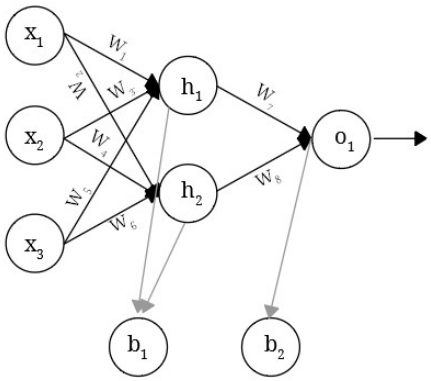


Figure 2.11: Single hidden layer neural network with one output

Forward pass

In the forward pass the network will calculate the *net* output of the hidden neurons, then apply the activation function to calculate the output of the hidden neurons and continue the process for the output layer neurons in the following way,

$$net_{h_1} = (x_1 w_1 + x_2 w_3 + x_3 w_5 + b_1) \tag{2.18}$$
$$net_{h_2} = (x_1 w_2 + x_2 w_4 + x_3 w_6 + b_2) \tag{2.19}$$

Applying the activation function (sigmoid activation) to the *net* output of the neurons of the hidden layers is done as follows:

$$out_{h_1} = \sigma(net_{h_1}) \tag{2.20}$$

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} \tag{2.21}$$

$$out_{h_2} = \frac{1}{1 + e^{-net_{h_2}}} \tag{2.22}$$

Then the *net* output of the output layer is calculated by,

$$net_{o_1} = (h_1 w_7 + h_2 w_8 + b) \tag{2.23}$$

Applying the activation function to the *net* output to calculate the output of the output neuron o_1

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} \tag{2.24}$$

After the output at the output layer is obtained, the network will calculate the error or the cost function,

$$E_{(target,output)} = \sum \frac{1}{2}(target - output)^2 \tag{2.25}$$

$$E_{total} = E_{o_1} \tag{2.26}$$

$$E_{total} = \frac{1}{2}(target_{o_1} - out_{o_1}) \tag{2.27}$$

The error is calculated, and this error is then used in the backward pass to calculate the new weights so that the error is minimized by the update (Summation $\sum$ in the Equation 2.25 indicates that the error will be calculated for each output neuron and then summed up).

Backward Pass

To update the weights of the network, the effect of the error on those weights is calculated, that is $\frac{\partial E}{\partial w}$, which is the rate of change of the error function with respect to the weights also referred to as *gradient* with respect to w. This is calculated by the chain rule of calculus which is used to calculate the derivative of an unknown function by calculating the derivative of known functions.

The derivative of $\frac{\partial E_{total}}{\partial w_7}$ using the chain rule is as follows,

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_7} \tag{2.28}$$

Calculating each term individually,

$$\frac{\partial E_{total}}{\partial out_{o_1}} = \text{change in total loss w.r.t output of } o_1 \tag{2.29}$$

From Equation 2.30, we can write,

$$E_{total} = \frac{1}{2}(target_{o_1} - out_{o_1})^2 \tag{2.30}$$

Taking the partial derivative of E_{total} with respect to out_{h_1}, the part $\frac{1}{2}(target_{o_2} - out_{o1})^2$ becomes 0 because out_{h_1} does not affect it and hence it is a constant.

$$\frac{\partial E_{total}}{\partial out_{o_1}} = 2 * \frac{1}{2}(target_{o_1} - out_{o_1})^{2-1} \tag{2.31}$$

Now for the second term in the equation Equation 2.28, we find out the rate of change of out_{o_1} w.r.t. net_{o_1}, hence we need to calculate the partial derivative of the sigmoid function.

$$out_{o_1} = \frac{1}{1 - e^{net_{o_1}}} \tag{2.32}$$

$$\frac{\partial out_{o_1}}{\partial net_{o_1}} = out_{o_1} * (1 - out_{o_1}) \tag{2.33}$$

$$\tag{2.34}$$

Finally, how much the net_{o_1} changes w.r.t w_7,

$$net_{o_1} = w_7 * out_{h_1} + W_8 * out_{h_2} + b_2 \tag{2.35}$$

$$\frac{\partial net_{o_1}}{\partial w_7} = 1 * out_{h_1} * w_7^{(1-1)} + 0 + 0 \tag{2.36}$$

Calculating the new weight value w_7 using all the values from Equation 2.28,

$$w_{7_{new}} = w_7 - \text{learning rate} * \frac{\partial E_{total}}{\partial w_7} \tag{2.37}$$

$$\tag{2.38}$$

To understand the process of backpropagation, please refer to Section A.1.

2.6.2 Backpropagation Through Time

The Backpropagation through Time (BPTT) is an extension of Backpropagation which is based on transforming a feedback network to a feedforward network by unfolding it over time [AIS04]. Therefore, when a network processes a sequence which is n times long, then n duplicate networks are created, and the feedback connections are changed to be feed-forward connections from one duplicate to another. This method is similar to training a large feed-forward neural network with weights being modified and treated as shared weights.

2.7 Natural Language Processing

Natural Language processing is a the sub-field of AI which deals with natural languages. Natural language in a written form is converted into numbers that a computer can process to find patterns. These patterns then can be used for various purposes such as text summarization, sentiment analysis and many more.

Processing and analyzing natural language can be divided into five stages [ID10] as shown in the figure below.

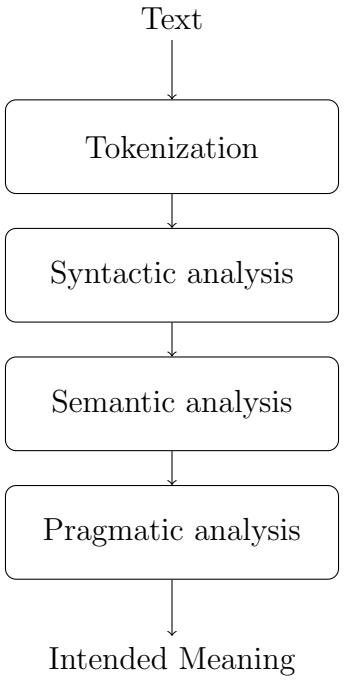


Figure 2.12: Stages of processing natural language, adapted from [ID10]

Taking an example of a sentence, tokenization is dividing the sentence into individual tokens (words or characters). Syntactic analysis is the analysis of the order and the structure of these tokens. Semantic analysis refers to the analysis of the meaning of words and their relations. Pragmatic analysis studies the context in text.

2.8 Text Normalization

Textual data contains a lot of inflection. It is the process of word formation which happens due to the representation of words for different grammatical categories such as tenses, voices. Removing these inflections for processing text is necessary.

Stemming and Lemmatization are two techniques used to normalize the text. Both techniques reduce the words to their root form. Stemming reduces the word into their base form, but this base may or may not be the morphological root of the word. It should be sufficient that the related words are mapped to the same base, even if the base is not a valid root. For example, words *argued*, *arguing*, *argue* and *argues* will all be stemmed to *argu*, even though the base *argu* is not a valid term in itself. The process of stemming is more heuristic. It removes affixes such as *-ed,-ize,*

-s,-de without taking into account that the base might not be a word in the same language. On the other hand, lemmatization reduces the words by ensuring that the base belongs to the language. The base word in lemmatization is called a *lemma*. Lemmatization requires a complete dictionary of the language being dealt with and morphological analysis to lemmatize correctly. Lemmatization is used in cases where it is necessary to get valid words and also in cases where verbs and nouns are to be treated differently.

Words	Stemming	Lemmatization
argue	argu	argue
arguing	argu	argue
argued	argu	argue
argues	argu	argue

Table 2.1: Comparison of word normalization techniques - stemming and lemmatization

2.9 Text Representation

All text-based computer systems require some representation of textual data depending on the type of problem at hand. Unlike other data formats, textual data is semi-structured or even unstructured. The representation of text in such a system is done by transforming the text into a numerical form. Detailed below are two of the most popular text representation techniques, TF-IDF, and Word Embeddings.

2.9.1 Term Frequency-Inverse Document Frequency

TF-IDF is the most common weighting method for describing textual data. Machine learning techniques such as Support Vector Machines and K Nearest Neighbours make use of this weighting scheme for text categorization. This technique works by first calculating *Term Frequency* which is how often a word appears in a particular document and *Inverse Document Frequency* which measure how infrequent a term is in the corpus [SM05].

Term Frequency is calculated as follows,

$$tf_{a,b} = \frac{n_{a,b}}{\sum_k n_{a,b}} \tag{2.39}$$

where:

a = word in the document
b = document from corpus
k = total documents in the corpus
$n_{a,b}$ = frequency of word a in document b

Inverse Document Frequency (IDF) is a measure of the importance of a word. In the case of *Term Frequency* all the words are considered equally important, but in the

case of Inverse Document Frequency, not all words are considered equally. While IDF measures the importance of words in the corpus, some words which are referred to as *stop words* such as *is, of, are, the* which appear quite often in the document are of very little importance. Hence IDF will weigh less the frequently appearing words in the corpus compared to rarely appearing words. [Rob04].

$$idf_w = \log \frac{Z}{z_a} \tag{2.40}$$

where:

$$Z = \text{number of documents in corpus}$$
$$z_a = \text{number of documents with term } a \text{ in it}$$

From equation Equation 2.39 and Equation 2.40, the $TF - IDF$ score is computed as,

$$w_{a,b} = tf_{a,b} \times \log \frac{N}{n_a} \tag{2.41}$$

2.9.2 Word Embeddings

Conventionally, text representation in natural language processing involves techniques like the *bag-of-words* model in which text is represented by the words in the text (like a bag with all the words of the text without any order), the *skip-gram* model which is a generalization of *n-gram* models where the words do not need to be successive for the text in consideration but can be skipped, hence named *skip-gram* [GAL+06] and *TF-IDF*. These techniques are a simple representation of various features of a text. However, in these techniques, the order of the words and the grammar is not considered. Hence, these techniques do not capture the semantics of the text. [MDP+11].

Due to the limitations mentioned above, word embeddings were introduced. Word embeddings are a vector representation of the meaning of words in the corpus. This means that words are placed in a high-dimensional vector space where words with a similar meaning are placed close to each other.

To better understand how word vectors are employed, we consider an example of reviews of cars from 0 to 5 in various categories, where 0 is worst, and 5 is the best category. A car "Volkswagen beetle" scores a 3 on comfort (one of the evaluation criteria). We can represent this score on a scale of -1 to 1 as shown in Figure 2.13.

Now, this representation as shown in Figure 2.13 does not consider all the aspect of the car. Therefore, we consider another evaluation criterion leg room (amount of space for your legs) where Beetle scored 4. Hence we represent it on a scale of -1 to 1 and add another dimension to the Figure 2.13 which represents the leg room category as shown in Figure 2.14.

Figure 2.13: Comfort of "Volkswagen beetle" on scale of 0-5, rescaled to range -1 to 1

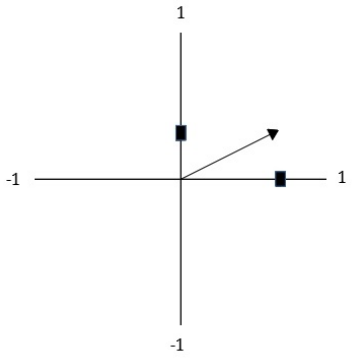


Figure 2.14: Comfort represented on x-axis and leg room on y-axis

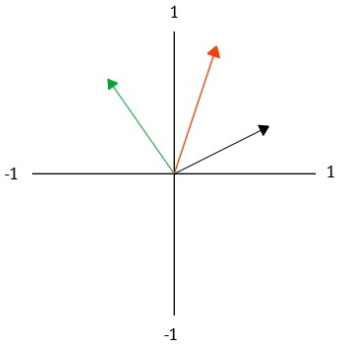


Figure 2.15: Vector representation of car features in a two dimensional space, black represents Volkswagen, red represents Mercedes and green represents Audi

Adding the comfort and leg room scores of a few other cars we have a vector representation of those cars in a two-dimensional space Figure 2.15.

Suppose someone likes his/her Mercedes and is in search of a similar car based on those two evaluation criteria. Using the vectors from Figure 2.15, we can easily find the similarity between two cars with the help of *cosine similarity*.

$$\text{cosine-similarity (Volkswagen, Mercedes)} = 0.955 \tag{2.42}$$

Cars	Vectors
Volkswagen	0.2, 0.4
Mercedes	0.1, 0.6
Audi	-0.3, 0.5

Table 2.2: Vectors of cars

$$\text{cosine-similarity (Volkswagen, Audi)} = 0.536 \tag{2.43}$$

$$\text{cosine-similarity (Mercedes, Audi)} = 0.761 \tag{2.44}$$

Adding more dimensions will capture more information and can increases the quality of the results. These vectors represent the cars with numbers which is necessary for machines to understand and also we can calculate similarities between these vectors. This is the basic principle as to why represent words as numeric vectors.

One popular algorithm is for learning the word vectors is **Word2Vec** [MCCD13], in this implementation word vectors also known as word embeddings are trained using two models, *continuous bag-of-words* model and *continuous skip-gram model*. In continuous bag-of-words, the projection of words into the vector space is not influenced by the order of words and words from the future are also used in this model. The continuous skip-gram model tries to increase the classification of the next word based on other words in a fixed window sized sentence, rather than predicting words based on the context. The main difference between both approaches is that in the continuous bag-of-words method, the model predicts words based on other words in its proximity and in the continuous skip-gram method, the model predicts the neighboring words based on the current word.

Consider the following sentence to understand how the aforementioned methods works,

> *I am the master of my fate, I am the captain of my soul* - Invictus, William Ernest Henley.

Continuous Bag-of-words Model

This model works by considering words not only from a few words in the past but also a few words from the future to predict the target word.

Figure 2.16: Continuous bag-of-words model data set creation

Consider a target word *"fate"*, and we want to predict it; thus we take a sliding window over the sentence and create a dataset for training the word vectors as shown in Figure 2.16.

This process will continue for the whole sentence in this case and the entire corpus in case we have a large dataset. This dataset is feed into a neural network.

Input 1	Input 2	Input 3	Input 4	Output
i	am	master	of	the
am	the	of	my	master
the	master	my	fate	of
master	of	fate	i	my
of	my	i	am	fate

Table 2.3: Dataset for training continuous bag-of-words model

Continuous Skip-gram Model

This model tries to predict the neighboring word based on the current word. So the training process for the sentence mentioned above is as follows.

We can take a sliding window over the sentence and construct a dataset for training the skip-gram model,

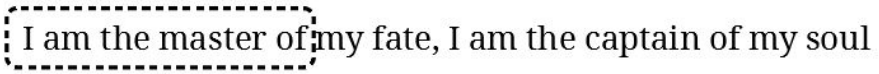


Figure 2.17: Continuous skip-gram model dataset creation

To understand how the dataset for continuous skip-gram training of the word2vec is created, consider the word *"the"* as input, and then we start building dataset around it. Two words before the word *"the"* and two words after it are taken as outputs as shown in the Table 2.4

Input	Output
the	i
the	am
the	master
the	of

Table 2.4: Dataset for training the continuous skip-gram model

This process will continue for the whole sentence until each every word of the sentence is in the training dataset against its neighboring words. This dataset is then fed to the untrained neural network to predict the output.

After the network converges, words with similar meaning are placed together in the embedding space. For example, words like "train" and "station" are placed close vicinity of each other in the vector space.

Figure 2.18 below shows words such as train, terminal, truck placed in close proximity of each other.

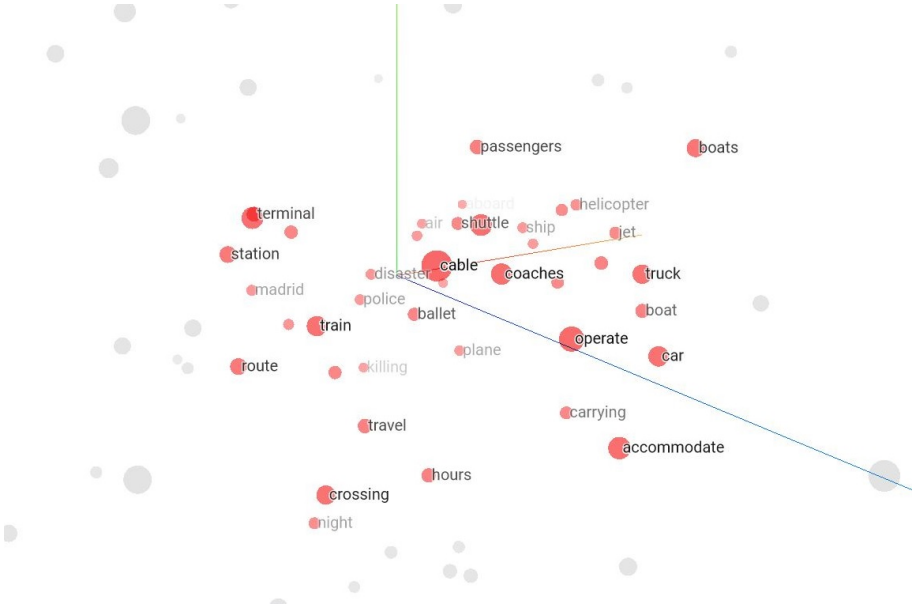


Figure 2.18: Word embedding showing words like train and station close to each other

2.9.3 Cross Lingual Word Embedding

Word vectors for different languages are in different vector spaces; hence they cannot be combined, but for classifying multilingual text, vectors from all the languages need to be in a single vector space or it has to be aligned into a single vector space. Using a single model for classification not only mitigates the error caused by a language detector in multilingual systems, where a language detector first detects the language and then the respective classifier is invoked. In that affair, the error propagates downwards and amplifies.

To train multilingual word embeddings Duong et al. [DKM+16], proposed using bilingual dictionaries and monolingual data. Their model uses an extension of the continuous bag-of-word (CBOW) model [MCCD13]. This method has benefits as often there is no parallel (multilingual text aligned together) data when working on domain-specific problems; also it is comparatively easy to obtain or create bilingual domain-specific dictionaries.

Facebook's MUSE Python library [CLR+17] aligns word vectors from multiple languages into a single vector space, they have used monolingual word embeddings trained using Facebook's Fasttext's [BGJM17] tool and aligned them in a common vector space. MUSE's algorithm learns the mapping between two embeddings and then aligns them together. It exploits the similarities of a monolingual embedding space [MLS13] to learn their mapping for alignment.

2.9.4 Long Short Term Memory

Neural networks have shown remarkable capabilities in processing and modeling natural language. Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are common architectures for handling natural language. The

LSTM [HS97] is a variant of RNNs, unlike other feedforward neural networks these networks have recurrent connections in them which allow the information to persist.

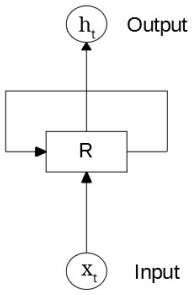


Figure 2.19: Single RNN Cell

The Figure 2.19 shows a single RNN cell having a recurrent connection. This recurrent connection allows it to retain the previous state and makes use of its context [GLF+09]. Due to this property RNNs are very effective in processing text.

RNNs are effective but suffer from the problem that their capacity to retain the previous states is fairly limited. Due to this the influence of the input either diminishes or increases exponentially. This is referred to as the problem of *vanishing and exploding gradients*[HBF+01]. The problem of the vanishing gradient makes it tough for an RNNs to make use of more than a few previous states. LSTMs are a special kind of RNNss which address the issue of vanishing gradients. They contain recurrently connected sub-networks called *memory blocks*, and each of these blocks contains three gates; the input gate, the forget gate and the output gate. These gates allow an LSTM to store information for a longer period. If the input gate is closed, that is if it has no activation (or close to 0), the activation of the block will not be overwritten for the next incoming inputs. Also, the activation of the block is only available to the network if the output gate is open. The forget gate switches the recurrent connections on and off.

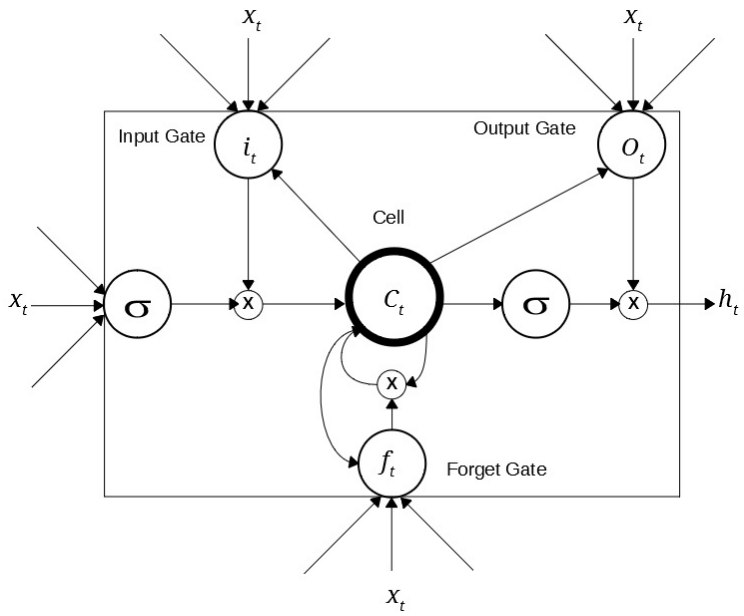


Figure 2.20: LSTM block with a single cell. The cell has recurrent connections with a forget gate. The three gates collect inputs from rest of the network.

An LSTM network learns the mapping of an input sequence to the output sequence using the following equations one by one from time $t = 1$ to n. Hence, the equations for the respective gates and the cell states are as follows:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.45}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2.46}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{2.47}$$

where:

i_t = input gate
f_t = forget gate
o_t = output gate
σ = sigmoid function
W_i = Weight matrix of input gate
W_f = Weight matrix of forget gate
W_o = Weight matrix of output gate
h_{t-1} = output from previous block at time $t - 1$
b_i = bias for input gate
b_f = bias for forget gate
b_o = bias for output gate

The Equation 2.45 is for the input gate which allows new information is to be stored in the cell state. Equation 2.46 is for the forget gate which is responsible for discarding information that is not useful anymore from the cell state. Equation 2.47 is for the output gate which is responsible for the activation of the whole LSTM block at time t

$$\tilde{c}_t = \tanh(weight_{cell}[h_{t-1}, x_t] + bais_{\text{cell}}) \tag{2.48}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c} \tag{2.49}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.50}$$

where:

$\tilde{c_t}$ = new contender for cell state at t
c_t = current cell state at t
$\odot$ = element-wise product

At any given time step, the current cell state calculates what is to be forgotten, that is $f_t \odot c_{t-1}$ from Equation 2.49 and which new cell state to be established from the current time step, that is $i_t \odot \tilde{c}$ form Equation 2.49. These are then passed through a tanh function to filter out which of the two should be forgotten and which should be the output at the current time step. The output h_t can then be passed through a softmax function to get the prediction probabilities for the current block.

2.10 Bidirectional Long Short Term Memory

Bidirectional Long Short Term Memorys (BiLSTMs) are able to process a given sequence in both directions (forward and backward) [SP97]. In Figure 2.21 we can see that a BiLSTM contains two layers out of which one processes the sequence in the forward direction and one processes the sequence in the backward direction. The output layer is connected to both layers which enables it to process both forward or future context and backward or past context. BiLSTMs have been show to performed better than standard LSTMs and RNNss in sequence learning problems [BBF+00] [FSS99].

From Figure 2.21, it can be seen that there is no connection between the forward passing layer and backward passing layer, hence outputs from the forward pass are not connected to the input of the backward pass and vice versa, so training a BiLSTM is the same as unidirectional LSTM for the most part.

2.11 Sentence-based approach for training LSTMs

Documents containing legal text are often very long. Although these long sequences can be processed and trained using LSTMs [HS97] theoretically, practically it is often limited by hardware. To process such long documents, they can be divided into smaller sub-texts which are easy to process. In [VA16], the authors have divided the text into sub-texts to consider the dependencies between the written text.

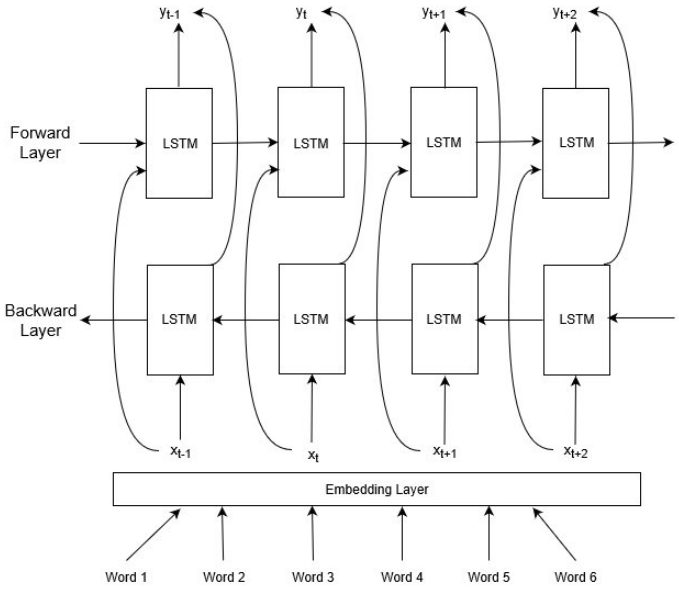


Figure 2.21: Bidirectional LSTM

Various other techniques have been studied and applied on textual data on various levels. Text data has been annotated, on document level [MO06], on paragraph level [FOD+09], on sentence level [SHMO09, SEK+08] and on the phrase level [WWH05] for various NLP tasks.

A *sliding-window* based approach is also used in this thesis for breaking the long documents into smaller chunks.

To understand how the sliding window algorithm works, consider a quote by Mahatma Gandhi - *"A man is but the product of his thoughts; what he thinks, he becomes."*

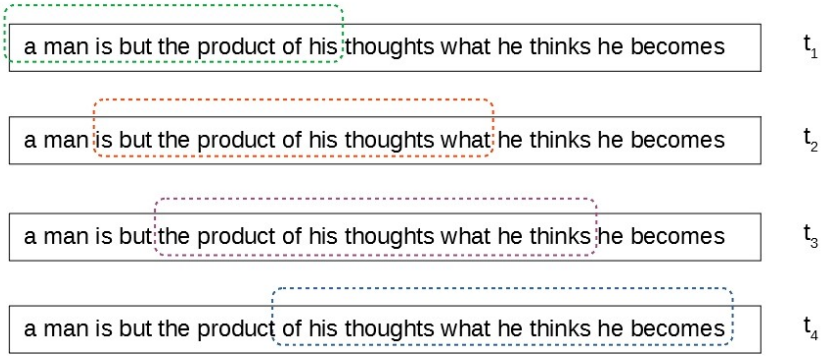


Figure 2.22: Sliding window at time step t_1, t_2, t_3 and t_4 with the window size of 10 words per sentence and slide of 2 word per sentence per time step

Figure 2.22 shows a sliding window, with a window size of 10 words per sentence and a slide of 2 words per sentence to create four sentences at each time step t_1, t_2,

t_3 and t_4. The following is the list of sentences that the algorithm will create with the configuration as mentioned above.

- Sentence 1 at time step t_1: a man is but the product of his
- Sentence 2 at time step t_2: is but the product of his thoughts what
- Sentence 3 at time step t_3: the product of his thoughts what he thinks
- Sentence 1 at time step t_4: of his thoughts what he thinks he becomes

In the case when there are fewer words left than the specified length of the window, then the process will halt with the remaining words in the last sentence.

This approach not only enables us to train the LSTMs on long text sequences but it also increases the training data. Each sentence created by applying the sliding window approach on a single document will have the same class label as the document. Therefore if **Doc A** $\rightarrow$ **Class 1** then all the sentences of that corresponding document will be labeled as **Class 1**

2.12 Evaluation Matrices

The evaluation of a classification model is an essential task. An evaluation of a model gives us insights on how the model will perform in the real world. The following is the list of evaluation measures used in this thesis to evaluate the classification performance of the models.

To evaluate the performance of any multi-class classification model, we need first to understand the *confusion matrix*.

A confusion matrix shows the performance of a classifier in a tabulated form on data where the labels of the data are known.

		Actual Values	
		P	**N**
Predicted Values	**P**	True Positive	False Positive
	N	False Negative	True Negative

Table 2.5: Confusion Matrix

- **Positive (P)**: Positive condition (e.g. is mango)
- **Negative (N)**: Negative condition (e.g. is not a mango)
- **True Positive (TP)**: Number of instances that were Positive (**P**) and identified and as Positive (**P**) by the algorithm.
- **False Positive (FP)**: Number of instances that were Negative (**N**) but identified as Positive (**P**) by the algorithm.

- **True Negative (TN)**: Number of instances that were Negative (**N**) and predicted as Negative (**N**) by the algorithm.
- **False Negative (FN)**: Number of instances that were Positive (**P**) but identified as Negative (**N**)

Accuracy

Accuracy is the ratio of the number of rightly identified instances of all classes over the total number of instances in the dataset.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.51}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.52}$$

Precision

Informally, precision indicates how many of the positive instances were identified correctly. Formally, it is the ratio of TP and sum of TP and FP.

$$Precision = \frac{\text{Correctly identified positive instances}}{\text{Total number of positive instances}} \tag{2.53}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.54}$$

Recall

Recall also known as True positive rate is the number of positive values that are correctly identified, that is the ratio of TP and sum of TP and FN.

$$Recall = \frac{\text{Correctly identified positive conditions}}{\text{Total number of positive prediction by algorithm}} \tag{2.55}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.56}$$

F-measure (F1-Score)

F-measure is the harmonic mean of precision and recall. The harmonic mean is one of the several averaging strategies applicable in a situation where the average of rates is needed. It is expressed in terms of precision and recall as following,

$$\text{F-measure (F1-Score)} = \left(\frac{precision^{-1} * recall^{-1}}{2} \right) \tag{2.57}$$

$$\text{F-measure (F1-Score)} = 2 * \frac{precision * recall}{precision + recall} \tag{2.58}$$

Macro and Micro Averages

Precision, Recall, and F-measure will give us the performance of the classifier on individual classes and to represent the performance of the classifier on the whole dataset we need to average these values across all the categories. The results of averaging the values of precision, recall, and f-measure across all the classes in case of an imbalance and/or multi-label dataset is misleading as it will weigh each class equally even though the number of samples in some categories was higher and therefore the effect of these classes on precision, recall, and f-score will be more.

Consider the following example:

- Class A: 5 TP and 5 FP
- Class B: 1 TP and 1 FP
- Class C: 20 TP and 90 FP
- Class D: 10 TP and 10 FP

By calculating precision on these classes we get, $P_A, P_B, P_D = 0.5$ and $P_C = 0.2$.

In macro-average, first the individual precision is calculated, and then the these precision values are averaged [MRS09], so macro-average is:

$$\text{macro-average precision} = \frac{0.5 + 0.5 + 0.5 + 0.18}{4} = 0.42 \tag{2.59}$$

The problem is quite clear as class C which is 84% of the total dataset is represented in the macro-average precision by $\frac{1}{4}$ ratio [MRS09].

To mitigate this, in micro-average, we sum up all the predictions of all the classes and then compute the average. So, the micro-average precision for the above example would be:

$$\text{micro-average precision} = \frac{5 + 1 + 20 + 10}{10 + 2 + 110 + 20} = 0.2535 \tag{2.60}$$

Hence, if the performance of micro-average metrics is better than the performance of macro-average metrics, it means that the classifier is performing better on minority classes.

3. Concept

This chapter describes in detail the conceptual aspects of this thesis. The concept is divided into three phases, in the first phase data collection and cleaning aspects are described, in the next phase clustering, and resampling of the data is described, and in the last phase, machine learning and deep learning models with their evaluation strategies are described.

3.1 Phase 1: Data Collection and Cleaning

In the first phase, the data collection techniques are described in detail, also how the data was cleaned to make it suitable for the machine learning algorithm is described here.

Data Collection

Legal laws are written to be applicable in various situations. For example, a law for agricultural products may be applicable in cases where trading of goods happens. Laws are also comprised of many languages so that they can be used by various nations. The applicability of these legal texts in different situations makes the classification of these texts a multi-label classification problem.

There are many datasets available for the legal domain. CFPB Credit Card Agreements DB[1], EUR-Lex Dataset[2] are some well known datasets for legal domain-specific tasks. However, the CFPB credit card agreements dataset is not available in the German language and also it is about one specific topic in the legal domain. The EUR-Lex Dataset is a large dataset available in both English and German, which was scraped from the Publication's office of European Union's website [MF10]. This dataset consists of 19,348 documents for a single language divided into 201 topics. This dataset is too large to handle as there are 201 classes given the amount of resources available to scrape, process and train the models. So instead EUR-Lex

[1] http://www.consumerfinance.gov/credit-cards/agreements/
[2] http://www.ke.tu-darmstadt.de/resources/eurlex

summaries are selected. This EUR-Lex Summaries[3] are also from the Publication office of the European Union's Parliament. These summaries are the European Union's laws explained and translated into a reader-friendly language. They cover 32 topics which correspond to the activity areas of the European Union. These summaries are available in multiple languages. The EUR-Lex summaries dataset is highly imbalanced. For this thesis, only *English and German* languages are considered. The summaries are not available in an easy, click to download manner, and hence they have to be crawled from the European Union's publication office's website. These summaries frequently change as the laws of the union change or if a new law is introduced.

While scraping the summaries, data from both languages need to be scraped together. This ensures that we are not missing out any summary from either of the languages for a fair comparison of multilingual and monolingual models. Saving the summaries from different languages separately could create the problem of data leaking between the English and the German version of the same document, which is the result of splitting data randomly for training and testing. For example, **Doc A** which belongs to **class 1** in English and **Doc B** which is the same document but in German which also belongs to **class 1**. Now, when spitting the data for training and testing **Doc A** might end up in the training set, and **Doc B** might end up in the testing set.

Data Cleaning

Text data is unstructured and noisy, meaning that it is not organized in a predefined manner and noisy because it contains a lot of information that may not be of any use for the task at hand. These irregularities make it difficult for a machine learning algorithm to learn from it. Textual data is non-uniform as the structure and content changes from one document to another, due to this non-uniformity it is challenging to have a single cleaning mechanism for all machine learning problems involving text. For example, a machine learning task that is designed to detect events might need date and time to effectively predict the event, whereas it would be viable to remove them for a sentiment analysis task.

Legal text is different from other text available because of the style, structure [BDCH11] and the use of special characters which are exclusive to the legal text. General text preprocessing techniques become feeble, hence during preprocessing of legal text special care is taken. The data set used here is the summarised version of the original legislation, which contains links to the original documents as well as background information about the law and other information that might not contribute anything in the classification process. These chunks of information can be removed, but it is particularly hard to remove them as they do not follow any pattern to where they appear in the document. Due to this reason, even though this information will increase the training time, it was not removed due to its demanding effort.

[3]https://eur-lex.europa.eu/browse/summaries.html

3.2 Phase 2: Data Clustering and Resampling

In the second phase, the data is first resampled and then clustered to prepare the data for training in the next phase. Also, in this phase, the word embeddings are created.

Data Resampling

Data for almost all the real world applications is in skewed distribution, and EUR-Lex summaries are no exception. Skewed distribution is when we have more samples in one class than others [WY12]. The skewed distribution also known as class imbalance in data is a difficult situation for a classifier to learn from as the classifier may be biased towards the majority classes and may shows poor performance on minority classes. Also, in the case of class imbalance in the dataset, accuracy is not a good performance evaluation measure [CBHK02].

To handle the imbalances in the dataset various techniques have been proposed. *Oversampling* and *Undersampling* are the two most common ones. Oversampling refers to the process of artificially increasing the number of samples in the minority classes, and Undersampling is when we remove instances from the majority class to even out the dataset. Undersampling in our case would mean that we reduce the instances from the majority classes which are few for classification task. And we could not find any work of oversampling a domain-specific and limited textual dataset using techniques like SMOTE [CBHK02]. Also it is questionable whether adding noisy artificial instances by SMOTE could really increase performance, given the small size of the dataset to train a generating algorithm.

The EUR-Lex summaries have multiple labels per document, this is generally the case with legal text as a single case may be applicable to many situations. So having multiple labels for a single document can be exploited to alleviate the skewness in the dataset. The sampling of the documents is done in such a way that it reduces samples from the majority classes on the basis of its presence in the minority classes. Hence, a document will be labeled as belonging to the class with the the fewest examples.

The resampling technique can be better understood with the following example. Consider the dataset in the Table 3.1 and the distribution of samples across classes in Table 3.2.

Doc ID	**Class Label**
Document A	1,5,2
Document B	3,2,5
Document C	4,1,5
Document D	2,3
Document E	2,5

Table 3.1: An table showing documents and their assignments in their respective classes.

Class ID	Number of Samples
1	2
2	4
3	1
4	1
5	4

Table 3.2: Distribution of samples in the dataset across 5 classes.

When applying the proposed resampling technique to the dataset in Table 3.1 and Table 3.2, *Document A* which has label 1,5 and 2 will be assigned to class 1 because the number of samples in class 1 are less than the number of samples in class 2 and class 5, similarly *Document B* will be assigned to class 3, *Document C* to class 4, *Document D* to class 2 and *Document E* to either 2 or 5. After resampling the Table 3.1 have labels shown in Table 3.3 and the multi-label dataset becomes a multiclass dataset.

Doc ID	Class Label
Document A	1
Document B	3
Document C	4
Document D	2
Document E	2 or 5

Table 3.3: Document assignment after proposed resampling

Class ID	Number of Samples
1	1
2	1 or 2
3	1
4	1
5	0 or 1

Table 3.4: Distribution of samples in the dataset across 5 classes after resampling.

Data Clustering

To see the effect of clustering, after resampling, the data is divided into n groups. To find the number of groups that the data can be divided into, a calculation of Silhouette Score and an Elbow Analysis are performed as described in Section 2.4.2. The division is so that all documents d of a category C_l will be grouped together in the same cluster K_n via must-link constraints.

That is, every document of a single class label must be present in a single cluster (must-link constraints). Each document will be assigned to the closest cluster without violating the must-link constraints. To achieve this, a constrained version of the k-means algorithm is applied (see Section 2.4.1). Only must-link constraints are used in the form of class labels, as each document will belong to one of the 32 classes, however cannot-link constraints, can not be obtained as we do not possess this information or have any prior knowledge about which two documents will not belong together.

When splitting the data in disjunct cluster, the number of classes to be predicted by each multi-class classifier of a cluster is reduced, thus resulting in specialization advantages for each classifier.

Monolingual and Bilingual Word Embeddings

The training of the word embeddings is the last step of the second phase. Three word embeddings are created, the monolingual word embedding, one bilingual word embedding trained on legal data and another bilingual word embedding trained on general-purpose data.

The details of the training the word embeddings will be discussed in the next chapter.

3.3 Phase 3: Model Architecture and Training

The third phase goes into details regarding the conceptual aspect of the three research questions. Different evaluation strategies for comparing them are introduced.

3.3.1 Research Question 1:

The first research question is:

> Can *BiLSTMs* achieve better results in terms of the evaluation metrics (Precision, Recall, and F-Score) than the thoroughly studied text classification methods such as *Support Vector Machines*?

Figure 3.1 shows the workflow for the above-mentioned research question. After acquiring the data from the EUR-Lex website, the data is cleaned or prepossessed (see Section 3.1). Upon preprocessing, the textual data needs to be converted using a suitable text representation technique according to the algorithm it will be trained on. Here, there are two algorithms being tested, SVM and BiLSTM, and both of them employ different techniques to represent text for training (See Section 2.9).

The SVM uses the TF-IDF representation whereas the BiLSTMs uses the Word Embeddings.

Due to hardware limitations training the BiLSTM is difficult if the length of a text sequence is very large. Hence each document is divided into sentences using the sliding window technique (See Section 2.11) and the document label is assigned to each sentence created from that document.

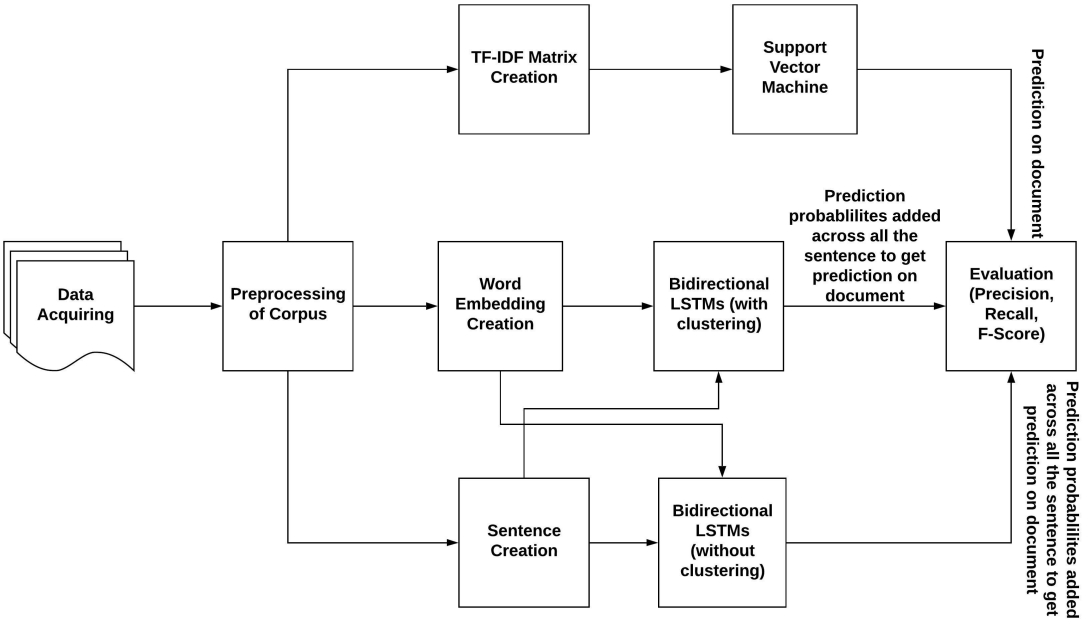


Figure 3.1: Flow chart representing the workflow for the first research question

To see the effect of clustering on the classification process, one classification model is trained on clustered data in a hierarchical manner. The clustered data contains documents from the corpora of both the languages; this is because clustering on a corpus of a single language will result in very less data for the clusters. This will show us if there is any improvement in classification performance of the classifier with the specialization advantage discussed in Section 3.2. The training workflow for the BiLSTM (with clustering) is shown below in Figure 3.2,

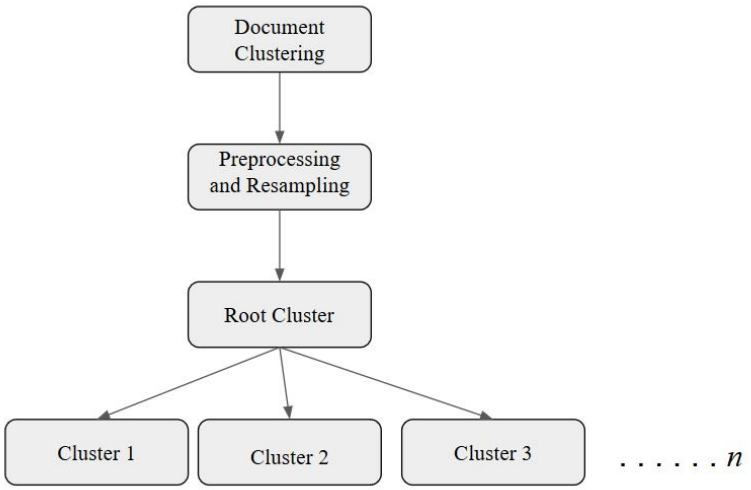


Figure 3.2: Classification workflow for clustered data in hierarchical manner

Evaluation

The SVM is trained on full documents, and the BiLSTM is trained on the sentences created using the sliding window technique, for this reason, results from both models cannot be compared. Ferguson et al. [FOD+09] performed sentiment analysis of financial blogs on paragraph-level annotated text, they summed up the predictions given by the classifier. Similarly, in our case, the classifier gives a probability to each sentence belonging to each class. As each document is comprised of sentences, if we add all the probabilities for each class across all the sentences of a document we get probabilities for every class for that document, and then we assign the class with highest probability score to the document. The classification algorithm trained on clustered data yields n different sets of predictions, i.e., the scores from the n clusters are averaged to get the score on the whole dataset.

The Table 3.5 lists all the details of the classification algorithms used for the first research question.

Classification Algorithm	Text Representation	Corpus	Type of training	Evaluation Strategy
SVM	TF-IDF	English	Non Clustered	Document Level
BiLSTM	Monolingual Word Embeddings (Legal Text)	English	Non Clustered	Document Level and Sentence Level
BiLSTM	Bilingual Word Embedding (Legal Text)	English and German	Non Clustered	Document Level and Sentence Level
BiLSTM	Bilingual Word Embedding (Legal Text)	English and German	Clustered	Document Level and Sentence Level

Table 3.5: Classification algorithms for the first research question with their corresponding text representation techniques, the corpus used to train them, the type of training (clustered or non clustered) and the evaluation strategies.

3.3.2 Research Question 2:

The second research question is:

> Are general-purpose resources such as *pre-trained word embeddings* in case of *LSTMs* applicable to specific legal domain tasks in terms of the evaluation metrics? Also, further training them on the legal corpus, produces comparable results to the ones only trained on legal texts?

Figure 3.3 shows the general workflow for the second question. For this question we compare the general-purpose resources, that is word embeddings trained on textual data available from many different domains, with the word embeddings trained on the legal corpus.

As the domain-specific word embeddings are created using legal domain-specific data this word embedding will not be trained further. In case of general-purpose word embeddings, two models will be trained - one with a frozen (embedding weights will not be updated during the training process) embedding layer and other models will train the word embeddings. This is done to see if further training the general-purpose word embeddings produces results comparable to the ones trained on legal data.

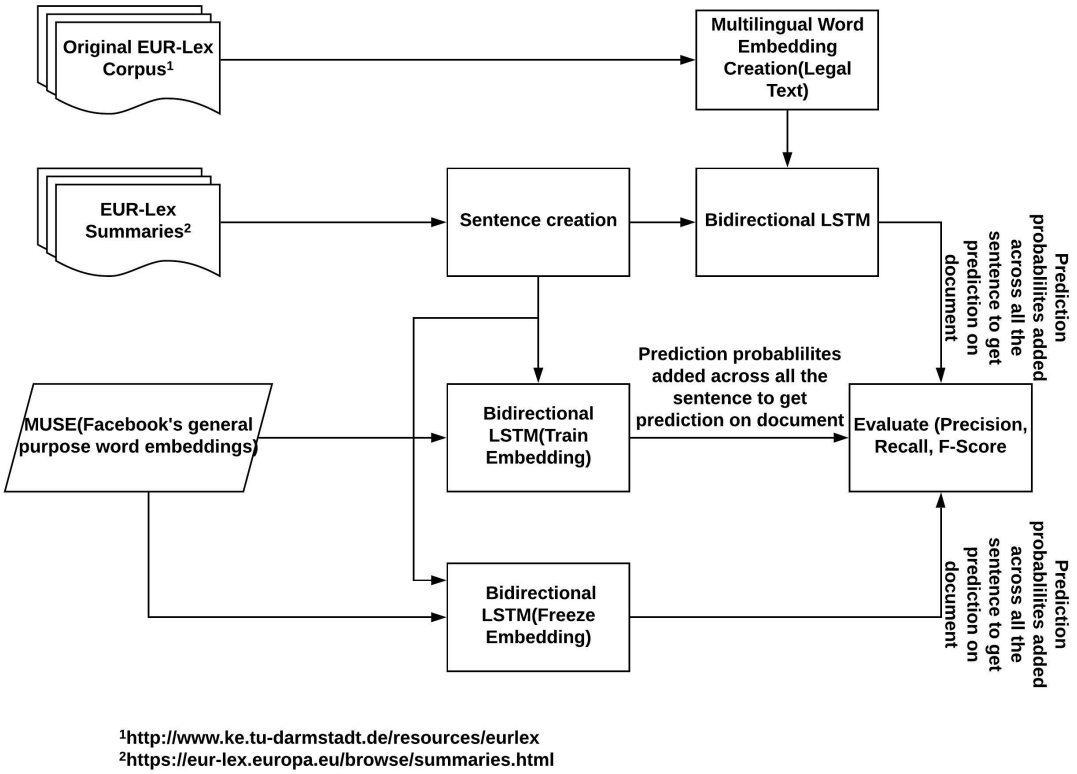


Figure 3.3: Flow chart representing the workflow for the second research question

Evaluation

Similar to the first question, the classification models are trained on sentences for the English and German corpora, hence all the models are evaluated on document level and sentence level with micro-average precision, recall and f1-score performance matrices as micro-average scores gives a better insight on how the classifier performs on minority classes. All the classification models are trained on clustered data in the similar way as shown in the Figure 3.2. As the training is done on the clustered data, the scores of the performance matrices are averaged together similar to the evaluation done in research question 1 (see Section 3.3.1).

The Table 3.6 below lists all the classification algorithms for the second research question,

Classification Algorithm	Word Embeddings	Embedding Training	Corpus	Type of Training	Evaluation Strategy
BiLSTM	Domain-Specific Bilingual (Legal Text)	False	English and German	Clustered	Document Level and Sentence Level
BiLSTM	General-Purpose Bilingual (Facebook's MUSE)	False	English and German	Clustered	Document Level and Sentence Level
BiLSTM	General-Purpose Bilingual (Facebook's MUSE)	True	English and German	Clustered	Document Level and Sentence Level

Table 3.6: Classification algorithms for the second research question with their corresponding text representation techniques, the corpus used to train them, the type of training (clustered or non clustered) and the evaluation strategies

3.3.3 Research Question 3:

The third research question is:

> Can *BiLSTM* perform better when training multiple languages in a single model, compared to training one model for each language separately?

Figure 3.4 shows the general workflow for the third research question. In this question, we explore the possible effects of having data in multiple languages on a classifier. C.-P. Wei et al.[WYL+14] suggest that when the training set for a single language is small it is less representative of the target semantic space and a classifier trained on this data set may not yield satisfactory results. However, if the text from another language is available for the predefined category then it can be beneficial and can improve the effectiveness of the classifier for the respective language. The naïve approaches of handling such data is to build separate classifiers for separate languages. This naïve method does not take the benefit of the availability of data across different language for a defined category.

To see the effect of the advantage of multilingual data, in this research question, first, two separate classifiers are trained on two different languages; namely, *English* and

German. The results from both of them are combined to get the average performance (we will average the micro-average precision, recall, and f-score across both the models to get the relative performance of the classifier on both the language) of classification of these two classifiers. The training and evaluation of these classifiers is similar to the training done in research question 1 and 2. This performance of the two classifiers is then compared to a single classifier that is trained on both the languages with multilingual word embeddings.

The training of all the classifiers for this question is done in a similar fashion as shown in Figure 3.2, and Table 3.7 below lists all the classification algorithms for the third research question,

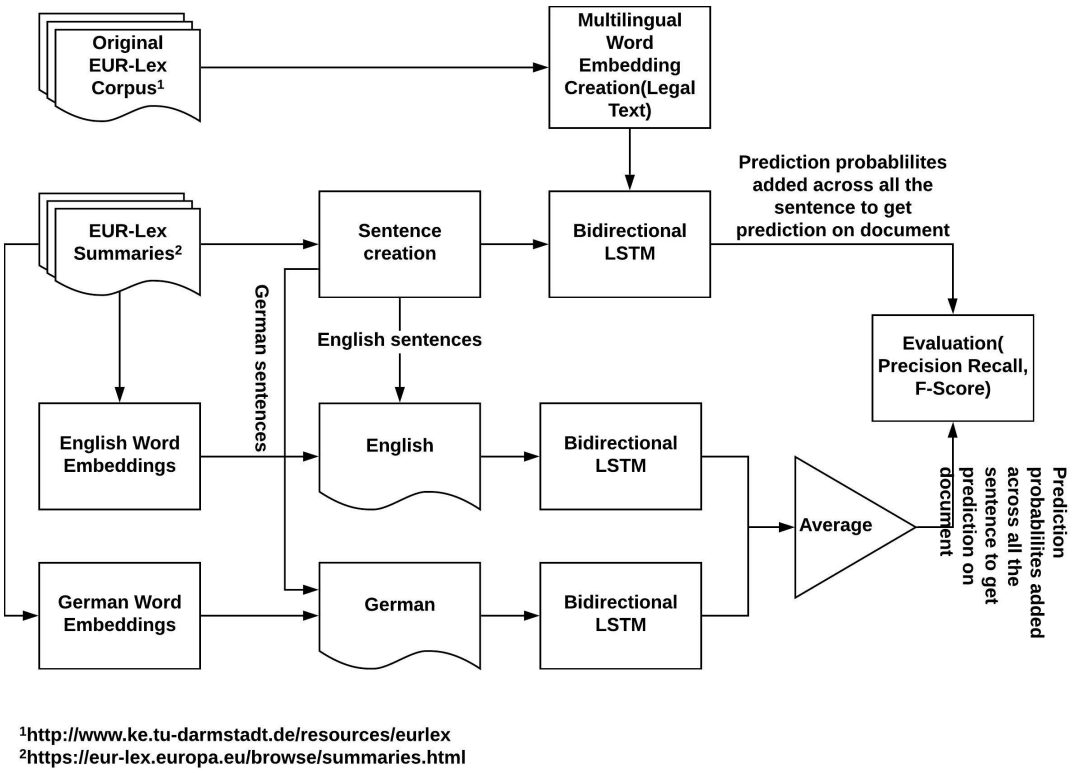


Figure 3.4: Flow chart representing the workflow for the third research question.

Evaluation

For the evaluation, the multilingual classifier is evaluated on the summation of predictions from the sentences of each document for each class. Also, both of the monolingual classifiers are evaluated on the summation of the predictions from the sentences of each document of each class but these results are then averaged to be able to compare them with the multilingual classifier.

Classification Algorithm	**Word Embeddings**	**Corpus**	**Type of training**	**Evaluation Strategy**
BiLSTM	Monolingual Domain-Specific (Legal Text)	English	Non Clustered	Document Level & Sentence Level
BiLSTM	Monolingual Domain-Specific (Legal Text)	German	Non Clustered	Document Level & Sentence Level
BiLSTM	Bilingual Domain-Specific (Legal Text)	English and German	Non Clustered	Document Level & Sentence Level

Table 3.7: Classification algorithms for the third research question with their corresponding text representation techniques, the corpus used to train them, the type of training (clustered or non clustered) and the evaluation strategies.

4. Implementation

This chapter describes in detail how the data is collected, which preprocessing techniques are applied to the data in order to make it fit for the machine learning algorithm to learn, how data is resampled, how it is clustered and what the details are of the architecture of the models and which evaluation strategies are applied. This chapter also contains detail about all the tools and libraries that are used for completing the above-mentioned tasks.

4.1 Data Collection

To begin with the implementation, the first step is to obtain the data. Obtaining the data is particularly challenging as these summaries are frequently changing, so scraping the data at different times may result in the different assignment of a single document in different categories. Also, another challenge is that when documents for English and German are scraped separately, it happens quite often that some of the documents from one of the language are missing. Due to these challenges, a carefully crafted scraper which scrapes the data only if the documents from both of the languages are available is used. This, in turn, added a time overhead.

For scraping the data, the Python library *urllib*[1] is used, this library scrapes the HTML data out of the desired web pages. Then Python library *Beautiful Soup*[2] is used on the data collected from the web pages to parse the HTML content and produce a simple text document. During the parsing of the data, the corresponding label of the document is also parsed in order to complete the training dataset. First, the title of the document is scraped. Then all the HTML elements of the web page are removed to construct the text document. A new folder with the name of the category that is the label of the document is then used to store the documents of that class, and each document is given the corresponding title as the name of the file.

The EUR-Lex data set is an imbalanced dataset. That is the number of instances across the classes is not even as shown in Figure 4.1.

[1] https://docs.python.org/3/library/urllib.html
[2] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

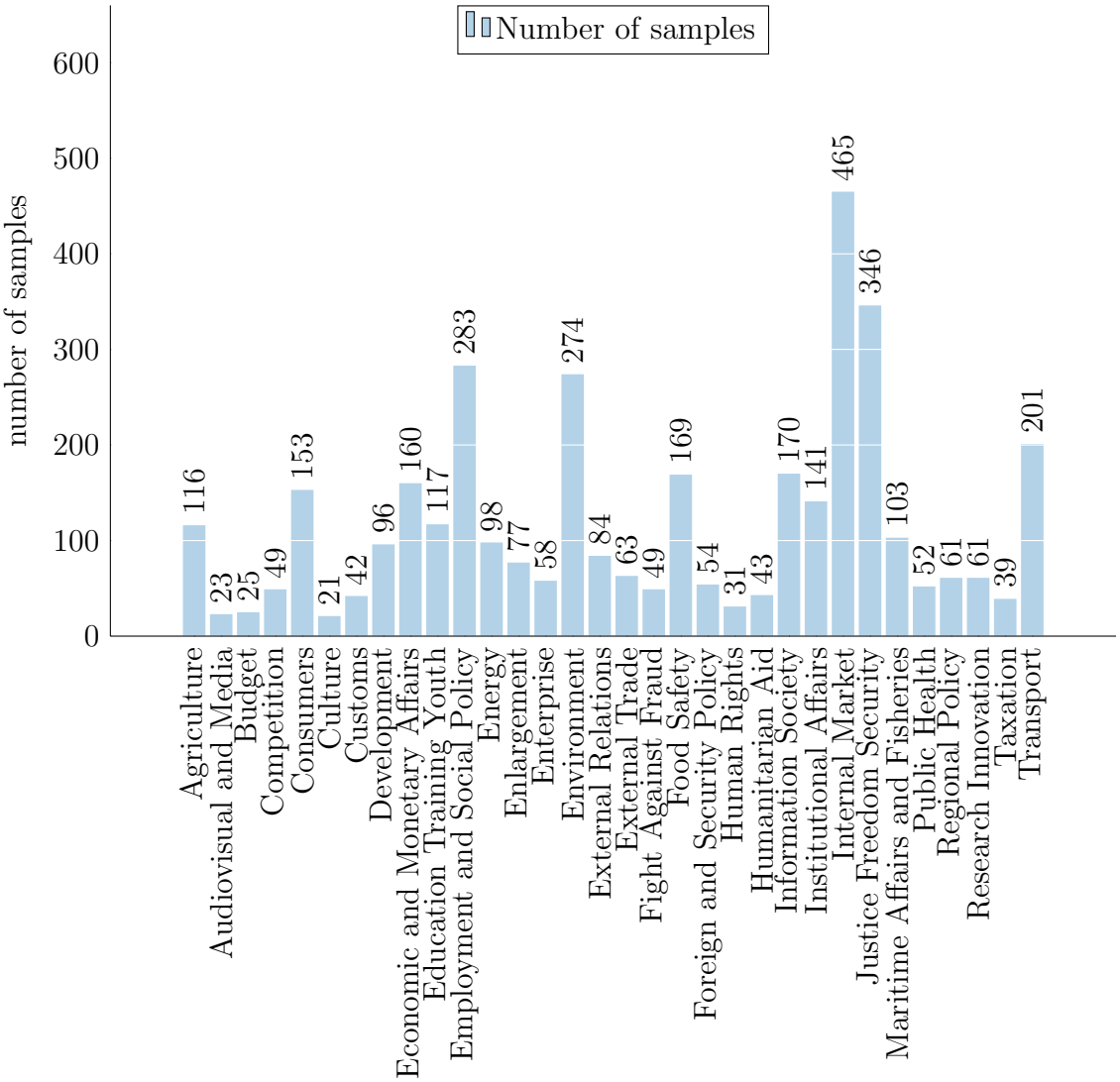


Figure 4.1: Document Distribution of EUR-Lex summaries

4.2 Data Cleaning

The preprocessing involves removing of *punctuation*, *numbers*, *currency symbols* as they do not contribute anything in the classification process. Then the next step is to normalize the text, and this step is necessary because of the *inflection* added due to the modification of words. Lemmatization is used here to normalize the text. As the last step, *stop words* from both the languages are removed because, they also do not contribute anything, and they increase the training time of the algorithm. For the German language, *umlauts - ä, ö and ü* are converted into their base form that is *ä* to *ae*, *ö* to *oe* and *ü* to *ue*. There were other Unicode characters which were also removed.

Following are the steps in which the preprocessing is done.

1. As a first step, *stop words* are removed.
2. The next step is to lemmatize the words.

3. In the next step other unnecessary symbols are removed. For example, § is the symbol of a paragraph and is extensively used in the legal text. The symbol • is also another example.

4. In this step numbers and punctuation are removed.

5. Conversion of umlauts to its base form.

The order of the steps is also essential as it will help in reducing the overload on some of the processes. For example, when the stop words are removed in the first step, then the lemmatizer would not have to go through those words, and that will decrease the time taken to process the text.

For removing the stop words the Python library *NLTK*[3] (Natural Language Toolkit) is used. It provides stop words in both English and German language. To lemmatize the words the *spaCy*[4] library is used for both the languages. To remove unnecessary symbols, custom functions are written. Number removal and conversion of umlauts were also done using custom functions.

4.3 Clustering

To cluster the data using the K-means algorithm, first, the value of k is obtained using the Silhouette score (see Section 2.4.2.1) and Elbow Analysis (see Section 2.4.2.2).

Silhouette score and elbow analysis is computed using the *scikit-learn*[5] library. First the documents from English corpus are converted into TF-IDF vectors using the *scikit-learn* library. These TF-IDF vectors are then used for training 8 k-means cluster which produces the cluster assignments for each value of k. The value 8 is heuristic as having more than 8 clusters would not be feasible in the context of the amount of resources required for training the classifiers for these clusters and also, dividing data into more than 8 clusters would leave less data for classification in those clusters. The cluster assignments are then used along with the TF-IDF matrix to find the silhouette score. The highest value of the silhouette score means better the assignment of the clusters (see Section 2.4.2.1)

The values of the silhouette score obtained for EUR-Lex summaries are listed below in the Table 4.1

And the elbow analysis for the same is shown in the Figure 4.2, which indicates that the data should be divided into 2 clusters. The silhouette score is high at 2 clusters, that means that other values of k will not model the data significantly better.

After obtaining the value of $k = 2$ for the number of clusters, the next step is to apply constrained k-means clustering, for this an open source implementation of constrained k-means is used here called as *COP-Kmeans*[6] [Bab17].

[3]https://www.nltk.org/
[4]https://spacy.io/
[5]https://scikit-learn.org/stable/
[6]https://github.com/Behrouz-Babaki/COP-Kmeans

k	Silhoutte score
2	0.05693
3	0.04417
4	0.04749
5	0.05515
6	0.05480
7	0.05647
8	0.05537

Table 4.1: Silhouette scores for 8 values of k

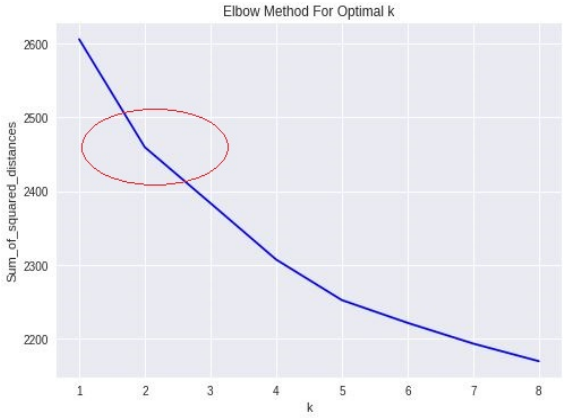


Figure 4.2: Elbow analysis validating the value of k at 2

The COP-Kmeans requires the dataset (TF-IDF matrix of the corpus), the value of k, and two lists of constraints (must-link and cannot-link). As mentioned already, no prior information is available for the cannot-link constraints; hence, only must-link constraints are applied.

The Table 4.2 shows the cluster assignments of all the 32 categories of the EUR-Lex summaries obtained after applying COP-K-means with $k = 2$.

Class Label	Cluster Assignment
Agriculture	1
Audiovisual and Media	1
Budget	2
Competition	1
Consumers	1
Culture	2
Customs	2
Development	2
Economic and Monetary Affairs	2
Education Training Youth	2
Employment and Social Policy	1
Energy	1
Enlargement	2
Enterprise	1
Environment	1
External Relations	2
External Trade	2
Fight Against Fraud	2
Food Safety	1
Foreign and Security Policy	2
Human Rights	2
Humanitarian Aid	2
Information Society	1
Institutional Affairs	2
Internal Market	1
Justice Freedom Security	2
Maritime Affairs and Fisheries	2
Public Health	1
Regional Policy	2
Research Innovation	2
Taxation	1
Transport	1

Table 4.2: Assignment of EUR-Lex summaries into clusters

4.4 Data Resampling

The method behind the resampling of the dataset is to loop over the dataset one by one to find duplicate documents. When a duplicate document is found, we check the labels of all the duplicate documents and compare the total number of samples in all the labels for which the duplicate documents are found. We remove the duplicate instance from the class containing the most number of samples. This method is time consuming as the number of loops the algorithm goes through before finishing is n x n where n is the number of documents.

To make the algorithm more efficient, first, the total number of samples is counted. Then, a list of non-duplicate documents without the labels is created. The generated list is then used to iterate over all the duplicate documents, by comparing every document and storing the corresponding labels of the duplicate document. After a single document from the non duplicated list completes iterating over all the duplicate documents, the labels of the duplicated documents are checked for their class distribution, and then the duplicated document is removed from the majority class. Also, if a duplicate document is found in the same class, then only one is kept, and the other is removed.

The pseudo code for the above-mentioned algorithm where x is a list of non duplicated documents. x_d is the list with duplicated documents and y_d contains the corresponding labels for the x_d as shown in Algorithm 1

Algorithm 1 Resampling Algorithm

```
for all labels 𝒍 in y_d do
    dist ← Counter(𝒍)
end for
for i = 1 : count(x) do
    x_i ← x[i]
    for j = 1 : count(x_d, y_d) do
        x_j ← x_d[j]
        y_j ← y_d[j]
        if x_i == x_j then
            tmp_list ← y_j
        end if
    end for
    if len(tmp_list) == 1 then
        ℓ ← tmp_list
    else
        for k = 1 : count(tmp_list) do
            ℓ ← min(dist[temp_list[k]])
        end for
    end if
    x_i → doc {Resampled document data}
    ℓ → labels {Labels for the resampled document data}
end for
```

Table 4.3 shows the total number of documents in the original dataset and the number of documents after the resampling technique is applied.

Category	Original	Resampled
Agriculture	116	90
Audiovisual and Media	23	18
Budget	25	25
Competition	49	47
Consumers	153	125
Culture	21	21
Customs	42	39
Development	96	63
Economic and Monetary Affairs	160	149
Education Training Youth	117	102
Employment and Social Policy	283	174
Energy	98	74
Enlargement	77	64
Enterprise	58	56
Environment	274	134
External Relations	84	59
External Trade	63	56
Fight Against Fraud	49	31
Food Safety	169	108
Foreign and Security Policy	54	49
Human Rights	31	35
Humanitarian Aid	43	30
Information Society	170	149
Institutional Affairs	141	128
Internal Market	465	234
Justice Freedom Security	346	223
Maritime Affairs and Fisheries	103	90
Public Health	52	52
Regional Policy	61	58
Research Innovation	61	55
Taxation	39	39
Transport	201	139

Table 4.3: Distribution of number of samples before and after resampling

4.5 Training the Word Vectors

The model used in the creation of the word vectors is a neural network, and neural networks are data hungry as it requires a lot of data to train them properly. As a result, another domain-specific dataset is used in this experiment because the data from the summaries used for classification is small. Hence, the whole EUR-Lex dataset was used as it contains 19,348 documents mostly consisting regulations, decisions and directives of European Union [LMF10].

4.5.1 Training Cross Lingual Word Embedding

We are the technique suggested by Duong et. al [DKM+16] (see Section 2.9.3) to train bilingual domain-specific word embeddings. The dataset used for training the domain-specific word embeddings is the EUR-Lex dataset, as the dataset used for classification (EUR-Lex summaries) is less compared to the EUR Lex dataset. For creating a bilingual dictionary, the *EuroVoc* thesaurus [SPH02] provided by the Publication office of the European Union is used. It is available in 24 official languages recognized by the European Union. This thesaurus is domain-specific hence it does not involve common words that might occur in the documents. Due to this reason, this thesaurus is combined with bilingual dictionaries that are available from *Facebook MUSE* [CLR+17]. These bilingual dictionaries were created using Facebook's internal translation tools. The authors claim that these dictionaries handle the polysemy of the words better then other bilingual dictonaries available. This combined bilingual dictionary with the original EUR-Lex dataset is used to create the legal domain-specific bilingual word embeddings.

Monolingual word embeddings for training BiLSTMs are trained using the *Skip-gram* method with Fasttext[7][GLF+09] tool. These embeddings were also trained on the the EUR-Lex corpus. These monolingual word embeddings are in English and German language and are in 300 dimensions.

For training general-purpose bilingual word embeddings we again Fasttext which provides word embeddings in 157 languages trained on Wikipedia[8] and Common Crawl[9] data using the Continuous bag-of-words model in 300 dimensions. The word embeddings for multiple languages cannot be combined together because they belong in different vector spaces. Facebook's MUSE[10] [CLR+17] with the help of parallel dictionaries is used to align both word embeddings (English and German) into a single vector space. Upon alignment, both the embeddings can be simply concatenated and can be used for classification purpose. These word embeddings are created from various sources, it contains text from various domains and hence they are being used as general-purpose word embeddings.

4.6 Architecture and Training

This section describes in detail the specifics of the architectures of the algorithms used for all the three questions mentioned in Section 3.3.

Before beginning to train the models, it is necessary to divide the dataset into a training set and a testing set. Generally, 70% of the data is for training purposes, and 30% to evaluate the model. This division is stratified, which means that the class distribution in training and testing set will remain almost equal. The stratified division ensures that there are enough examples for each class in training, and test phase of the model.

[7]https://fasttext.cc/docs/en/cr awl-vectors.html
[8]https://www.wikipedia.org/
[9]http://commoncrawl.org/
[10]https://github.com/facebookresearch/MUSE

Neural network-based classification models are prone to *overfitting* [Pre98]. An overfitted model will perform well on the training data but fail to perform well on the testing data. To put it simply, while training the error on the training set will keep on decreasing with, the error on the unseen data starts getting worse. To avoid this situation regularization techniques such as Dropout, L1 and L2 regularization and early stopping are used.

Dropout and L1, L2 regularization are explicit regularization techniques, that is they explicitly reduce the model complexity. In Dropout regularization techniques, during the training, a fraction of randomly selected neurons are ignored. Hence, their activation contribution to the forward pass and weight updates are removed. While Dropout regularization reduces the model complexity by removing randomly selected neurons during training, L1 and L2 regularization reduce the model complexity by imposing a penalty to the weight of all the features and features with non zero weights respectively.

On the other hand, the early stopping regularization is implicit which does not affect the complexity of the model directly [ZBH+16]. Early stopping monitors the training process and halts it once the performance of the model starts degrading. It does so by monitoring the performance of the classifier on the validation data.

Dropout, L1 - L2 regularization, and early stopping can be configured in *keras* while building the model. During the training, we split 30% of training data for validation purpose so that there is no need for splitting validation data at the time when we split the dataset into training and testing set.

4.6.1 First Research Question

In the first research question SVMs are compared to BiLSTMs. When using an SVM algorithm, we do not know beforehand the value of C. To find that out, we need some kind of selection mechanism. The aim is to find out the value of C such that it can predict accurately on unknown data. One method of finding the value of C is called *Grid Search Cross Validation* (Grid Search CV). This method is straight forward; first, we select the number of values of C, then we divide the training set into n equal-sized subsets. Then we train the SVM using one value from C on the $n-1$ subset and predict on the one left subset, we repeat this process until every instance of the training set is predicted once for every value of C.

This process is computationally expensive as each subset will be trained on every value of C. We used the following values of C (0.001,0.01, 0.1,1,10,100).

To train the SVM, a TF-IDF matrix of the English corpus is created using the *scikit-learn* python library and then this TF-IDF matrix is used to train a linear SVM from the same *scikit-learn* library.

The SVM is compared against two BiLSTM models, one without clustered data on single language (English) and one with clustered data (English and German). The technique used for training both SVM and BiLSTM is different, while SVM is trained on whole documents, BiLSTMs are trained on sentences created using the sliding window technique (see Section 2.11, Section 3.3.1). The Figure 4.3 shows the

architecture of the BiLSTM model trained on English corpus with all the 32 classes. The BiLSTM models are created using the *Keras*[11] Python library.

The hyperparameters for the BiLSTM trained on the English corpus without clustered data is listed in Table 4.4.

Hyperparameter	Value
Sentence Size	30 words
Batch Size	32
Embedding Dimension	200
Hidden 1 Size	40
Hidden 2 Size	40
Dropout 1	0.5
Dropout 2	0.5
$l2$ regularization 1	0.04
$l2$ regularization 2	0.01
Learning Rate	0.001

Table 4.4: Hyperparameter of BiLSTM for training English language corpus without clustered data

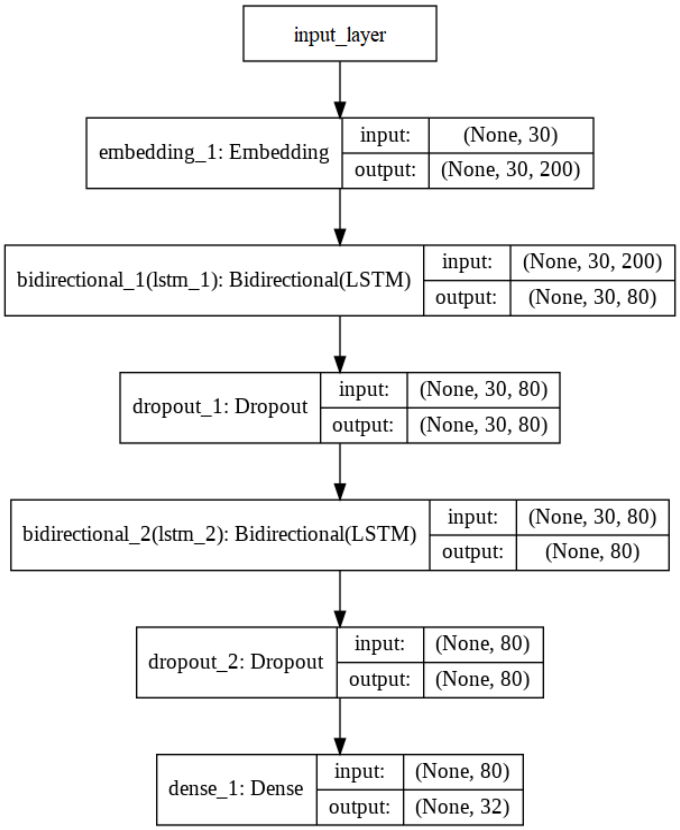


Figure 4.3: Architecture of BiLSTM for training English language corpus without clustered data

[11] https://keras.io/

The hyperparameters for training the BiLSTMs on clustered data are exactly the same to the one without clustered data trained on English language Table 4.4.

The architecture for the BiLSTM trained on cluster 1 and cluster 2 data is shown in Figure 4.4.

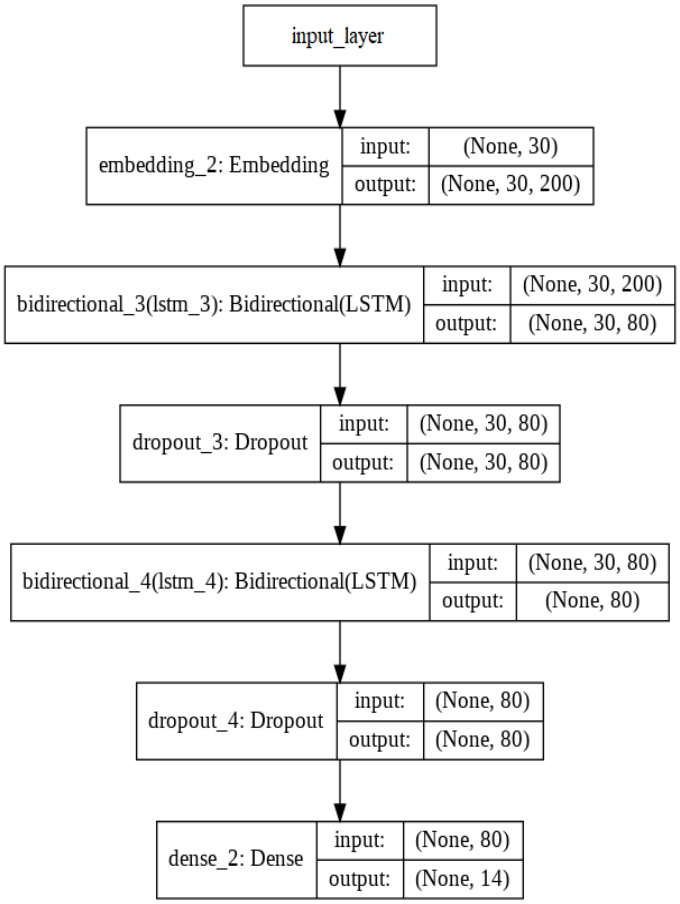


Figure 4.4: Architecture of BiLSTM for training English and German language corpus on cluster 1 and cluster 2 data.

4.6.2 Second Research Question

In the second question general-purpose embeddings are compared to word embeddings created using the EUR-Lex corpus (see Section 4.5.1).

Both the approaches using the general-purpose resource and the domain-specific resources, use a BiLSTMs trained on clustered data to see the effect of clustering, similar to the BiLSTMs in Section 4.6.1 The hyperparameters for both approaches is listed in Table 4.5 for general-purpose embeddings and Table 4.6 for domain-specific embeddings.

Hyperparameter	Value
Sentence Size	30 words
Batch Size	32
Embedding Dimension	300
Hidden 1 Size	40
Hidden 2 Size	40
Dropout 1	0.5
Dropout 2	0.5
$l2$ regularization 1	0.04
$l2$ regularization 2	0.01
Learning Rate	0.001

Table 4.5: Hyperparameters of the BiLSTM for training on English and German language with clustered data using general-purpose word embeddings created using Facebook's MUSE python library

Hyperparameter	Value
Sentence Size	30 words
Batch Size	32
Embedding Dimension	200
Hidden 1 Size	40
Hidden 2 Size	40
Dropout 1	0.5
Dropout 2	0.5
$l2$ regularization 1	0.04
$l2$ regularization 2	0.01
Learning Rate	0.001

Table 4.6: Hyperparameter of the BiLSTM for training on English and German language with clustered data using domain-specific word embeddings created from EUR-Lex dataset

The only difference between the training parameters of both the approaches are the embedding dimensions; the general-purpose embedding is in 300 dimensions whereas the domain-specific embedding is in 200 dimensions. The reason for this difference is that aligning two embeddings in a single vector space is a computationally inexpensive task, then learning the embedding from scratch for two languages. Decreasing

the dimensions will result in the reduction of training time, and hence the dimension of domain-specific word embeddings are 100 less than general-purpose embeddings.

The architecture for the BiLSTM trained using general-purpose and domain-specific word embeddings is shown in Figure 4.5

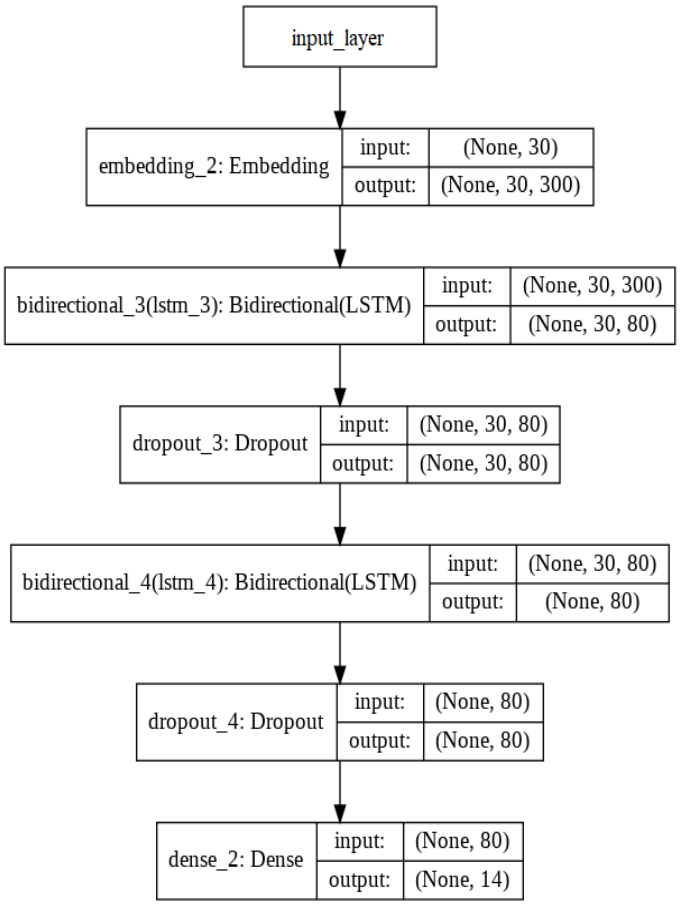


Figure 4.5: Architecture of the BiLSTM for training on an English and German language corpus on clustered data using general-purpose word embeddings aligned using Facebook's MUSE.

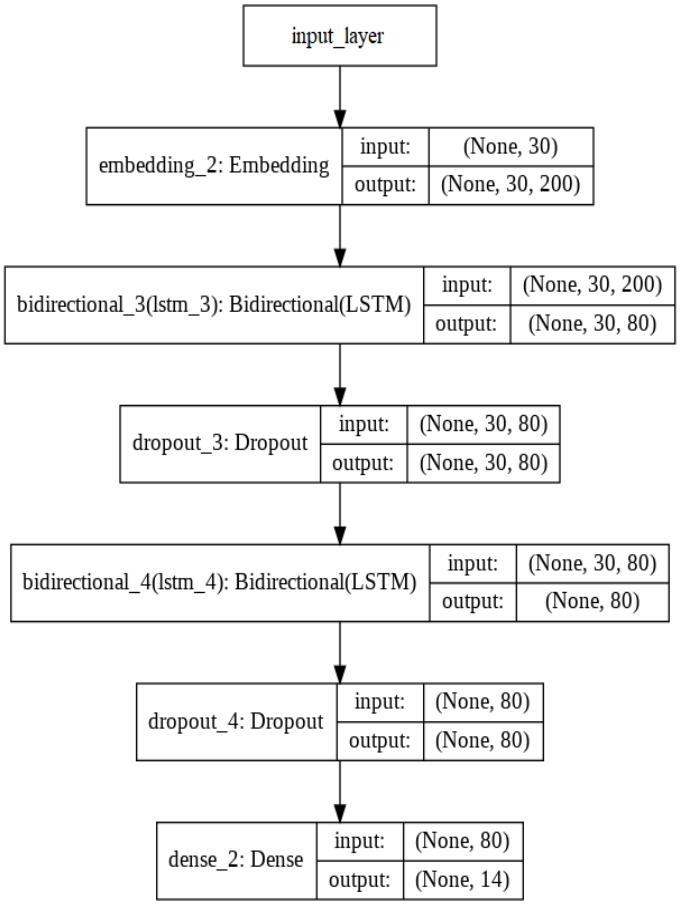


Figure 4.6: Architecture of the BiLSTM for training on an English and German language on clustered data using domain specific embeddings trained on the EUR-Lex dataset.

For evaluating the predictions, each document is divided into sentences using the same sliding window technique used to create the training data, then predictions are stored for every sentence of a document, and the predictions are summed up, and the class with the highest value is considered the predicted class of the document.

4.6.3 Third Research Question

The third question evaluates the capability of training multiple languages in a single BiLSTM model compared to training different models for different languages. For this purpose two models, one for the English and another for the German language will be compared to a single model trained on both the English and the German language together.

The hyperparameters and architecture for all the models are the same to ensure a fair comparison. The hyperparameters are for this question listed in Table 4.7 and the architecture for this question is shown in Figure 4.7.

Hyperparameter	Value
Sentence Size	30 words
Batch Size	32
Embedding Dimension	200
Hidden 1 Size	40
Hidden 2 Size	40
Dropout 1	0.5
Dropout 2	0.5
$l2$ regularization 1	0.04
$l2$ regularization 2	0.03
Learning Rate	0.001

Table 4.7: Hyperparameters of the BiLSTM for training on the English and the German language without clustered data and trained on both languages separately and together.

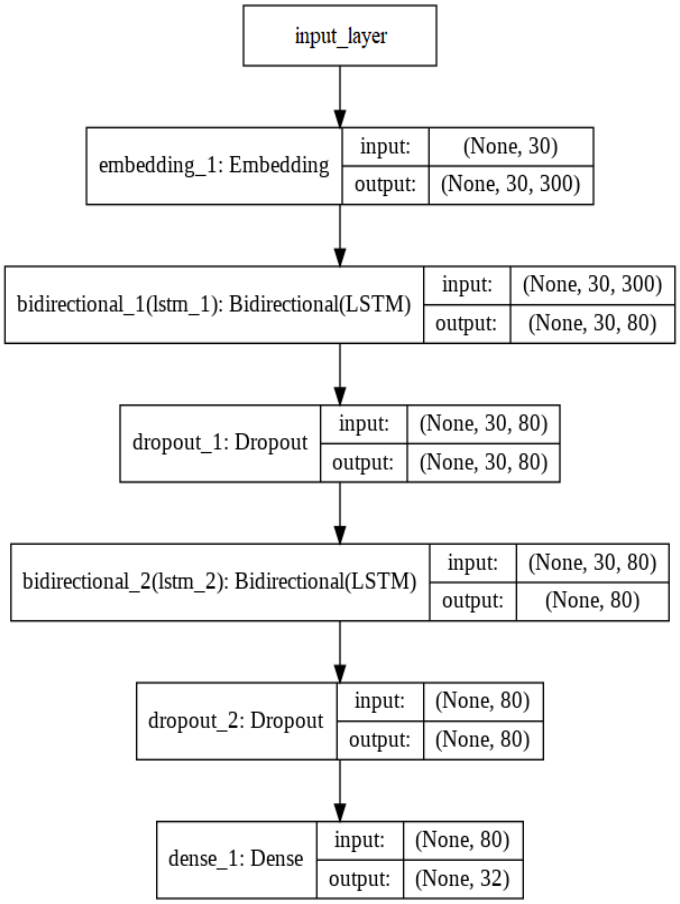


Figure 4.7: Architecture of the BiLSTM for training on the English and the German language without clustered data and trained on both languages together and separately.

5. Evaluation

Evaluating a classification algorithm also known as classification model is necessary to find how competent the algorithm is on data it has never seen. The classification model generates probabilities when predicting unobserved data.

5.1 Experimental Setup

Most of the experiments are carried out on *Google Colab* which is a free Google service that provides Jupyter notebooks with Python environment that stores the notebooks on Google Drive. It provides a GPU (Graphics Processing Unit) or TPU (Tensor Processing Unit) and has pre-installed various popular machine learning and deep learning frameworks. The exact detail of the system is listed in the Table 5.1.

Hardware	**Specifications**
CPU	2vCPU Intel(R) Xeon(R) Processors @2.20Ghz
GPU	1xTesla K80 12GB(11.439GB Usable) GDDR5 VideoRAM
TPU	Google's custom developed application-specific integrated circuits
RAM	12GB
DISK	358.27 GB

Table 5.1: Hardware specification for the experimental setup

All the experiments are performed using *Python 3* environment. *Scikit-learn*, an open source machine learning library in Python is used here for data processing and training the SVM. For the BiLSTMs, *Keras* an open source neural network library written in Python is used with a *Tensorflow* backend.

The packages, their version numbers and the description of the packages are listed below in Table 5.2

Package Name	Version Number	Description
scikit-learn	0.20.3	An open source machine learning library for data mining and data analysis.
keras	2.2.4	An open source neural network library for fast prototyping, uses likes of Tensorflow or Theano.
tensorflow	1.13.1	An open source software library for numerical computation using data flow graphs.
beautifulsoup4	4.6.3	A python library for parsing HTML and XML documents.
matplotlib	3.0.3	A python library for plotting and visualization.
nltk	3.2.5	A python library for statistical Natural Language Processing.
seaborn	0.7.1	A python library for statistical data visualization.
spacy	2.0.18	An open-source software library for advanced Natural Language Processing.
numpy	1.14.6	An python library for creating and manipulating, large and multiple dimensional arrays.
scipy	1.1.0	An python library for scientific and technical computing.
Fasttext	0.2.0	An python library for training word vectors.
MUSE	NA	A library for aligning word vectors into a single vector space.
XlingualEmb	NA	A library for learning bilingual word vectors.

Table 5.2: List of packages used

5.2 Evaluation Approach

For evaluation, initially 70% of data is used for training the classification models, and then 30% of the data is used for evaluating it. The evaluation is done on sentence level and on document level Section 3.3.1 using performance evaluation matrices described in Section 2.12. The evaluation of each question will go in detail about the performance of each classification model. Beside the described performance evaluation matrices, as the dataset used for training is an imbalanced one, the evaluation

will also go into the details about the performance of the classifiers on each class in order to get better insights of how well it is performing for the underrepresented classes (the minority classes).

5.2.1 Evaluation for the first research question

The first research question compares the performance of SVM and BiLSTM trained on the English corpus. It also evaluates the classification performance of the BiLSTM trained on unclustered bilingual corpora compared with the one trained on clustered bilingual corpora.

Before training the SVM, we had to find the value of C, which we found to be 1 among all other values.

The Figure 5.1 shows the Micro-average and Figure 5.2 shows the Macro-average performance of different classifiers evaluated on sentence and document level. The results of the algorithm trained on clustered data are averaged together as mentioned in Section 3.3.1

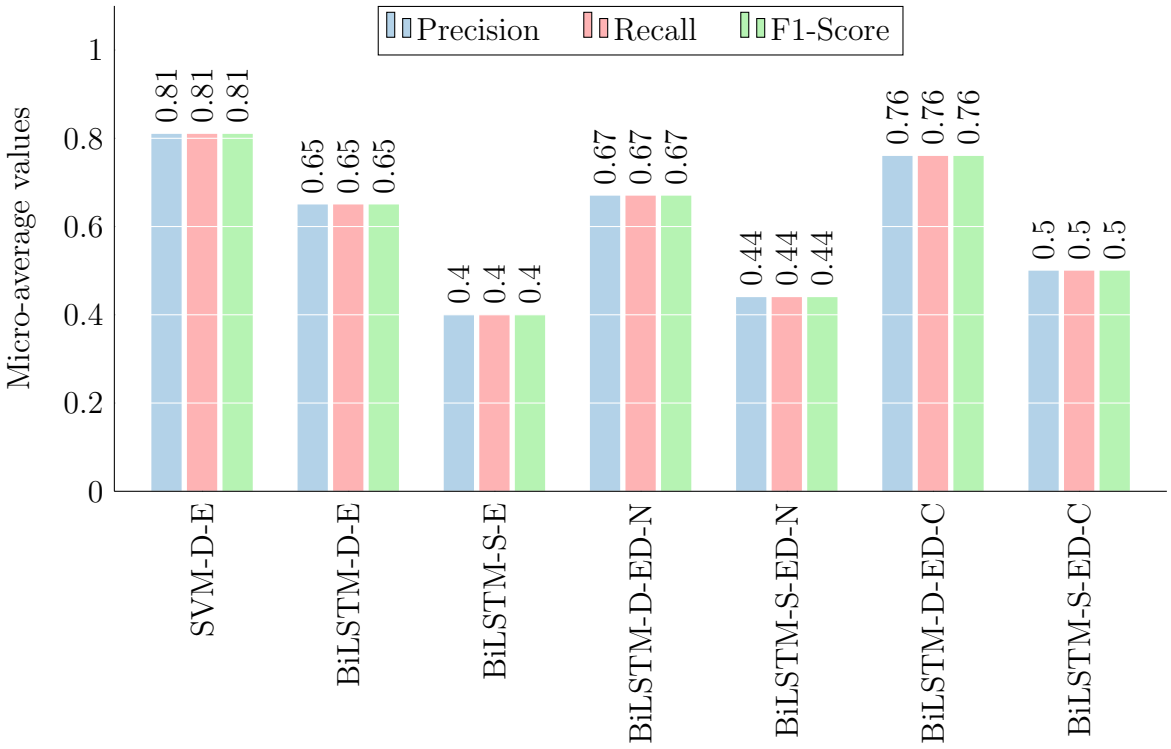


Figure 5.1: Micro-averaged *Precision*, *Recall* and *F1-Score* of SVM and BiLSTM in different configurations. The first suffix D or S indicates evaluation on document or sentence level respectively, the second suffix E or D represents the language of the corpus used respectively. The third suffix N or C indicates non clustered and clustered respectively.

Figure 5.1 shows that the performance of the SVM is better than BiLSTM when trained only on English corpus on document level. The result of the same BiLSTM when evaluated on a sentence level is less compared to when evaluated on document level. The results on the sentence level are significantly lower compared to the results

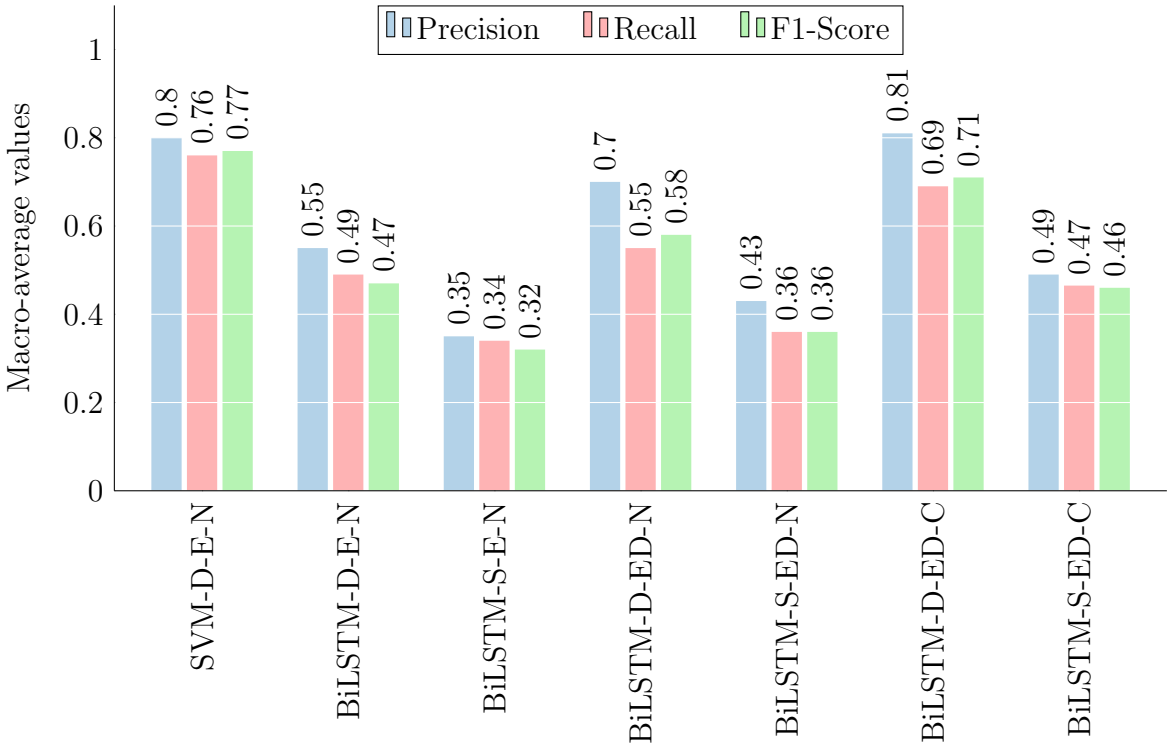


Figure 5.2: Macro-averaged *Precision*, *Recall* and *F1-Score* of SVM and BiLSTM in different configurations. The first suffix D or S indicates evaluation on document or sentence level respectively, the second suffix E or D represents the language of the corpus used respectively. The third suffix N or C indicates non clustered and clustered respectively.

on the document level. This behavior can be further explained by considering an example document which is categorized in class *A* and contains three sentences. As described in the Section 2.11 all the three sentences of this document will have the same class as the document.

Sentence Identifier	**Prediction Score of each class, [Class A, Class B]**	**True class**	**Predicted class**
Sentence I	[0.9, 0.1]	A	A
Sentence II	[0.49, 0.51]	A	B
Sentence III	[0.45, 0.55]	A	B
Summation of prediction score for the whole document (normalized)	[0.62 , 0.38]	A	A

Table 5.3: Document with three sentences, prediction score for each class, true class and predicted class

From the Table 5.3, we can see that the predicted score of *Sentence I* for class *A* is 0.9 and for class *B* is 0.1, hence it is predicted in class A but for *Sentence B* and *Sentence C* although they belong to class *A* they are predicted in class *B* since the classifier was uncertain about the decision. However, when we combine the predicted score for all the classes from all the sentences and normalize them, the overall score for the document is higher for class *A*, so the precision of the above classifier on sentence level will be $\frac{1}{3} = 0.33$ and on document level the precision is $\frac{1}{1} = 1$ and therefore when evaluating on document level, the performance matrices are better.

The Table 5.4 shows the class-wise *precision, recall* and *F1-Score* values for BiLSTM trained on English and German corpus with clustering and without clustering.

The effect of clustering is also evident from Figure 5.1 and Figure 5.2. The specialization advantage of clustering improves the classification performance on the document level as well as on the sentence level. Also, we can see that the micro-average f1-score for non clustered data in BiLSTM on the bilingual corpus is higher than macro-average values, this indicates that the performance of the classifier on minority classes on clustered data is better than non clustered data. This can be confirmed by looking at the class-wise precision, recall and f1-score values in the Table 5.4 for class *Audiovisual and Media* and *Culture* which has 18 and 21 instances, respectively (see Table 4.3).

The micro-average values for precision, recall and f1-score in Figure 5.1 are the same for each classifier. Hence, the calculation of the micro-average precision, recall and f1-score for BiLSTM-D-ED-C cluster 1 is done in Section A.2 to confirm that it is indeed the same.

Category	P_{LSTM}	R_{LSTM}	F_{LSTM}	P_{LSTM-C}	R_{LSTM-C}	F_{LSTM-C}
Agriculture	0.74	0.78	0.76	0.90	0.86	0.88
Audiovisual and Media	0.00	0.00	0.00	1.00	0.10	0.18
Budget	0.90	0.45	0.60	0.78	0.70	0.74
Competition	0.90	0.63	0.75	0.96	0.83	0.89
Consumers	0.54	0.54	0.54	0.59	0.65	0.62
Culture	0.00	0.00	0.00	0.93	1.00	0.97
Customs	0.78	0.47	0.58	0.64	0.70	0.67
Development	0.45	0.81	0.57	0.64	0.83	0.72
Economic and Monetary Affairs	0.85	0.93	0.89	0.95	0.87	0.91
Education Training Youth	0.64	0.94	0.77	0.86	0.94	0.90
Employment and Social Policy	0.68	0.83	0.75	0.71	0.88	0.79
Energy	0.71	0.71	0.71	0.97	0.64	0.77
Enlargement	0.70	0.44	0.54	0.76	0.59	0.67
Enterprise	0.44	0.15	0.23	0.65	0.42	0.51
Environment	0.69	0.82	0.75	0.70	0.84	0.76
External Relations	1.00	0.23	0.37	0.92	0.55	0.69
External Trade	0.53	0.61	0.57	0.61	0.71	0.66
Fight Against Fraud	1.00	0.25	0.40	0.53	0.50	0.52
Food Safety	0.90	0.82	0.86	0.93	0.82	0.87
Foreign and Security Policy	0.65	0.54	0.59	0.62	0.83	0.71
Human Rights	1.00	0.12	0.22	0.59	0.71	0.64
Humanitarian Aid	1.00	0.29	0.44	0.67	0.57	0.62
Information Society	0.54	0.72	0.62	0.71	0.84	0.77
Institutional Affairs	0.62	0.60	0.61	0.67	0.65	0.66
Internal Market	0.62	0.70	0.66	0.72	0.75	0.74
Justice Freedom Security	0.53	0.86	0.66	0.81	0.67	0.74
Maritime Affairs and Fisheries	0.91	0.80	0.85	0.94	0.91	0.92
Public Health	1.00	0.39	0.56	0.86	0.43	0.58
Regional Policy	0.88	0.50	0.64	0.72	0.86	0.78
Research Innovation	0.71	0.43	0.53	0.60	0.96	0.74
Taxation	0.94	0.54	0.68	0.95	0.71	0.82
Transport	0.67	0.81	0.73	0.76	0.86	0.81

Table 5.4: Class-wise precision (P) and recall (R) and F1-Score (F) for the BiLSTM (denoted as LSTM for readability) trained on English and German corpus evaluated on document level. The suffix C indicates the results for clustered data. The best *precision*, *recall* and *f1-scores* among both the classifiers is highlighted in blue, red and green respectively. If the values across both the classifiers are same it not highlighted.

5.2.2 Evaluation for the second research question

In the second research question the effects of general-purpose resources such as word embeddings for the classification of the legal domain-specific task are investigated. All the classification models are trained on English and German Corpora with clustering.

The Figure 5.3 show the micro-average precision, recall, and f1-score values and Figure 5.4 shows the macro-average precision, recall and f1-score values of different hierarchical classifiers trained on English and German bilingual corpora and evaluated on sentence and document level.

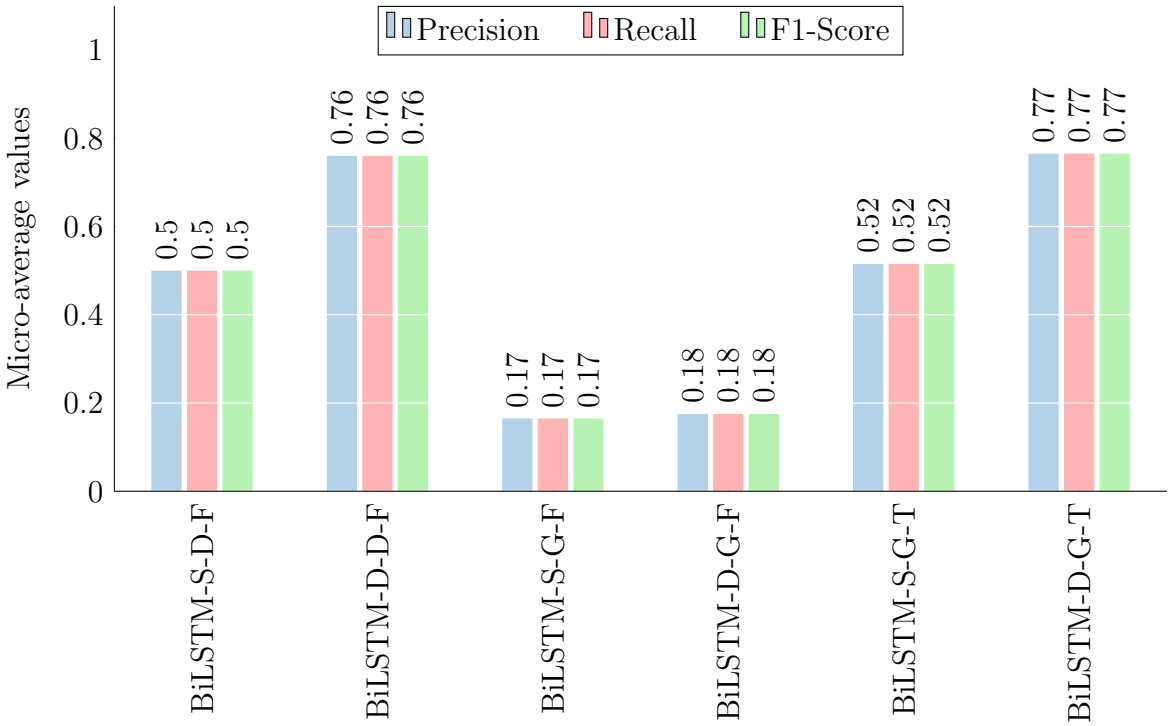


Figure 5.3: Micro-average *Precision*, *Recall* and *F1-Score* of the BiLSTM trained with general-purpose embeddings and domain-specific embeddings. The first suffix S or D indicates the evaluation on sentence or document level, the second suffix D or G represents the domain-specific or general-purpose word embeddings used in the models. The third suffix F or T indicates the status of embedding training, F represents Frozen and T stands for Trainable.

From Figure 5.3 and Figure 5.4 it is quite evident that domain-specific word embeddings perform notably better than the general-purpose word embeddings. BiLSTM-D-ED-C-F is the model with a domain-specific embedding and has the embedding layer frozen, which means that the embedding layer's weights are not updated during the training of the network. BiLSTM-G-ED-C-F is the model with frozen general-purpose word embeddings, similar to the previous model; the weights of this model are not updated. BiLSTM-G-ED-C-T is the model with trainable general-purpose word embeddings, unlike the previous two models, in this model we allow the model

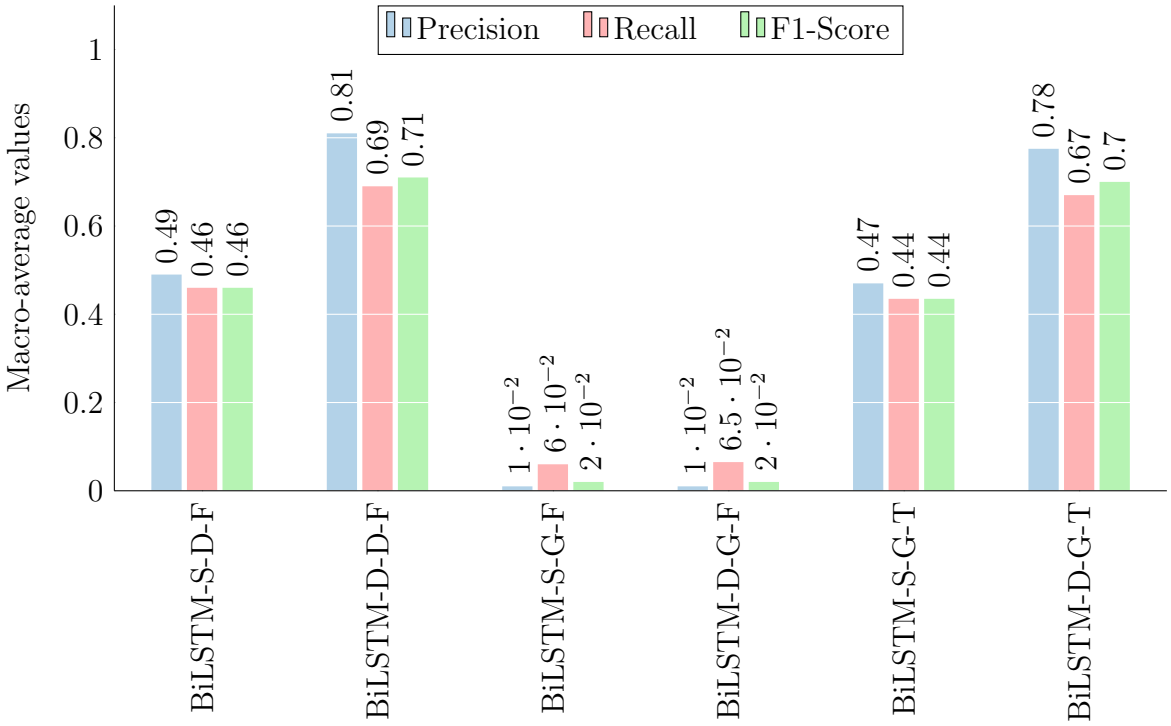


Figure 5.4: Macro-average *Precision*, *Recall* and *F1-Score* of BiLSTM trained with general-purpose embeddings and domain-specific embeddings. The first suffix S or D indicates the evaluation on sentence or document level, the second suffix D or G represents the domain-specific or general-purpose word embeddings used in the models. The third suffix F or T indicates the status of embedding training, F represents Frozen and T stands for Trainable.

to update the weights of the embeddings. This was necessary to evaluate as the general-purpose embeddings are trained on text from sources like Wikipedia or news article, which are not as complex in style and structure compared to legal text. This is the reason as to why it outperformeds the model whose embedding weights were frozen.

The classification models BiLSTM-S-G-F and BiLSTM-D-G-F shown in Figure 5.3 and Figure 5.4 performed the worst. They overfitted on the data and did predicted everything to a single class. One of the reasons for this performance could be that the embedding matrix was never updated. As mentioned above that the general-purpose word embeddings use a free text corpus such as Wikipedia and News articles, which contains information about multiple domains. Hence, when these embeddings were trained on Wikipedia and the News article corpus, the semantics captured are different from the ones that are specifically trained on the legal corpus. One example of this can be seen in an Image Recognition task when applying transfer learning; it is advisable that the dataset for prediction or classification task being performed should be similar to the ones we are doing transfer learning from [IS18].

Another reason is that general-purpose word embeddings do not contain specialized words that are used in the legal domain. The EUR-lex summaries of English and

German corpus for cluster 1 data contain 43339 words and when the embedding matrix is created only 20310 words are found in the general-purpose embedding. That is only half of the words. So training the general-purpose embeddings is necessary to get better performance which can be seen from the Figure 5.4.

The visualization randomly selected 10 words of general-purpose embeddings before training (which is the visualization of frozen word embeddings) and after training (which is a visualization of trained word embeddings) is in Section A.3.

The results are comparable. The Table 5.5 shows in detail the class-wise precision, recall, and f1-score of BiLSTM trained on English and German corpus with the general-purpose word embeddings and on the domain-specific word embeddings.

As we can see from the Table 5.5 the performance of domain-specific word embeddings is better in general compared to the general-purpose embeddings. For the class *Audiovisual and Media*, general-purpose embeddings do not classify a single document.

Category	P_{LSTM-D}	R_{LSTM-D}	F_{LSTM-D}	P_{LSTM-G}	R_{LSTM-G}	F_{LSTM-C}
Agriculture	0.90	0.86	0.88	0.82	0.80	0.81
Audiovisual and Media	1.00	0.10	0.18	0.00	0.00	0.00
Budget	0.78	0.70	0.74	0.67	0.20	0.31
Competition	0.96	0.83	0.89	0.89	0.83	0.86
Consumers	0.59	0.65	0.62	0.65	0.65	0.65
Culture	0.93	1.00	0.97	1.00	0.50	0.67
Customs	0.64	0.70	0.67	0.94	0.53	0.68
Development	0.64	0.83	0.72	0.64	0.81	0.72
Economic and Monetary Affairs	0.95	0.87	0.91	0.90	0.95	0.93
Education Training Youth	0.86	0.94	0.90	0.91	0.94	0.92
Employment and Social Policy	0.71	0.88	0.79	0.71	0.86	0.78
Energy	0.97	0.64	0.77	0.98	0.71	0.82
Enlargement	0.76	0.59	0.67	0.65	0.62	0.63
Enterprise	0.65	0.42	0.51	0.64	0.35	0.45
Environment	0.70	0.84	0.76	0.80	0.89	0.84
External Relations	0.92	0.55	0.69	0.68	0.59	0.63
External Trade	0.61	0.71	0.66	0.72	0.46	0.57
Fight Against Fraud	0.53	0.50	0.52	0.83	0.31	0.45
Food Safety	0.93	0.82	0.87	0.84	0.87	0.85
Foreign and Security Policy	0.62	0.83	0.71	0.79	0.62	0.70
Human Rights	0.59	0.71	0.64	0.89	0.33	0.48
Humanitarian Aid	0.67	0.57	0.62	1.00	0.57	0.73
Information Society	0.71	0.84	0.77	0.76	0.90	0.82
Institutional Affairs	0.67	0.65	0.66	0.55	0.80	0.65
Internal Market	0.72	0.75	0.74	0.74	0.80	0.77
Justice Freedom Security	0.81	0.67	0.74	0.69	0.89	0.78
Maritime Affairs and Fisheries	0.94	0.91	0.92	0.87	0.89	0.88
Public Health	0.86	0.43	0.58	0.92	0.52	0.67
Regional Policy	0.72	0.86	0.78	0.81	0.90	0.85
Research Innovation	0.60	0.96	0.74	0.79	0.79	0.79
Taxation	0.95	0.71	0.82	0.95	0.64	0.77
Transport	0.76	0.86	0.81	0.82	0.89	0.85

Table 5.5: Class-wise precision (P) and recall (R) and F1-Score (F) for BiLSTM-D-ED-C-T (represented with suffix LSTM-D) and BiLSTM-D-ED-C-T (represented with suffix LSTM-G) on evaluated on the document level. The best *precision*, *recall* and *f1-scores* among both the classifiers are highlighted in blue, red and green respectively. If the values across both the classifiers are the same they are not highlighted.

5.2.3 Evaluation for third research question

The third research question investigates the effect of having a multilingual parallel corpus on the classification task. A multilingual parallel corpus here means having different language documents for a predefined category.

As described in Figure 3.3 two BiLSTMs classification models, one trained with the English corpus and other trained with the German corpus are compared with a single BiLSTM classification model trained on an English and German corpus.

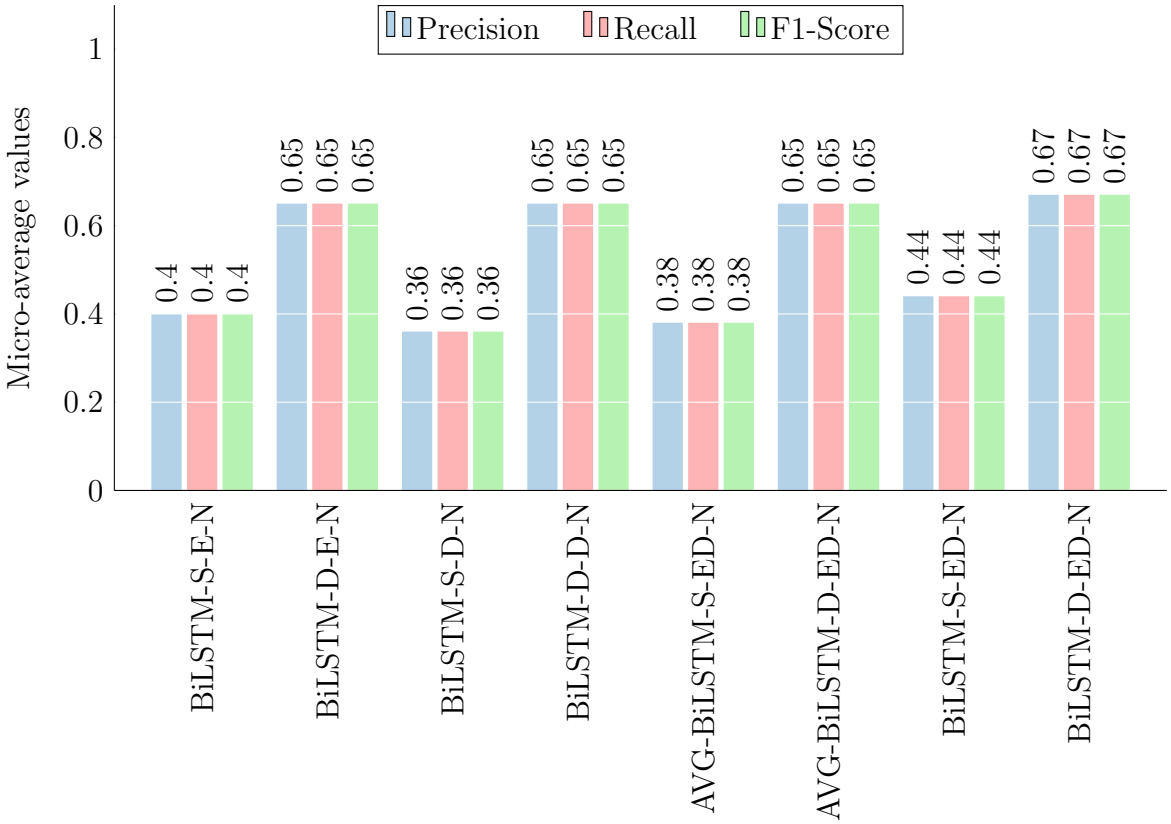


Figure 5.5: Micro-average *Precision*, *Recall* and *F1-Score* for the BiLSTM. The first suffix specifies the method of evaluation (S = Sentence and D = Document), the suffix second E, D or ED specifies the language of the corpus, English, German or both English and German respectively. The second suffix N represents that model is trained on non clustered data. AVG-BiLSTM-ED-N represents the average score from BiLSTM-E-N and BiLSTM-D-N

Figure 5.5 shows the micro-average precision, recall, and f1-scores indicate that employing a single classifier for the bilingual corpus is slightly better than two different classifiers for two different languages. Figure 5.6 also confirms the above mentioned statement. One explanation for the better performance of the bilingual classifier is that it has more examples to learn from than the previous case. C.-P. Wei et al. showed that polylingual data could be used to increase the performance of a classifier [WYL+14].

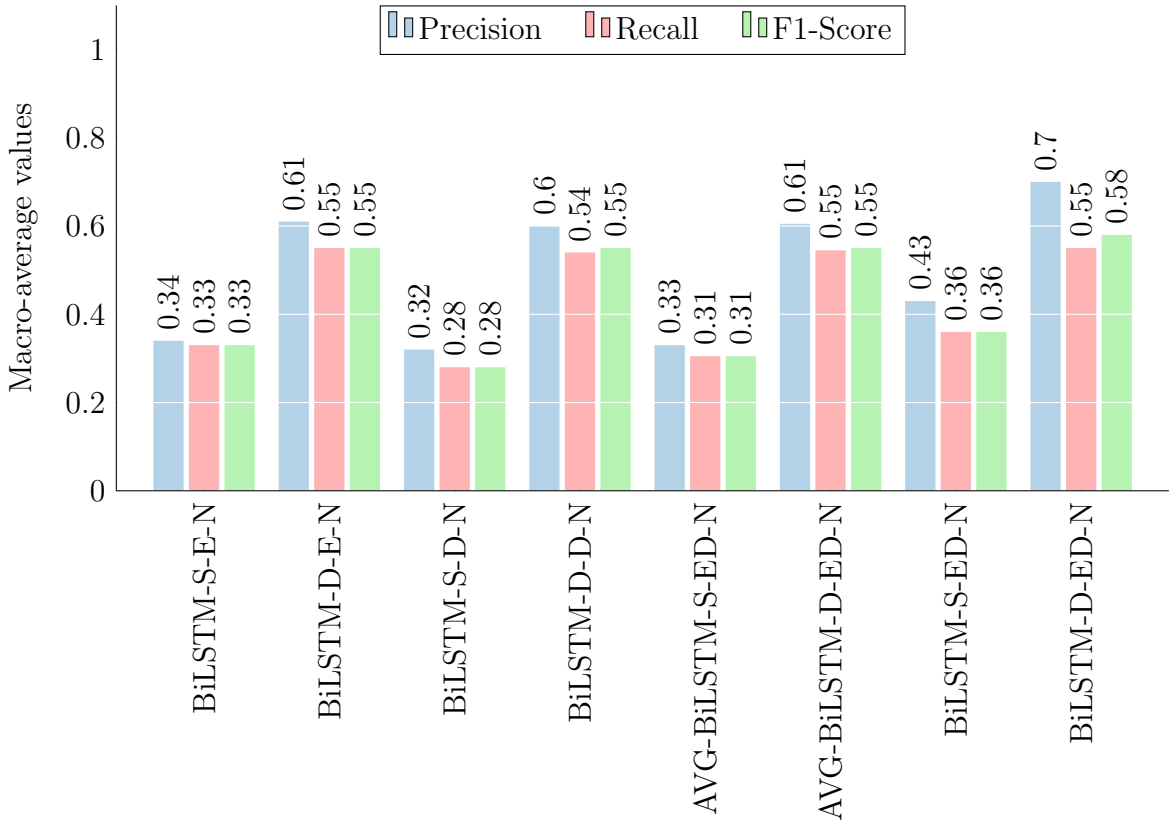


Figure 5.6: Macro-average *Precision*, *Recall* and *F1-Score* for BiLSTM. The first suffix specifies the method of evaluation (S = Sentence and D = Document), the suffix second E, D or ED specifies the language of the corpus, English, German or both English and German respectively. The second suffix N represents that model is trained on non clustered data. AVG-BiLSTM-ED-N represents the average score of BiLSTM-E-N and BiLSTM-D-N

The class-wise precision, recall, and f1-scores of classifiers AVG-BiLSTM-ED-N and BiLSTM-ED-C are shown in Table 5.6

The effects of having multiple languages for training are clear from the Table 5.6. For classes *Budget* the classifier with a single language in training did not predict a single sample during testing. However, when another language is added to the training data, classification shows signs of improvement. The improvement is significant in case of class *Budget* when for a classifier with a single language the precision, recall and f1-score was 0.00 and for bilingual classifier the precision, recall, and f1-score is 0.90, 0.45 and 0.60, respectively. However, for the class *Culture* this is opposite, the bilingual classifier was unable to predict any sample from that class whereas on the other hand the monolingual classifiers did classify with precision, recall and f-score values of 0.57, 0.62, 0.60.

Category	P_{LSTM-A}	R_{LSTM-A}	F_{LSTM-A}	P_{LSTM-B}	R_{LSTM-B}	F_{LSTM-B}
Agriculture	0.65	0.82	0.72	0.74	0.78	0.76
Audiovisual and Media	0.00	0.00	0.00	0.00	0.00	0.00
Budget	0.00	0.00	0.00	0.90	0.45	0.60
Competition	0.50	0.16	0.25	0.90	0.63	0.75
Consumers	0.57	0.62	0.60	0.54	0.54	0.54
Culture	0.61	0.57	0.53	0.00	0.00	0.00
Customs	0.68	0.05	0.58	0.78	0.47	0.58
Development	0.60	0.77	0.67	0.45	0.81	0.57
Economic and Monetary Affairs	0.91	0.93	0.92	0.85	0.93	0.89
Education Training Youth	0.77	0.88	0.82	0.64	0.94	0.77
Employment and Social Policy	0.62	0.78	0.69	0.68	0.83	0.75
Energy	0.76	0.61	0.67	0.71	0.71	0.71
Enlargement	0.53	0.53	0.53	0.70	0.44	0.54
Enterprise	0.19	0.23	0.20	0.44	0.15	0.23
Environment	0.62	0.80	0.70	0.69	0.82	0.75
External Relations	0.73	0.36	0.48	1.00	0.23	0.37
External Trade	0.47	0.64	0.53	0.53	0.61	0.57
Fight Against Fraud	0.40	0.25	0.31	1.00	0.25	0.40
Food Safety	0.95	0.75	0.85	0.90	0.82	0.86
Foreign and Security Policy	0.57	0.50	0.53	0.65	0.54	0.59
Human Rights	1.00	0.25	0.39	1.00	0.12	0.22
Humanitarian Aid	1.00	0.36	0.52	1.00	0.29	0.44
Information Society	0.62	0.68	0.65	0.54	0.72	0.62
Institutional Affairs	0.48	0.70	0.56	0.62	0.60	0.61
Internal Market	0.69	0.65	0.67	0.62	0.70	0.66
Justice Freedom Security	0.61	0.85	0.71	0.53	0.86	0.66
Maritime Affairs and Fisheries	0.84	0.81	0.82	0.91	0.80	0.85
Public Health	0.37	0.13	0.20	1.00	0.39	0.56
Regional Policy	0.62	0.55	0.57	0.88	0.50	0.64
Research Innovation	0.46	0.43	0.44	0.71	0.43	0.53
Taxation	0.85	0.57	0.65	0.94	0.54	0.68
Transport	0.65	0.71	0.68	0.67	0.81	0.73

Table 5.6: Class-wise precision (P) and recall (R) and F1-Score (F) for AVG-BiLSTM-ED-N (represented with suffix LSTM-A) and BiLSTM-ED-N (represented with suffix LSTM-B) on evaluated on document level. The best *precision*, *recall* and *f1-scores* among both the classifiers is highlighted in blue, red and green respectively. If the values across both the classifiers are same it not highlighted.

6. Related Work

This chapter regards previous works done in the domain of text classification on legal text. Then we regard work done using the splitting of data into n clusters and then the work done in the classification of multilingual text.

We first mention work about multi-class classification on legal text documents and subsequently general problems of multi-class classification under class imbalance. An approach for more fine-grained document classification has been proposed by Mencia et al. who perform multilabel classification on a large scale problem using EUR-Lex law texts. They predicted 4,000 labels for 19,596 [MF10]. We employ their published document collection for training our word embeddings. In contrast, we examine a setting with high-level labels, bilingual documents, class imbalance, and limited dataset size. Boella et al. added a module in the existing software (EUNOMOS, a knowledege management software) by classifying pieces of law into different categories in accordance with labeled data. This classification takes into account the similarity between laws from different domains, which in turn helps in the identification of more relevant information.

Maat et al. performed multi-class classification on sentences from Dutch legislation and compared a machine learning classifier with a pattern-based classifier. The sentences were created by dividing the document into categories like definition, obligation, repeal, application, and provision which are structural parts of a legal document [dMKW$^+$10]. Song et al. performed an analysis of multilingual opinionated sentences [SLB$^+$07]. We have employed a similar approach of training the classifiers with sentences created from the documents. Unlike both the approaches, here the whole document is divided into sentences and is assigned a label that the document had.

Previous studies on hierarchical text categorization by Ceci et al. have shown how hierarchical structures can be used to improve the efficiency and effectiveness of text classification. They have defined a top-down approach where a document is first classified into a higher-level category (*e.g. Science*) and then into a lower-level category (*e.g Biology, Chemistry*) [CM07]. We are not following the same approach

but something similar. Instead of classifying the documents into subcategories we are dividing the dataset into n groups and then employ a root classifier to predict which of the n classifiers should be invoked in order to predict the given sentence.

Gonalves and Quaresma [GQ10] have shown the benefit of combining multiple SVMs for several languages and classify international agreements of the European Union. Their results indicate that encompassing several languages can improve classifier performance. They set the minimum amount of documents per category to 50, thus limiting the lack of training documents and class imbalance. This is a restriction which we did not apply, and another difference is that we use summaries instead of the longer, original text as inputs for our classifiers to focus on their capability to handle difficult datasets. C.-P. Wei et al. argued that when the training set for a single language is small it is less representative of the target semantic space and a classifier trained on this data set would not yield satisfactory results. They have shown that adding multilingual data for predefined categories can help in increasing the performance of the classifier [WYL+14]. The dataset we are using is comparatively less, and it is available in multiple languages in the same predefined categories. Hence we use those categories to see how well our classifier performs on a single language and on multiple languages.

Zhang et al. proposed the use of distribution of positive examples in an imbalanced dataset to dynamically determine a threshold which will bring the number of positive and negative examples in the dataset close [ZWS13]. This is somewhat similar to what we have done to resample the dataset. However, contrary to their approach we are not trying to bring the number of positive and negative sample close; instead, we are using the number of samples to remove the sample from the majority class. To conclude the related work, we find multiple aspects of our proposed approaches and methods in the literature, such as multi-language settings, class-distributional criteria, ensemble methods and hierarchies.

7. Conclusion, Limitaions and Future Work

In this chapter, the summary of this thesis and the concluding remarks are presented in Section 7.1 along with the discussion on some limitations about the approaches and methods used in this thesis in Section 7.2. Future work is presented in Section 7.3.

7.1 Conclusion

This thesis shows that the SVM outperforms the BiLSTM trained on English and German language in various configurations. However, other characteristics of the BiLSTM make it viable in situations where there is an abundance of data to train as the SVMs are not highly scalable compared to the deep learning counterpart. Also, the BiLSTMs can process multiple languages using a single model; this is however not the case with SVMs as it would require the use of a language detector which not only adds processing overload but also can have poor results as the error by the language detector propagates downwards in the hierarchy. **??** shows that multilingual input helps the BiLSTM to achieve comparable results to the SVM. It also concludes that clustering the data affects the performance of the BiLSTM due to the specialization effect.

The results of the second research question exhibit that general-purpose resources can perform equivalent to the domain-specific resources on domain-specific tasks when the weights are allowed to update, however looking closely at the Table 5.5 we can see that the overall performance of the classifier with general-purpose resources is quite comparable to the classifier trained on domain-specific embeddings, but when the classes are underrepresented in the training set, the classifier with domain-specific embeddings performs better on those classes than the general-purpose embeddings. **??** shows that the classifier with a frozen word embedding layer failed to train correctly and overfitted, the cause for this while looking at the performance of the classifier with trainable general-purpose word embedding layer, can be due to the weight updates. The general-purpose word embeddings are trained on data

from various sources. These sources might contain data from multiple domains, and the fact that the semantics and syntactics of the language used in the different fields vary extensively, these word embeddings might not have captured the legal domain-specific semantics and syntactics. During the training of the word embeddings, no new words are added to the vocabulary; hence the difference between the performance of trainable and not trainable general-purpose word embeddings is due to the weight update.

From the evaluation of the third research question, it is convincing that a classifier benefits from multilingual inputs. The improvement might be an attribute of the fact that the number of samples per class for a monolingual classifier is less compared to the bilingual classifier. Less training data for the monolingual classifier means that it has less representation of the target semantic space and adding more samples in the form of a different language would increase the performance of the classifier. However on some classes the performance of the monolingual classifier is better than the bilingual classifier.

7.2 Limitations

During the data resampling phase, exploiting the multi-label property of the dataset, the documents belonging to more than one category were removed from the class with the highest number of samples. This process introduces a bias in the training dataset towards the minority classes. Furthermore, eliminating the samples from the majority class will reduce the representativeness of that class. This could be severe in the case where the data point being removed is a unique representation of the class it is being removed from.

Clustering the data for classification is counter-intuitive. Clustering on one hand groups similar objects or samples together using some form of distance measure calculated using instance attributes. Segregating instances based on similarity produces unfavorable conditions for a classifier to learn patterns to distinguish between similar instances of a group. To illustrate this hypothesis, we would like to present an application use case of classifying different brands of beer and cola when the labels on them are removed. It is apparent to distinguish between a beer bottle and cola bottle but if we were to cluster them based on alcohol content, then all the beer bottles will be in a separate cluster, and all the cola bottles will be in a different group. It would be difficult for a classifier to distinguish between a Heineken beer, a Budweiser beer, and a Carlsberg beer as the color, and the shape of all these bottles is same. The color and the shape, however, will not be the only features used for classification but are good enough to make a point. On the other hand, in our dataset the clustering approach has been effective to solve the classification problem better than a non-clustering approach

7.3 Future Work

With the abundance of text data available with their labels, it becomes rather easy to test different approaches to provide a better and target specific solutions for some problems in text categorization genre. This thesis tries to answer a small

fraction of those problems. SVMss are thoroughly studied and used in ample of text categorization problems, but other algorithms such as Naive Bayes classifier and k-nearest neighbors also have shown to work well with textual data. Similarly, Convolutional Neural Networks have been demonstrated to outperform BiLSTMss in many classification tasks.

During the data resampling phase, duplicate samples from the majority class were removed, and this created a bias towards the minority class. This can be addressed in the future by having an agnostic evaluation strategy, which considers the multi-label aspect of the data during the testing phase.

In the case of general-purpose word embeddings, many different algorithms with their advantages and disadvantages have been shown to perform better than their predecessors. Google's multilingual word embedding *BERT*, Zalando's *Flair* and Allen NLP's *ELMo* word multilingual word embeddings are other few general-purpose word embeddings which can be tested to see if they can perform well for legal domain specific tasks?

It has been shown during the evaluation that multilingual input helps in the betterment of the classifier performance. It would be interesting to know at what point adding more languages becomes futile when considering training cost versus the performance increase.

A. Appendix

A.1 Backpropagation Example

To better understand the backpropagation algorithm mention in Section 2.6.1, consider the following example.

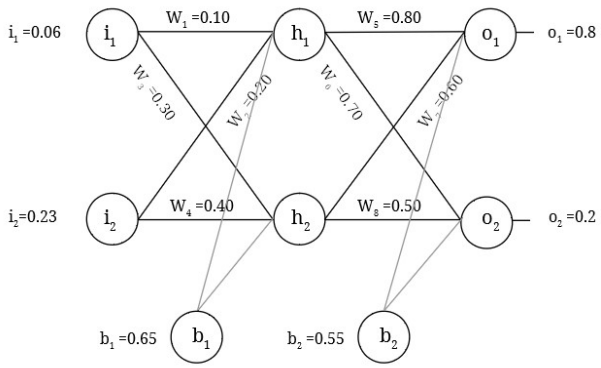


Figure A.1: Basic structure of a neural network with weights and bais initialized

The goal of backpropagation is the adjust the weights such that the neural network can correctly map the input to the output. To begin with consider the neural network in the Figure A.1 with inputs to the neural network be $i_1 = 0.06$ and $i_2 = 0.23$ to which the neural network should output $o_1 = 0.8$ and $o_2 = 0.2$.

In the forward pass, the neural network is feed in the inputs and given the weights and biases in the Figure A.1, it has to predict the output. It has to figure out the output at each hidden layer neuron, i.e h_1 and h_2 and squash each output using activation function and do the same at output layer. The activation function in this case is a *logistic function* of which sigmoid activation is a special case.

To calculate the input at h_1 as net_{h_1} we can use Equation 2.13 as follows,

$$net_{h_1} = W_1 * i_1 + W_2 * i_2 + b_1 \tag{A.1}$$
$$net_{h_1} = 0.10 * 0.60 + 0.20 * 0.23 + 0.65 \tag{A.2}$$
$$net_{h_1} = 0.756 \tag{A.3}$$

And to get the output out_{h_1} at h_1 we apply logistic function to Equation A.3,

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} \tag{A.4}$$
$$out_{h_1} = \frac{1}{1 + e^{0.756}} \tag{A.5}$$
$$out_{h_1} = 0.680 \tag{A.6}$$

Similarly calculating out_{h_2} at h_2 will be:

$$out_{h_2} = 0.681 \tag{A.7}$$

We can apply same process to output layer neuron, using the output from the hidden layer neuron h_1 and h_1 as input.

$$net_{o_1} = W_5 * h_1 + W_7 * h_2 + b_2 \tag{A.8}$$
$$net_{o_1} = 0.8 * 0.680 + 0.6 * 0.681 + 0.55 \tag{A.9}$$
$$net_{o_1} = 1.50 \tag{A.10}$$

And output out_{o_1} at o_1 we apply logistic function to Equation A.10

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} \tag{A.11}$$
$$out_{o_1} = \frac{1}{1 + e^{1.50}} \tag{A.12}$$
$$out_{o_1} = 0.8175 \tag{A.13}$$

Similarly output at o_2 can be obtained using the same process:

$$out_{o_2} = 0.7957 \tag{A.14}$$

We have got the output of the final layer, now we can calculate the error. An error function calculates the difference between desired output also known as target and the output predicted by the network. For the purpose of this example we will consider the standard Euclidean distance between the target and the output predicted by the network which we will henceforth refer to as output.

$$E_{(target,output)} = \frac{1}{2}(target - output)^2 \tag{A.15}$$

As we already know the values of the desired output and the predicted output, putting those values in the Equation A.15 we can calculate error for o_1 and o_2 as follows,

$$E_{o_1} = \frac{1}{2}(0.8 - 0.8175)^2 \tag{A.16}$$

$$E_{o_1} = 0.00015 \tag{A.17}$$

Similarly for E_{o_2},

$$E_{o_2} = 0.1774 \tag{A.18}$$

Combining Equation A.17 and Equation A.18 we can calculate total error E_{total} as,

$$E_{total} = E_{o_1} + E_{o_2} \tag{A.19}$$

$$E_{total} = 0.17755 \tag{A.20}$$

Now as the error is calculated we can update the weights so that the predicted outputs are closer to the desired outputs. Considering the weight W_5, we want to find out how much change in W_5 affects the error, i.e the rate of E_{total} w.r.t W_5, i.e $\frac{\partial E_{total}}{\partial W_5}$.

Applying chain rule,

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial W_5} \tag{A.21}$$

Calculating each term individually,

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \text{change in total loss w.r.t output of } h_1 \tag{A.22}$$

From Equation A.19, we can write,

$$E_{total} = \frac{1}{2}(target_{o_1} - out_{o_1})^2 + \frac{1}{2}(target_{o_2} - out_{o1})^2 \tag{A.23}$$

Taking the partial derivative of E_{total} with respect to out_{h_1}, the part $\frac{1}{2}(target_{o_2} - out_{o1})^2$ becomes 0 because out_{h_1} does not effect it and hence it is a constant.

$$\frac{\partial E_{total}}{\partial out_{o_1}} = 2 * \frac{1}{2}(target_{o_1} - out_{o_1})^{2-1} * -1 + 0 \tag{A.24}$$

$$\frac{\partial E_{total}}{\partial out_{o_1}} = -(target_{o_1} - out_{o_1}) = -(0.8 - 0.8175) = -0.0175 \tag{A.25}$$

Now for the second term in the equation Equation A.21, we find out the rate of change of out_{h_1} w.r.t net_{h_1}, hence we need to calculate the partial derivative of the logistic function.

$$out_{h_1} = \frac{1}{1 - e^{net_{h_1}}} \tag{A.26}$$

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1} * (1 - out_{h_1}) \tag{A.27}$$

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = 0.8175(1 - 0.8175) = 0.1491 \tag{A.28}$$

Finally, how much the n_{o_1} changes w.r.t W_5,

$$net_{o_1} = W_5 * out_{h_1} + W_7 * out_{h_2} + b_2 \tag{A.29}$$

$$\frac{\partial net_{o_1}}{\partial W_5} = 1 * out_{h_1} * W_5^{(1-1)} + 0 + 0 = out_{h_1} = 0.680 \tag{A.30}$$

Puttinng Equation A.25, Equation A.28 and Equation A.30 together, we get:

$$\frac{\partial E_{total}}{\partial W_5} = -0.0175 * 0.1491 * 0.680 = -0.00177429 \tag{A.31}$$

To get the new updated weights, we then subtract this the value obtained in Equation A.31 from the old weight multiplying it with learning rate which is 0.1 in our case:

$$W_{5_{new}} = w_5 - \text{learning rate} * \frac{\partial E_{total}}{\partial w_5} = 0.8 - 0.1 * (-0.00177429) = 0.800177429 \tag{A.32}$$

Similarly, we can calculate all the weight updates for $W_{1_{new}}$, $W_{2_{new}}$, $W_{3_{new}}$,$W_{4_{new}}$, $W_{6_{new}}$,$W_{7_{new}}$,$W_{8_{new}}$ using the method mentioned above.

A.2 Calculating Micro-average Precision Recall and F1-Score

The Table A.1 and Table A.2 shows the precision, recall and f1-scores of BiLSTM for cluster 1 trained on clustered data show in Figure 5.2 and Figure 5.1 as BiLSTM-D-ED-C evaluated on document level.

Classes	Precision	Recall	F1-Score	# Samples
Agriculture	0.90	0.86	0.88	50
Audiovisual and Media	1.00	0.10	0.18	10
Competition	0.96	0.83	0.89	30
Consumers	0.59	0.65	0.62	74
Employment and Social Policy	0.71	0.88	0.79	94
Energy	0.97	0.64	0.77	56
Enterprise	0.65	0.42	0.51	26
Environment	0.70	0.84	0.76	88
Food Safety	0.93	0.82	0.87	76
Information Society	0.71	0.84	0.77	80
Internal Market	0.72	0.75	0.74	148
Public Health	0.86	0.43	0.58	44
Taxation	0.95	0.71	0.82	28
Transport	0.76	0.86	0.81	94
micro avg	**0.76**	**0.76**	**0.76**	**898**
macro avg	**0.81**	**0.69**	**0.71**	**898**

Table A.1: CLass-wise precision, recall and f1-score

Predicted Values

Actual Value	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total Samples
1	43	0	0	1	2	0	0	2	2	0	0	0	0	0	50
2	0	1	1	0	0	0	0	0	0	8	0	0	0	0	10
3	0	0	25	0	0	0	0	0	0	0	1	0	0	4	30
4	2	0	0	48	1	0	0	2	2	2	11	0	0	6	74
5	0	0	0	0	83	0	1	0	0	3	7	0	0	0	94
6	0	0	0	1	0	36	0	14	0	1	2	2	0	0	56
7	3	0	0	0	3	0	11	2	0	3	4	0	0	0	26
8	0	0	0	6	2	0	0	74	0	0	2	0	0	4	88
9	0	0	0	6	0	0	0	4	62	0	4	0	0	0	76
10	0	0	0	0	2	0	0	0	0	67	7	0	0	4	80
11	0	0	0	12	7	1	4	1	0	4	111	1	1	6	148
12	0	0	0	7	15	0	0	0	1	0	2	19	0	0	44
13	0	0	0	0	2	0	0	1	0	2	1	0	20	2	28
14	0	0	0	0	0	0	1	6	0	4	2	0	0	81	94

Table A.2: Confusion matrix for cluster 1 of BiLSTM trained on English and German corpus

To calculate the micro average precision we first need the values of TP and FP. These values are available in confusion matrix in Table A.2

As descibed in Section 2.12, we calculate micro-average precision for a classifier by adding values of $\frac{TP}{TP+FP}$ for all the classes.

Hence, for *Agriculture* the value in the Table A.2 for TP is highlighted in blue and FP are highlighted in red and FN are highlighted in yellow.

$$\text{Micro-Avg Precision for } \textit{Agriculture} = \frac{43}{43+2+3} \tag{A.33}$$

$$\text{Micro-Avg Recall for } \textit{Agriculture} = \frac{43}{43+1+2+2+2} \tag{A.34}$$

Similarly, getting the values of all the classes from the Table A.2 we get,

$$\begin{aligned}\text{Micro-Avg Precision} = &\frac{43+1+25+48+83}{43+2+3+1+0+25+1+48+33+83+34} \\ &+ \frac{36+11+74+62+67+111}{11+6+74+32+62+5+67+27+111+43} \\ &+ \frac{19+20+81}{19+3+20+1+81+26}\end{aligned}$$

$$\text{Micro-Avg Precision} = \frac{618}{898} = 0.7583 \approx 0.76 \tag{A.35}$$

Similarly, to calculate the micro-average recall we get all the TP and TN values from the confusion matrix,

$$\begin{aligned}\text{Micro-Avg Precision} = &\frac{43+1+25}{43+1+2+2+2+8+1+1+25+1+4} \\ &+ \frac{48+83}{48+2+1+2+2+2+11+6+83+1+3+7} \\ &+ \frac{36+11}{36+1+14+1+2+2+11+3+3+2+3+4} \\ &+ \frac{74+62+67}{74+6+2+2+4+62+6+4+4+67+2+7+4} \\ &+ \frac{111}{111+12+7+1+4+1+4+1+1+6} \\ &+ \frac{19+20}{19+7+15+1+2+20+2+1+2+1+2} \\ &+ \frac{81}{81+1+6+4+2}\end{aligned}$$

$$\text{Micro-Avg Recall} = \frac{618}{898} = 0.7583 \approx 0.76 \tag{A.36}$$

$$\text{Micro-Avg F1-Score} = 2 * \left(\frac{\text{micro-avg precision} * \text{micro-avg recall}}{\text{micro-avg precision} + \text{micro-avg recall}} \right) \tag{A.37}$$

$$= 2 * \left(\frac{0.76 * 0.76}{0.76 + 0.76} \right) \tag{A.38}$$

$$= 0.76 \tag{A.39}$$

To, calculate macro average precision, we need to add the per-class precision values from Table A.1

Macro-avg precision is,

$$= \frac{0.90+1.00+0.96+0.59+0.71+0.97+0.65+0.70+0.93+0.71+0.72+0.86+0.95+0.76}{14}$$

$$\text{Macro-Avg Precision} = \frac{11.41}{14} \approx 0.81 \tag{A.40}$$

Macro-avg recall is,

$$= \frac{0.86+0.10+0.83+0.65+0.88+0.64+0.42+0.84+0.82+0.84+0.75+0.43+0.71+0.86}{14}$$

$$\text{Macro-Avg Precision} = \frac{}{14} = 0.68785714 \approx 0.69 \tag{A.41}$$

A.3 Visualization of word embeddings

The visualization of the ten randomly selected words *industrie, alike, customary, siting, thailand, multidisciplinarity, sites, transaktions, hauskatze, formetanate* is given below.

The first step in visualization is to reduce the dimension, for this we are using t-Distributed Stochastic Neighbor Embedding (t-SNE) dimensionality reduction technique. Reducing the dimension is necessary because these words are in 300 dimensional vector space and it would be impossible to consider all the dimension during visualization.

As it is clear from the Figure A.2 that words like *transaktions* and *formetanate* are not present in the general-purpose embedding and hence their embedding weights are zero and they are coinciding with one another in the Figure A.2.

So we can see that after training *transaktions* and *formetanate* now are at different position Figure A.3. To see the effect of training on these words, we added two new words (*katze*,*normal*) and visualized it.

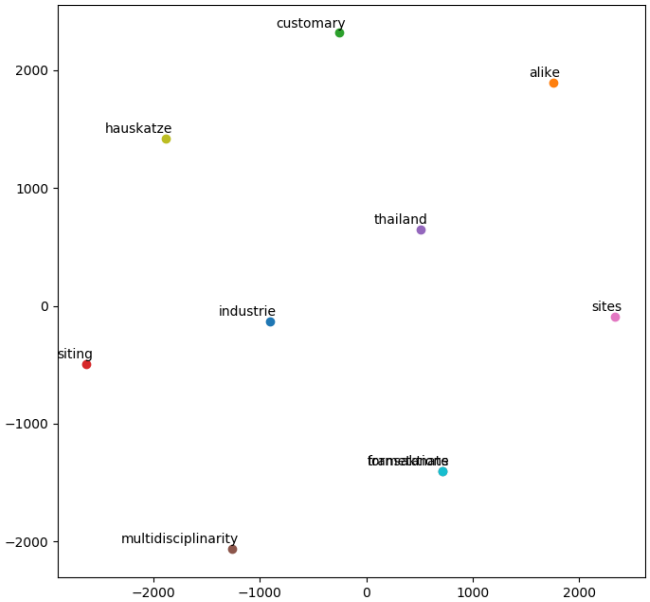


Figure A.2: Visualization of ten randomly selected words for frozen word embedding

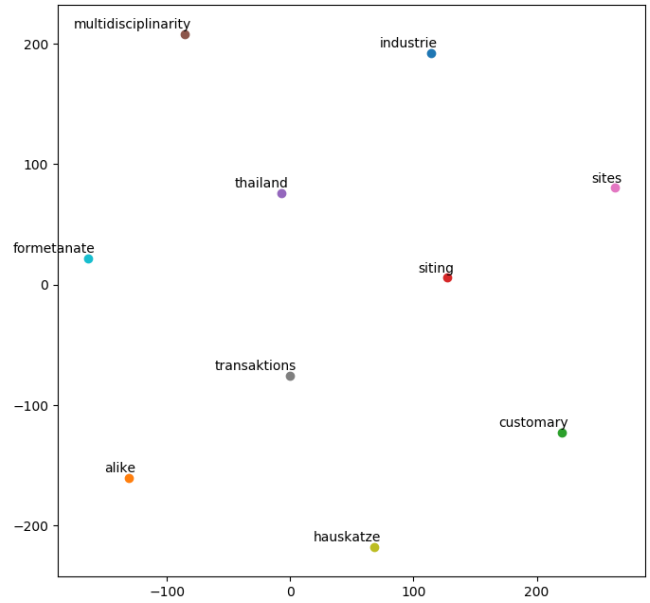


Figure A.3: Visualization of ten randomly selected words for trained word embedding

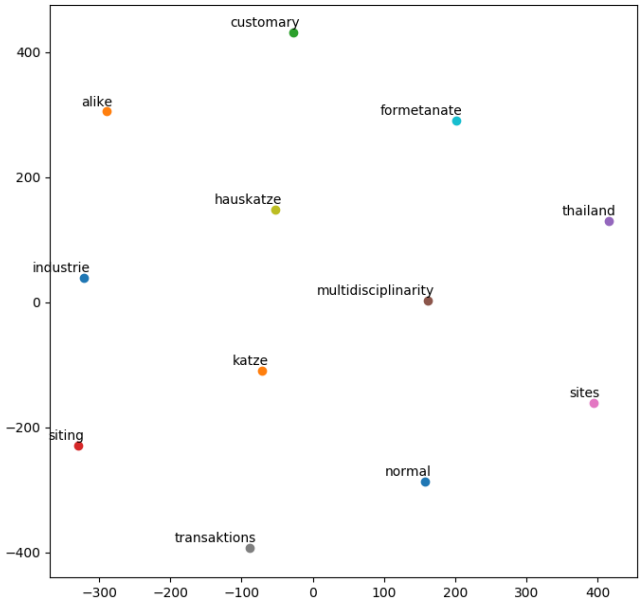


Figure A.4: Visualization after adding word *katze* and *normal* to the frozen embeddings

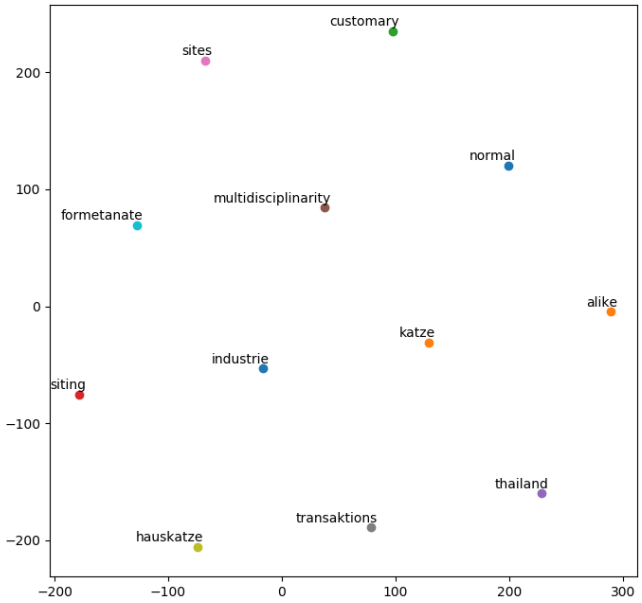


Figure A.5: Visualization after adding word *katze* and *normal* to the trained embeddings

The figure Figure A.4 shows the added two words before the training of the word embeddings, as we can see that the word *katze* and *hauskatze* are not that far from each other. We can calculate the cosine similarity between these two words before training as follows

$$\text{cosine similarity before training}_{(katze,hauskatze)} = \frac{\mathbf{katze} \cdot \mathbf{hauskatze}}{\|\mathbf{katze}\| \, \|\mathbf{hausekatze}\|} \tag{A.42}$$

$$\tag{A.43}$$

$$= 0.78172445 \tag{A.44}$$

and the cosine similarity after the embedding is trained is as follows,

$$\text{cosine similarity after training}_{(katze,hauskatze)} = 0.8065112 \tag{A.45}$$

Similarly, for the words *customary* and *normal* which are synonyms, the cosine similarity before training the embedding and after training embedding

$$\text{cosine similarity before training}_{(customary,normal)} = 0.18904422 \tag{A.46}$$

$$\text{cosine similarity after training}_{(katze,hauskatze)} = 0.28809269 \tag{A.47}$$

Hence, training the embedding further is necessary.

Bibliography

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018. (cited on Page 2)

[AIS04] Abdul Manan Ahmad, Saliza Ismail, and DF Samaon. Recurrent neural network with backpropagation through time for speech recognition. In *IEEE International Symposium on Communications and Information Technology, 2004. ISCIT 2004.*, volume 1, pages 98–102. IEEE, 2004. (cited on Page 18)

[Aly05] Mohamed Aly. Survey on multiclass classification methods. *Neural networks*, 19:1–9, 2005. (cited on Page 6)

[B+95] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995. (cited on Page 6)

[Bab17] Behrouz Babaki. Cop-kmeans version 1.5, July 2017. (cited on Page 47)

[BBF+00] Pierre Baldi, Søren Brunak, Paolo Frasconi, Gianluca Pollastri, and Giovanni Soda. Bidirectional dynamics for protein secondary structure prediction. In *Sequence Learning*, pages 80–104. Springer, 2000. (cited on Page 28)

[BDCH11] Guido Boella, Luigi Di Caro, and Llio Humphreys. Using classification to support legal knowledge engineers in the eunomos legal document management system. In *Fifth international workshop on Jurisinformatics (JURISIN)*, 2011. (cited on Page 2 and 34)

[BDCH+12] Guido Boella, Luigi Di Caro, Llio Humphreys, Livio Robaldo, and Leon van der Torre. Nlp challenges for eunomos, a tool to build and manage legal knowledge. *Language resources and evaluation (LREC)*, pages 3672–3678, 2012. (cited on Page 1)

[BFL+88] P. Biebricher, N. Fuhr, G. Lustig, M. Schwantner, and G. Knorz. The automatic indexing system air/phys - from research to applications. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '88, pages 333–342, New York, NY, USA, 1988. ACM. (cited on Page 5)

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. (cited on Page 2 and 25)

[BSF+94] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. (cited on Page 14)

[CAPLL+14] Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc., 2014. (cited on Page 3)

[CB06] Fabrice Colas and Pavel Brazdil. Comparison of svm and some older classification algorithms in text classification tasks. In Max Bramer, editor, *Artificial Intelligence in Theory and Practice*, pages 169–178, Boston, MA, 2006. Springer US. (cited on Page 2)

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. (cited on Page 35)

[CC08] Michael Chau and Hsinchun Chen. A machine learning approach to web page filtering using content and structure analysis. *Decis. Support Syst.*, 44(2):482–494, January 2008. (cited on Page 2)

[CJMdS17] Edilson Anselmo Corrêa Júnior, Vanessa Queiroz Marinho, and Leandro Borges dos Santos. Nilc-usp at semeval-2017 task 4: A multi-view ensemble for twitter sentiment analysis. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017. (cited on Page 2)

[CLR+17] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017. (cited on Page 25 and 52)

[CM07] Michelangelo Ceci and Donato Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78, Feb 2007. (cited on Page 75)

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. (cited on Page 6)

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. (cited on Page 2)

[DKM+16] Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, Texas, USA, November 2016. Association for Computational Linguistics. (cited on Page 25 and 52)

[dMKW+10] Emile de Maat, Kai Krabben, Radboud Winkels, et al. Machine learning versus knowledge based classification of legal texts. In *JURIX*, pages 87–96, 2010. (cited on Page 2 and 75)

[FGP06] Weiguo Fan, Michael D. Gordon, and Praveen Pathak. An integrated two-stage model for intelligent information routing. *Decis. Support Syst.*, 42(1):362–374, October 2006. (cited on Page 2)

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001. (cited on Page 10)

[FOD+09] Paul Ferguson, Neil O'Hare, Michael Davy, Adam Bermingham, Paraic Sheridan, Cathal Gurrin, and Alan F Smeaton. Exploring the use of paragraph-level annotations for sentiment analysis of financial blogs. 2009. (cited on Page 29 and 39)

[For08] George Forman. Bns feature scaling: An improved representation over tf-idf for svm text classification. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 263–270, New York, NY, USA, 2008. ACM. (cited on Page 2)

[FSS99] Toshiaki Fukada, Mike Schuster, and Yoshinori Sagisaka. Phoneme boundary estimation using bidirectional recurrent neural networks and its applications. *Systems and Computers in Japan*, 30(4):20–30, 1999. (cited on Page 28)

[GAL+06] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *LREC*, pages 1222–1225, 2006. (cited on Page 21)

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. (cited on Page 16)

[GLF+09] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009. (cited on Page 26 and 52)

[GQ10] Teresa Gonalves and Paulo Quaresma. Multilingual text classification through combination of monolingual classifiers. *CEUR Workshop Proceedings*, 605, 01 2010. (cited on Page 76)

[HBF+01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. (cited on Page 26)

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. (cited on Page 26 and 28)

[HSP99] Geoffrey E Hinton, Terrence Joseph Sejnowski, and Tomaso A Poggio. *Unsupervised learning: foundations of neural computation*. MIT press, 1999. (cited on Page 6)

[HW91] Philip J. Hayes and Steven P. Weinstein. Construe/tis: A system for content-based indexing of a database of news stories. In *Proceedings of the The Second Conference on Innovative Applications of Artificial Intelligence*, IAAI '90, pages 49–64. AAAI Press, 1991. (cited on Page 5)

[ID10] Nitin Indurkhya and Fred J Damerau. *Handbook of natural language processing*, volume 2. CRC Press, 2010. (cited on Page ix and 19)

[IS18] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *arXiv preprint arXiv:1801.05746*, 2018. (cited on Page 68)

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. (cited on Page 2 and 10)

[KS96] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996. (cited on Page 8)

[LMF10] Eneldo Loza Mencía and Johannes Fürnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia, editors, *Semantic Processing of Legal Texts – Where the Language of Law Meets the Law of Language*, volume 6036 of *Lecture Notes in Artificial Intelligence*, pages 192–215. Springer-Verlag, 1 edition, May 2010. accompanying EUR-Lex dataset available at http://www.ke.tu-darmstadt.de/resources/eurlex. (cited on Page 51)

[Luh58] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958. (cited on Page 5)

[LWHM16] Jörg Landthaler, Bernhard Waltl, Patrick Holl, and Florian Matthes. Extending full text search for legal document collections using word embeddings. In *JURIX*, pages 73–82, 2016. (cited on Page 1)

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. (cited on Page 23 and 25)

[MDP+11] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011. (cited on Page 21)

[MF10] Eneldo Loza Mencía and Johannes Fürnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Semantic Processing of Legal Texts*, pages 192–215. Springer, 2010. (cited on Page 33 and 75)

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. (cited on Page 14)

[Mit97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. (cited on Page 5)

[MLS13] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013. (cited on Page 25)

[MO06] Craig Macdonald and Iadh Ounis. The trec blogs06 collection: Creating and analysing a blog test collection. *Department of Computer Science, University of Glasgow Tech Report TR-2006-224*, 1:3–1, 2006. (cited on Page 29)

[MRS09] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):279–280, 2009. (cited on Page 32)

[MRS10] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010. (cited on Page 10)

[MS97] Dieter Merkl and Erich Schweighofer. The exploration of legal text corpora with hierarchical neural networks: A guided tour in public international law. In *ICAIL*, volume 97, pages 98–105. Citeseer, 1997. (cited on Page iii and 2)

[MŽBCB05] Martin Možina, Jure Žabkar, Trevor Bench-Capon, and Ivan Bratko. Argument based machine learning applied to law. *Artificial Intelligence and Law*, 13(1):53–73, 2005. (cited on Page 2)

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018. (cited on Page 2)

[Pre98] Lutz Prechelt. *Early Stopping - But When?*, pages 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. (cited on Page 53)

[RHW+88] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988. (cited on Page 16)

[Rob04] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004. (cited on Page 21)

[Rou87] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. (cited on Page 8)

[RPHF11] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011. (cited on Page 6)

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002. (cited on Page 2)

[SEK+08] Yohei Seki, David Kirk Evans, Lun-Wei Ku, Le Sun, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. Overview of multilingual opinion analysis task at ntcir-7. In *NTCIR*, 2008. (cited on Page 29)

[SHMO09] Rodrygo LT Santos, Ben He, Craig Macdonald, and Iadh Ounis. Integrating proximity to subjective sentences for blog opinion retrieval. In *European Conference on Information Retrieval*, pages 325–336. Springer, 2009. (cited on Page 29)

[SLB+07] Dawei Song, Raymond Y.K. Lau, Peter D. Bruza, Kam-Fai Wong, and Ding-Yi Chen. An intelligent information agent for document title classification and filtering in document-intensive domains. *Decision Support Systems*, 44(1):251–265, November 2007. (cited on Page 2 and 75)

[SM05] Pascal Soucy and Guy W Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *IJCAI*, volume 5, pages 1130–1135, 2005. (cited on Page 20)

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. (cited on Page 28)

[SPH02] Ralf Steinberger, Bruno Pouliquen, and Johan Hagman. Cross-lingual document similarity calculation using the multilingual thesaurus eurovoc. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 415–424. Springer, 2002. (cited on Page 52)

[Tho53] Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953. (cited on Page 8)

[VA16] Zeev Volkovich and Renata Avros. Text classification using a novel time series based methodology. *Procedia Computer Science*, 96:53–62, 2016. (cited on Page 28)

[vGB18] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018. (cited on Page 2)

[WCR+01] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001. (cited on Page 7)

[WLKF16] Yue Wu, Jun Li, Yu Kong, and Yun Fu. Deep convolutional neural network with independent softmax for large scale face recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1063–1067. ACM, 2016. (cited on Page 15)

[WWH05] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005. (cited on Page 29)

[WY12] Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012. (cited on Page 35)

[WYL+14] Chih-Ping Wei, Chin-Sheng Yang, Ching-Hsien Lee, Huihua Shi, and Christopher C. Yang. Exploiting poly-lingual documents for improving text categorization effectiveness. *Decis. Support Syst.*, 57:64–76, January 2014. (cited on Page 3, 42, 71, and 76)

[ZBH+16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016. (cited on Page 53)

[ZSCM13] Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington,

USA, October 2013. Association for Computational Linguistics. (cited on Page 3)

[ZWS13] Jing Zhang, Xindong Wu, and Victor S. Sheng. Imbalanced multiple noisy labeling for supervised learning. In *AAAI*, 2013. (cited on Page 76)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 13. May 2019