Standing on the Shoulders of Giants

Combining Google's BERT, Facebook's RoBERTa and FastAI's ULMFiT through Model Ensembling for Text Classification of Dutch Data Processing Agreements

Author: Bas de Koning (468379)

Supervisor: Dr. Eran Raviv

Second-assessor: Dr. Radek Karpienko

A thesis submitted in fulfillment of the requirements for the MSc in Data Science and Marketing Analytics

at the

Erasmus School of Economics

31 March, 2022

Abstract

In this research, text classification is applied to Dutch Data Processing Agreements (DPA). A DPA governs the obligations and rights between two commercial parties regarding the processing of personal data of consumers obtained by one party and processed by the other. Text classification on DPA's enables lawyers to decrease time spent on reviewing these contracts while making contracting more consistent and human error-free. Researching these potentials is worthwhile as the global legal market (worth 1 trillion US Dollars annually) remains underdigitalized. This research contributes by assessing which state-of-the-art text classification algorithm is the most capable and if the performance can be elevated by applying Model Ensembling. The considered state-of-the-art models are Bidirectional Encoder Representations from Transformers (BERT), a Robustly optimized BERT (RoBERTa), and Universal Language Model Fine-tuning for Text Classification (ULMFiT). Previous research has shown that BERT and RoBERTa outperform ULMFiT on multiple language tasks. In this research, the Rotation Estimated version of Dutch BERT (i.e. BERTje) outperforms ULMFiT with a macro-weighted F_1-score of 0.88 on 100% of the data, but with limited data, ULMFiT is better at predicting classes that differentiate based on a sequential pattern. At last, Model Ensembling improves model performance by 3% when enough data is available, giving a macro-weighted F_1-score of 0.91 on 100% of the data. The Model Ensembling is done with two different models to compare performance, i.e., Logistic Regression and Linear Discriminant Analysis (LDA). Results show that with limited data Logistic Regression performs better than LDA, but LDA outperforms Logistic Regression when enough data is available.

Table of Contents

1 Introduction

The field of Law is at the center of all business activity, causing it to be one of the largest markets with an annual global turnover of 1 trillion (10^{12}) US Dollars (Toews, 2019). Yet, this market remains profoundly underdigitalized, as it is notoriously slow to adopt new technologies (Toews, 2019). Rich (2020) found that contract review and drafting can be made more efficient by text classification algorithms, as it is estimated that inefficient contracting leads to a value loss of 5% to 40% on a given deal. By implementing text classification algorithms, lawyers can shift their focus from routine activities to more high-value work (Dale, 2019; Toews, 2019).

Text classification is a sub-domain of Natural Language Processing (NLP). Text classification is the field of computational techniques that focuses on the automatic analysis of human language (Cambria & White, 2014). The general idea is to input labeled texts into a Machine Learning algorithm that learns how certain words can predict a set of classes. To date, the highest performance in text classification has been achieved by different implementations of Neural Networks, which has led to a focus on this methodology (Devlin, Chang, Lee, & Toutanova, 2018). As a result, Neural Networks are the focus of this research and are explained in detail in section 4.2.

Van der Ploeg, Austin, and Steyerberg (2014) found that Neural Networks are data-hungry, meaning that they can only perform well on data sets that contain many observations. Howard and Ruder (2018) found a procedure that satisfies this data hunger through unsupervised pretraining of the Neural Network, essentially applying transfer learning in an NLP setting. Transfer learning in NLP consists of training a model on large amounts of general textual data, enabling the model to learn general properties of language, i.e., the meaning of words, grammar, and word dependencies. The model then only needs to be adjusted to a specific purpose, which in this research is the classification of legal clauses.

Collecting labeled data for legal contracts is expensive, as the person labeling the data must be an expert in the legal domain. Furthermore, the classes in the data can be very unbalanced for contracts, as an unacceptable clause is less common. An unacceptable clause is a clause that contains such wording that it results in an extremely one-sided relation among parties. Favorably, Neural Networks are not harmed by class imbalance as long as there is enough data for each class to learn its parameters. There is thus a need to find the optimal use of data that does not need to account for class imbalance.

Rotation Estimation can maximize the utilization of the given data in Neural Networks, as

training a Neural Network requires a separate validation set. Rotation Estimation is a procedure in which the training set is split into k-Folds, where one of the k-Folds is the validation set, and the other *k-1*-Folds are the training set, and the model is trained k times with a different validation set every time. This could reduce the randomness of the split and, more importantly for Deep Learning, the model sees every observation in the training data set (Geisser, 1993). One downside is that it might become harder to signal bias with the validation set as the parameters have seen the observations in the validation set.

Sakkis et al. (2001) has built on top of this idea by introducing Model Ensembling with multiple Level 1 Models and a Level 2 Model. The Level 1 Models are trained to predict to which observation a class belongs. The Level 1 Models then try to predict the classes of the observations in the validation set. The Level 2 Model then learns how to optimally combine these predictions into a final prediction. The procedure of Model Ensembling is explained in more detail in section 4.8. The Level 1 Models in this research are state-of-the-art Neural Networks.

The state-of-the-art Neural Networks for text classification are Bidirectional Encoder Representations from Transformers (BERT), Robustly optimized BERT (RoBERTa), and Universal Language Model Fine-tuning for Text Classification (ULMFiT), which are explained in sections 2 and 4.

These models are considered state-of-the-art text classification algorithms, because of their performance in classifying consumer reviews. Text classification is mostly applied to consumer reviews, meaning there are few studies done where NLP is applied in a legal context (Grimes, 2017; Medhat, Hassan, & Korashy, 2014). The supporting literature will therefore mostly be about NLP on consumer reviews.

Theoretically, the underlying assumptions of modeling consumer reviews and contractual clauses are the same. Clauses in a contract can be classified based on their content and corresponding consequences, equivalent to how reviews can be classified based on their sentiment. Only, the required preprocessing differs, as legal documents and consumer reviews differ in structure. Contracts are generally longer, contain insertion fields, fewer spelling mistakes, and no emoticons. Moreover, clauses typically contain titles that convey information about the content of the clause, and each clause begins with an article number or symbol.

This research considers one type of contract, being Data Processing Agreements (DPA). A DPA governs the rights and obligations of two commercial parties on processing personal data of consumers, where generally one party has collected the personal data and delegates its processing

to the other party. A DPA is governed and made obligatory by the General Data Protection Regulation (GDPR). As a result, DPAs are typically well structured, meaning each clause handles one topic, allowing for multi-class classification. The main research question is as follows:

– *"How can we successfully classify clauses in a Dutch Data Processing Agreement with limited data?"*

To be able to answer the research question, the following sub-questions are answered:

1. *How must a DPA be preprocessed for text classification?*
2. *What are the current state-of-the-art text classification algorithms and what are their differences?*
3. *What are the resulting effective differences between the models in terms of performance?*
4. *How does the amount of training data affect the performance of the models?*
5. *Can Rotation Estimation and Model Ensembling improve prediction performance?*

This research contributes to the academic literature on the application of text classification, specifically in a legal context. This research examines how the current state-of-the-art NLP models perform on limited data of legal documents, and whether the performance can be enhanced by Model Ensembling. At last, this research aims to further stimulate investigation on how to add value by reducing review time, human error, and inconsistency.

This thesis continues as follows. Section 2 discusses the existing literature of text classification. Section 3 discusses the data itself, the appropriate preprocessing, the data correction methodology, the reasoning behind the data classes, and the data partitioning. Section 4 contains the methodology of the models. Section 5 evaluates and interprets the results. Section 6 answers the sub-questions and the main question. Section 7 discusses the limitations of this research and gives recommendations for future research.

2 Literature Review

This research aims to compare different Neural Networks in their ability to classify legal clauses on different amounts of data. For this purpose, the considerations and research that went into the development of the current state-of-the-art in NLP are explained in this section. This section concludes with a summary of the main theoretical differences between the three models compared.

2.1 Bag-of-Words

At the beginning of text classification, the bag-of-words (BoW) approach was used to numericalize documents. The BoW transforms a document into a numeric feature matrix with a fixed length, where each feature is the word occurrence, word frequency, or term frequency-inverse document frequency score. The BoW approach has two major disadvantages: the word order is ignored and it can barely encode the semantics of words (Zhang, Wang, & Liu, 2018). As the word order is ignored, a lot of information is lost, as the sentences 'Not only do I love the new iPhone' and 'Only, I do not love the new iPhone' are represented in the same way. The disadvantage of not representing word semantics in the data is that similarity among words is neglected, as relationships between words can not be quantified. As a solution N-grams were introduced, unfortunately, they were found to be ineffective in a BoW-approach (Bekkerman & Allan, 2004).

2.2 Contextualised Word Embeddings

The shortcomings of BoW are overcome by using Word Embeddings with a Continuous-BoW or skip-grams (Mikolov, Karafiát, Burget, Černockỳ, & Khudanpur, 2010). Word Embeddings consist of real numbered vectors that represent words, where each word has its own vector that contains information about that word. Conceptually, words that are similar in their meaning and grammar should have similar vectors. The first model that creates a vector for each word is the Word2Vec model, where it learns the word vectors by predicting one word based on its context (Mikolov, Chen, Corrado, & Dean, 2013).

Unfortunately, the Word2Vec has two downsides. To begin with, as every word is represented by one vector, multiple senses of a word cannot be represented. Most words have multiple senses that depend on the context. For example, the word 'company' in the sentences 'My friends and family are great ***company***' and 'Apple is a market-leading and innovative ***company***' has two different meanings that depend on the context. Furthermore, the meaning of a word that the vector can describe is incomplete, as the vector is created on a fixed number of words, meaning that not all words that influence the meaning of a word can be taken into account.

These downsides are overcome by training Neural Network on top of the Word Embeddings, which increases the number of trainable parameters and creates a Deep Learning model. The added trainable parameters are the weights and biases in the Neural Network. The meaning of a word is then encoded in the multiplication of the vector in the Word Embeddings with the weights and

biases in the model, enabling the representation of different senses of words (Y. Goldberg, 2017). The Neural Network and the Word Embeddings are estimated by training the model on a large corpus of texts, where it has to predict an output that is based on word dependencies, such as predicting the word given the context or vice versa. The parameters then have obtained knowledge about the grammar, meaning of words, and word dependencies (Howard & Ruder, 2018). Such a model is called a Language Model (LM). There are multiple variations in defining a Language Model, in this research, we focus on the LM of the BERT model and the LM of ULMFiT.

2.3 Recurrent Neural Networks

RNN is a good architecture for language modeling, as RNN is designed to handle sequential data. Text data is sequential as the sequence of words influences the meaning of a text. By inputting words sequentially RNN's achieve high-performance (Mikolov et al., 2010).

One downside of RNNs, however, is that they have difficulty learning long-term dependencies because meaningful words far back in the sequence become inaccessible due to vanishing gradients (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). Vanishing gradients disable weights that are further back in the sequence from learning word relations. How vanishing gradients relate to the learning process in RNNs is explained in section 4.3.2.

Vanishing gradients are especially problematic for legal documents, as sentences can become extensive. E.g., for the sentence: 'The obligations of the Processor arising from this Data Processing Agreement also apply to those who process personal data under the authority of the ***Processor***.', to correctly predict the last word in this sentence the model must understand whose authority is meant, meaning the model has to bridge a gap of over 19 words. As that gap becomes larger, it becomes more difficult for RNNs to access important information. RNNs are explained in section 4.3. ULMFiT is an RNN with Long Short-Term Memory (LSTM), which is explained in 4.3.3.

2.4 Universal Language Model Fine-tuning for Text Classification

ULMFiT is an Adam Weight Dropped-Long Short-Term Memory (AWD-LSTM) model (Howard & Ruder, 2018). ULMFiT increased the model performance by transfer learning to obtain contextualized Word Embeddings as explained in the introduction and section 2.2. The pretrained weights only have to be trained once per language and significantly improve performance and convergence (Howard & Ruder, 2018). For Dutch there is only one pretrained ULMFiT model, which is developed by van der Burgh and Verberne (2019) and pretrained on 79MB of text data.

The LM in AWD-LSTM is estimated by calculating the likelihood that a word is next in a sequence (Y. Goldberg, 2017). The RNN autoregressive predicts the next word in a sentence, where the possible output is every word in the Word Embeddings, resulting in a final classification layer with as many output neurons as there are words in the Word Embeddings. After each word in the sequence, every word in the Word Embeddings obtains its probability on the likelihood it follows after that word. Thus predicting the word given the context, where the context is equal to all preceding words in the sequence.

2.5 Transformers: BERT & RoBERTa

Transformers have an entirely different model structure, they do not use a recurrent structure as in RNNs. The Transformer was introduced by Vaswani et al. (2017) and outperformed the Google Neural Machine Translation while allowing for more parallelization. BERT was introduced by Devlin et al. (2018) and uses only one part (i.e. *'Encoder'*) of the Transformer model, as the other part (i.e. *'Decoder'*) is solely meant for translation, technical explanation of BERT is provided in section 4.5.

BERT is the current state-of-the-art for NLP and builds on top of all the best ideas for NLP. BERT was developed with the idea to make clever use of "free" data, replacing the need to create data. The model is explained in section 4.5 and visualised in Figures 6 to 10.

One of BERT's main advantages over other models is that it reads from both left to right and right to left. Pretraining of BERT must be done on a very large corpus of text with many iterations, as the model has at least 110 Million parameters. When there is not enough data, BERT is very prone to overfitting (H. D. Lee, Lee, & Kang, 2020a). BERT has two model sizes, BERT Base and BERT Large, there is unfortunately no Dutch BERT Large model yet.

There is currently only one pretrained Dutch BERT model available, being BERTje. BERTje is a BERT Base model and is pretrained on 12GB of uncompressed text of around 2.4 billion tokens (de Vries et al., 2019).

2.5.1 Pretraining

BERT is pretrained bidirectionally by using masking through auto-encoding, which means that some percentage of the input tokens are hidden at random so that the model can learn contextualized Word Embeddings by predicting the hidden tokens (Devlin et al., 2018). Furthermore,

'Next-Sentence Prediction' is also applied during pretraining, which lets the model learn relationships between sentences. It is technically explained in section 4.6.

2.5.2 RoBERTa

RoBERTa is a variation of BERT that is pretrained on a larger amount of data, without Next-Sentence Prediction. RoBERTa improves between 2-20% over BERT and was introduced by Facebook (Liu et al., 2019). RobBERT is a Dutch RoBERTa model and is pretrained on 39GB of text of around 6.6 billion words (Delobelle, Winters, & Berendt, 2020). RobBERT has outperformed all Dutch text classification models on all language tasks, except Named Entity Recognition.

2.6 Summary

The current state-of-the-art in NLP is dominated by Neural Networks, where the architecture varies among models. ULMFiT is an AWD-LSTM, which is an RNN with LSTM cells. BERT and RoBERTa are Transformers. BERT and RoBERTa are pretrained on a larger amount of data and consist of more parameters. RoBERTa is trained on even more data than BERT but without the Next-Sentence Prediction.

3 Data

This section describes the data classes, data collection, data preprocessing, data correction, and data partitioning.

3.1 Data Labels

Contracts undergo different judgments dependent on the use case. E.g., a business that outsources data processing has more interest in strict clauses than the business that processes the data. This does not change the correct classification of the content of a contract, it does however impact the conclusion of the contract review. Therefore, the labels only relate to the content of a contractual clause. The final judgment is done by looking up whether the content is 'ok', 'not ok' or 'alert' in a dictionary approach, where the dictionary depends on the use case. A dictionary approach simply entails matching keywords, hence this research only focuses on the correct classification of the content of a clause.

In total 68 classes are used. As it relates to the content of the clause, some translated examples are *liability-limited* ('aansprakelijkheid-beperkt'), *'liability-unlimited'* ('aansprakelijkheid-onbeperkt' and *'databreach-48hours'* ('datalek-48uur'). These examples illustrate how the action required depends on the use case, as only the content is classified.

3.2 Data Collection & Correction

The text data is collected and labeled by interns and lawyers of ICTRecht, a legal firm located in Utrecht, the Netherlands. As the data is labeled by a third party, the risk of researcher bias is reduced. The data consists of 600 labeled Dutch Data Processing Agreements (DPAs). The agreements are separated into multiple strings by cutting after each new-line argument ('\n'), where each string is labeled according to one of the 68 classes. In total there are 19860 labeled clauses in the DPA data set. The distribution of the classes is given in Table 3 in Appendix A.

Due to human error, there are mistakes in the labels. To find these mistakes, a ULMFiT model is trained on 100% of the data. Using that model, the validation loss per observation is calculated for the entire data set. This is done by flipping the training and validation set. With the loss per observation, the data frames are sorted in descending order. The cut-off point is determined where the first observation clearly has the right class. The observations with the highest losses are put into a separate file. This file is sent back to ICTRecht for reconsideration. By this procedure around 1.500 observations in the data set (roughly 10% of the whole data) have been corrected.

3.3 Data Preprocessing & Partitioning

Natural Language Processing requires preprocessing of the corpus, where the required preprocessing depends on the characteristics of the corpus, for example, legal documents are less likely to contain typos than reviews. For this matter, the following preprocessing steps are taken.

To begin with, the header of each article is put into sequence with their respective clause, embedded with special tokens to signal separation between the header and the content of the clause. For DPAs it was found that the model performance increases substantially by implementing the header of a clause into the sequence of text, by wrapping the header into a begin marker and an end marker. Furthermore, all words are included, as it is unlikely that a contract contains typos or uninformative words. Any symbols created by text conversions can be deleted through regular expressions, as they contain Unicode.

Furthermore, as every new line in a document is considered as a separate string, many strings

in the data contain noise, e.g., they are as long as two or three words, only consist of a sign field, or only contain the date. This noise is filtered out by `regular expressions` in a `function` call so that new data can easily undergo the same procedure.

At last, the beginning of each string is characterized by either a number depicting the article number of the clause or a symbol indicating it is a summation. These tokens do not convey anything about the content and implication of the clause. Therefore, for all strings that begin with a number, the number is replaced by "`new-article`". And for all strings that begin with a symbol, the symbol is replaced by "`summation`".

After the strings have been preprocessed, the data is split into a stratified test and training set with a 20% to 80% ratio. This results in 3950 observations in the test set and 15797 in the training set for 100% of the data. The test set needs to be of a considerate amount to properly evaluate the different algorithms, while not sacrificing too much training data. With a stratified test set of 10%, not all classes are represented, and as there are still 15797 observations to train, this is considered a good split.

The differences in model performances are evaluated on different subsets of the data, being 20%, 40%, 60%, 80%, and 100% of the training set. Within these subsets, a 20% validation set is created. This is required for proper fine-tuning to prevent overfitting. Furthermore, the 20% validation set is also used for Model Ensembling, thus requiring a substantial amount of observations.

4 Methodology

This section describes the mathematics behind the algorithms applied as well as their application and evaluation. This section continues as follows: first, the basics of Neural Networks are explained. Second, Recurrent Neural Networks, LSTM-cells, and AWD-LSTM regularisation are explained. Third, BERT and RoBERTa are explained. At last, the procedures of Rotation Estimation, Model Ensembling, and Model Evaluation are explained.

4.1 Practical Considerations

The GPU used for this research is an Nvidia Tesla T4 15 GB GPU, with CUDA Version 11.0. The underlying package for ULMFiT, BERTje, and RobBERT is `PyTorch`. ULMFiT is developed and provided by `FastAI`, BERTje and RobBERT are variations of the standard BERT model as developed and provided by `HuggingFace`. A random seed is set in every environment (CUDA,

CPU, PyTorch) to make the results as reproducible as possible.

4.2 Basic principles of Neural Networks

In this thesis multiple state-of-the-art Neural Network architectures are evaluated and ensembled. For sake of comprehension, a general introduction into Neural Networks is given in this section, explanation of the exact architectures used follows later in this chapter. In Figure 1 a simple deep Neural Network is visualized.

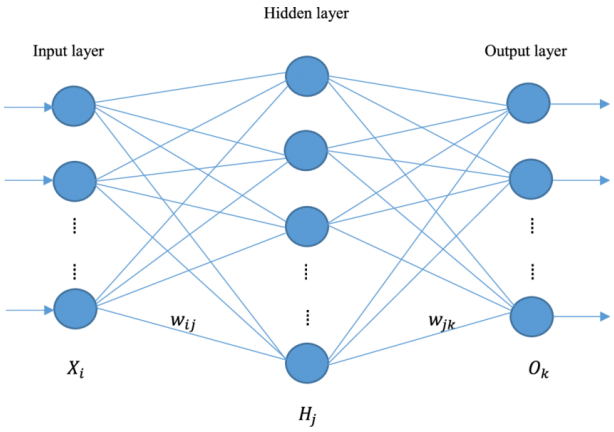


Figure 1: Feed Forward Artificial Neural Network architecture. Source: (Zhong & Enke, 2019)

All layers contain neurons, which are represented by circles. The vertical rows of neurons are layers, where the first layer is the input layer (X_i), the last layer is the output layer (O_k), and the layer in between is the hidden layer (H_j). The hidden layer and output layer process the data by using matrix multiplication. Within each neuron, a weight is multiplied by the input vector. The weights vector of a hidden layer consists of the weights of all the neurons, meaning they pass through every layer is one matrix multiplication between the input vector and the weight vector. In the output layer, the result of the model is generated, being either classification or regression.

4.2.1 Activation functions

The output of the multiplication within each neuron depends on the activation function. There are many activation functions (Lederer, 2021). The activation function Rectified Linear Units (ReLU) is found to be optimal for hidden layers, as it is a nonlinear function that can learn complex relationships while acting like a linear function (Goodfellow, Bengio, & Courville, 2016). ReLUs

react more proportionate in all areas, do not push large and small values into a small range and, as a result, are better able to differentiate between multiple values. ReLUs are thus more sensitive and saturate less, which means the model can learn more relationships in the data and it suffers less from the vanishing gradient problem, which is explained later in section 4.3.2 (Goodfellow et al., 2016). The formula of the ReLU is:

$$ReLU(s) = max(0, s) \tag{1}$$

The output o of a node q within layer l is equal to the transformation of the activation function a on scalar s, where s is equal to the sum of all outputs of the previous layer $l-1$ multiplied by a distinct weight w_q and added up by a distinct bias term b_q. The formula for the output o of neuron q in layer l is thus:

$$o_q^{[l]} = a^{[l]}(s_q^{[l]}) = a^{[l]}(\sum w_q^{[l]} o^{l-1]} + b_q^{[l]})\,, \tag{2}$$

In classification, the final layer is either a sigmoid for binary classification or a softmax for multi-class classification. The formula of the softmax activation function is as follows:

$$Softmax(s) = \frac{e^{s_p}}{\sum_j^C e^{s_j}}\,, \tag{3}$$

Where p refers to the actual class of the one-hot encoded vector $j = 1, 2, ..., C$, where C is the total number of classes, s is the scalar output of the weights and biases of the final layer for the actual class p and all classes j. By dividing the output by the sum of all possible outputs it ensures that the sum of all probabilities equals 1.

4.2.2 Loss function

Optimization of the parameters is done by minimizing a loss function, where the loss function depends on the problem. In this research, the loss function used is the cross-entropy loss function. Cross-entropy loss measures the difference between two probability distributions for several outcomes. These probabilities distributions are the true outcome and the predicted outcome, where the predicted outcome tries to approximate the true outcome. On the left side of the formula, the general equation for cross-entropy loss is given and on the right side it is adjusted for multi-class cross-entropy loss, i.e.:

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y \cdot \log \hat{y}_i = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y \cdot \log \frac{e^{s_p}}{\sum_j^C e^{s_j}}\,, \tag{4}$$

where y is a one-hot encoded vector where the value is 1 at the index of the actual class and 0 otherwise. $\hat{y}_i$ is a vector that contains the probability scores outputted by the Neural Network for each class. By multiplying the vector y with the probabilities vector $\hat{y}_i$, the loss is calculated based on the estimated probability of the class it should have predicted (C_p), with a reference value of 1. Doing so, it automatically accounts for the negative classes as the probability score of the positive class also depends on the probability scores of the negative classes. At last, PyTorch computes the loss per batch to speed up convergence.

4.2.3 Regularisation: Weight Decay, Dropout, and Early Stopping

To enable a model to make accurate predictions on new data, it has to generalize well (Goodfellow et al., 2016). Model generalization can be increased through regularisation, which leads to the bias-variance trade-off (Reed & Marks, 1999).

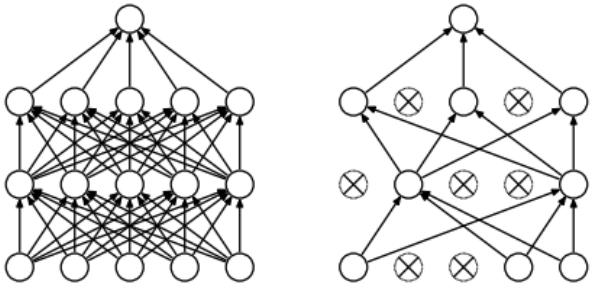
Overfitting is when the model has low variance on the training data and generalizes poorly on unseen data. Overfitting can be dealt with by either increasing the number of training samples or reducing the complexity of the network (i.e., increasing bias) (Bishop, 1995). A model can overfit the data when it has the capacity to do so, meaning that the model is too complex in relation to the number of data points. Artificial Neural Networks are likely to overfit the data, as they contain many parameters (Reed & Marks, 1999). Overfitting is signaled by using a validation set: the model overfits the training data when the loss on the validation set does not decrease while the loss on the training set decreases.

One way to reduce the complexity of the model (increase bias) is by ensuring the weights in the model remain small, as large weights cause sharp differences in the activation functions and therewith large changes in output for small changes in inputs (Reed & Marks, 1999). Weight Decay specifically serves that matter. Weight Decay and Dropout are the most common bias terms used in Neural Networks (Goodfellow et al., 2016).

Weight Decay is inspired on L_2-regularisation, where the sum of the square of the parameters is added to the loss function to prevent the parameters from becoming too large (James, Witten, Hastie, & Tibshirani, 2021). In Neural Networks, a fraction of the parameters is subtracted from its update by backpropagation. The implementation of weight decay is further explained in section 4.2.4.

Dropout reduces the model's complexity as it narrows the network. In Figure 2 the

effect of Dropout in a Neural Network is visualized. During training, when a unit is dropped, it is temporarily removed along with all its incoming and outgoing connections. As a result, each update to a layer during training is executed with a different form of that layer. Effectuating that it approximates training a large number of Neural Networks in parallel, as it estimates multiple variations at different time steps (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Moreover, the parameters become noisy and are forced to cooperate more, making the model more robust (Srivastava et al., 2014).



(a) Standard Neural Net (b) After applying dropout.

Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Figure 2: Dropout. Source: (Srivastava et al., 2014)

The probability that a neuron is dropped is depicted by the Bernoulli distribution. Dropout ensures the sum of the activations remains constant by rescaling the leftover activations by multiplying with $\frac{1}{1-p}$, where p is the probability of dropout. $p = 0.5$ maximizes the number of possible networks sampled from the probability distribution, maximizing the regularisation through dropout (Srivastava et al., 2014).

Early Stopping gives the possibility to automatically stop the training cycle when the validation loss has not decreased since a certain amount of epochs, the hyperparameter that determines this amount of epochs is called `patience`. The `patience` is set to 5 in ULMFiT, as it can take a few epochs for the model to improve and the epochs are fast. The `patience` in BERTje and RobBERT is set to 2, as the epochs take longer.

4.2.4 Optimization

Neural Networks optimize their parameters by minimizing a loss function through backpropagation, a learning rate, and an optimizer. These are explained in this section.

Backpropagation is the process of computing the partial derivatives to determine in which direction the weight and biases must be adjusted to decrease the loss function, formally put: $\frac{\partial L(\hat{y},y)}{\partial \hat{y}}$. This formula can be rewritten to: $\frac{\partial L(\theta)}{\partial \theta}$, where θ is the parameter vector of all weights and biases, as all other terms are constant. The $L(\theta)$ is computed at the output layer, which means that the partial derivatives of the loss with respect to the weights and biases are calculated recursively, for which the chain rule is applied (Werbos, 1990).

Optimizer: Stochastic Gradient Descent (SGD) is a first-order derivative optimization algorithm that adjusts the weights and biases sequentially in the direction that decreases the error rapidly, formally put:

$$\theta_{updated} = \theta - \gamma \frac{\partial L(\theta)}{\partial \theta}\,, \tag{5}$$

where $\theta_{updated}$ are the updated parameters and γ is the learning rate. The learning rate determines the pace of convergence: a high learning rate amplifies the updates, but runs the risk overshooting the local minimum causing convergence failure (Kingma & Ba, 2017).

Optimizer: Adam is a variation of SGD which improves convergence by momentum. Momentum entails the alteration of the parameter update for every parameter by dividing the gradients for each parameter with the square of their recent gradients (Kingma & Ba, 2017).

Adam must be decoupled from weight decay (`AdamW`), which results in a good optimizer (Loshchilov & Hutter, 2019). Adam has a low memory requirement, is invariant to diagonal rescaling of gradients, and is computationally efficient, making it well suited for problems with large data and parameters (Kingma & Ba, 2017). `AdamW` is the default optimizer in both Transformers and `FastAI`.

Decoupling weight decay from Adam means that weight decay is not computed in the loss. The weights are updated in the following step-wise manner: first, the gradients are computed, then the unbiased exponential moving average is calculated over two different ranges determined by β_1 and β_2 (i.e., formulas 6 and 7), the weight update is calculated and applied.

$$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1},\ where\ g_t = \frac{\partial L(\theta)}{\partial \theta}\,, \tag{6}$$

$$v_t = \frac{\beta_2 m_{t-1} + (1-\beta_2) g_t}{1-\beta_2}\,, \tag{7}$$

$$\theta_{updated} = \theta_t - \gamma \cdot \lambda + \frac{m_t}{v_t} \tag{8}$$

Where β_1 and β_2 are hyperparameters that are set to their default values, being 0.9 and 0.99 respectively. These values are empirically found to work well (Kingma & Ba, 2017). The weight

update is then calculated by applying formula 8, where γ is the learning rate and λ is the weight decay.

Adam is applied on the loss function, which PyTorch computes per batch, as mentioned in section 4.2.2. The batch size determines how many observations are in each batch. The recommended batch size for BERT and RoBERTa is 32, but due to computational limitations, the batch size is 12 (Liu et al., 2019). The batch size used for ULMFiT is 64.

4.3 Recurrent Neural Networks

Everything stated in the previous section is directly applicable to the RNN architecture. The architecture of a simple RNN is given in Figure 3. It illustrates how the network is unfolded over a sequence of length n. s represents the neurons, x the word inputs, and y is the dependent variable. A simple RNN has a layer with one neuron for each word input, where the weight matrix is the same for all input layers.

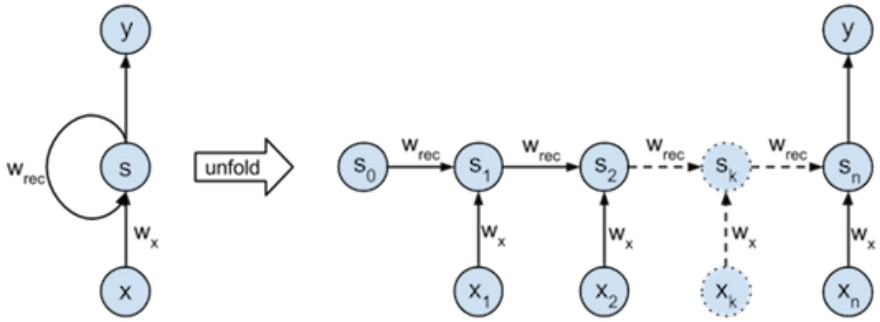


Figure 3: Recurrent Neural Network architecture. Source: (Roelants, n.d.)

In this model, the gradients at each step are calculated for all previous layers, meaning the gradients are calculated in the usual way, alias backpropagation through time (BPTT). Consequently, the model becomes as deep as the number of tokens in a document, making the model computational infeasible for larger documents, as the derivatives need to be calculated for all words with each their layer. The solution is called truncated backpropagation through time (TBPTT), which deletes the gradient history up until a certain activation, resulting in a 'stateful' model. The gradient history is deleted at the end of an input sequence, the length of this input sequence is a hyperparameter.

4.3.1 Bidirectional RNN's

A downside of an RNN-model is that an RNN-based LM only can read the context from either right to left or left to right. Training a bidirectional RNN Language Model is not possible, as the model then can see the word it must predict. As it is not possible to train an LM, it is not possible to pretrain the explanatory variables, being the weights in the model. This is a problem for limited data, as with enough data the parameters can be trained directly during the estimation of the classification model.

For limited data, the solution is to train two separate LM's and then average the predicted probabilities during classification, which worked well for ULMFiT (Howard & Ruder, 2018). Given that this research is about limited data, that procedure is applied.

4.3.2 Vanishing Gradients in RNN's

Although the unrolled RNN seems deep, as every hidden to hidden layer uses the same weight matrix, the network is effectively not that deep as it cannot compute any sophisticated computation that outperforms a linear model. The solution is to use a multi-layer RNN. However, multi-layer RNNs result in very deep models that have vanishing gradients (D. Goldberg, 2000). ReLUs as an activation function are the least susceptible to vanishing gradients, as all other activation functions require taking the derivative of a value between 0 and 1 (Hochreiter et al., 2001). Nonetheless, as parameter updates are based on backpropagating the loss value from the final layer to the first layer, the backpropagated error decreases exponentially as a function of distance from the final layer, regardless of the activation function (Sussillo, 2014). As a result, the weights that describe relations between words further apart cannot be quantified. The solution is applying LSTM-cells, which is further explained in section 4.3.3.

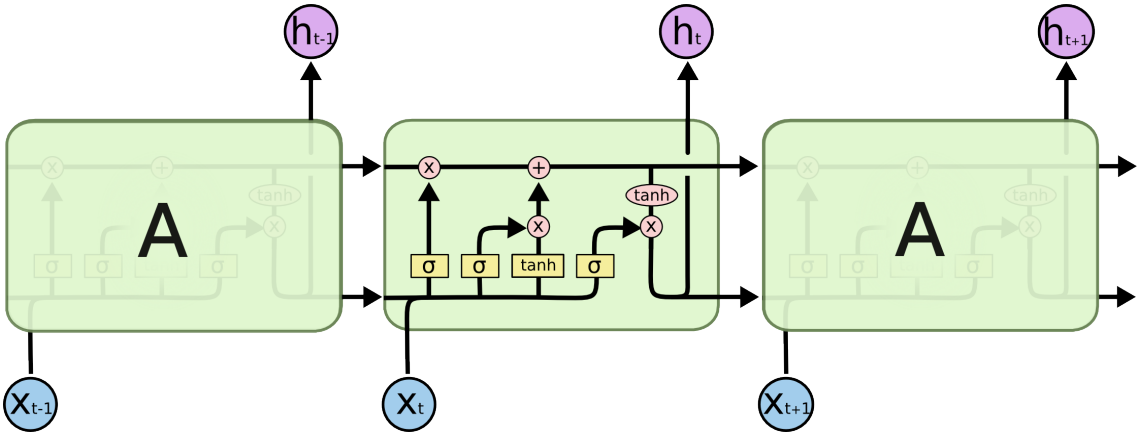


Figure 4: LSTM Neural Network architecture. Source: (Olah, 2015)

4.3.3 Long Short-Term Memory

LSTM is explained crudely, as explaining it from top to the bottom is out of scope for this research, for interested readers I refer to Hochreiter and Schmidhuber (1997) for a more technical explanation. LSTM-cells use four gates to retain memory and focus on the main classification task, separating these two tasks enables the model to learn weights that approximate word dependencies between words further apart in a sequence. For this mean, the LSTM has one hidden state and one cell state (Hochreiter & Schmidhuber, 1997). The cell state is responsible for retaining memory, while the hidden state focuses on the classification task.

The architecture of a LSTM cell is given in Figure 4, the cell state is represented by the horizontal line running through the top of the diagram and the hidden state by h_t. The gates are highlighted in yellow and are mathematically formulated from left to right:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),\ where\ \sigma = \frac{e^z}{e^z + 1}\,, \tag{9a}$$

$$i_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_i)\,, \tag{9b}$$

$$a_t = \tanh(W_h \cdot [h_{t-1}, x_t] + b_a),\ where\ \tanh = \frac{e^z - e^{-z}}{e^z + e^{-z}}\,, \tag{9c}$$

The equations of 9 represent the mechanisms to alter the cell state. Equation 9a is called the *forget* gate and outputs a value between 0 and 1 to determine which elements of the cell state to forget. The equation of 9b is the input gate and decides which values are updated by a sigmoid, equation 9c is the addition gate and creates a vector of new values ranging between 1 and -1 to add to the cell state. The cell state is updated as follows: first, the vector of the forget gate (9a)

is Hadamard multiplied ($\odot$) with the previous cell state and then added up by the result of the Hadamard multiplication of the input gate (9b) and addition gate (9c); formally put:

$$C(t) = f_t \odot C_{t-1} + i_t \odot a_t \,, \tag{10}$$

The hidden state that solely focuses on classification is determined by the output gate. The output gate determines which values of the cell state are used for calculating the hidden state; formally put:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_i) \,. \tag{11}$$

The cell state is first put through a tanh to rescale the values to range between -1 and 1 and then Hadamard multiplied by the output gate; formally put:

$$h(t) = o_t \odot \tanh(C_t) \,, \tag{12}$$

Applying LSTM cells increases memory capability, (partially) solving the problems caused by vanishing gradients. By separating the cell state and the hidden state the backpropagation is done separately for each state. The model thus has to update more parameters, which increases the capacity of the model but also increases the risk of overfitting. Merity, Keskar, and Socher (2017) wrote a paper that evaluates how to best regulate an LSTM architecture, which forms the basis on which ULMFiT is developed (Howard & Ruder, 2018).

4.3.4 Regularizing LSTM: AWD-LSTM

The model Merity et al. (2017) developed is the AWD-LSTM model, which is the best regularized LSTM architecture to date. AWD-LSTM uses Dropout, DropConnect, Weight Decay, Activation Regularization (AR), and Temporal Activation Regularization (TAR). The amount of regularization can be adjusted between epochs for all methods, meaning the regularization can be applied when overfitting is observed, potentially accelerating convergence.

Activation Regularization is similar to weight decay, as the weighted mean of the square of all activations of the last layer is added to the loss. This prevents the activations of the layers from increasing if it does not decrease the loss by a considerate amount, forcing the parameters to remain small. The weighting is a tuneable hyperparameter.

Temporal Activation Regularization is applied to make the difference between two consecutive activations as small as possible to stabilize the model. To this end, the squared difference

between the activations of the current time step and the previous time step is multiplied with a β and added to the loss function.

4.3.4.1 Dropout & DropConnect

AWD-LSTM uses dropout at multiple layers. It is applied to the embedding layer, input layer, weights, and hidden states. Finding the right ratio between the different dropouts is a tedious task. Howard and Ruder (2018) have found empirically good ratios for text classification, where their magnitude is adjusted through a dropout multiplier.

Applying naive-classical dropout techniques on the input and embedding layer disrupts the ability of the LSTM to learn long-term dependencies (Merity et al., 2017). Instead, DropConnect is applied, where elements are zeroed in a way that the same parts of hidden units are disabled throughout the sequence of every layer, to keep the masked weights consistent for each forward and backward pass within one epoch (Wan, Zeiler, Zhang, Cun, & Fergus, 2013). DropConnect is applied on all dropouts in ULMFiT, except for Output Dropout.

Embedding Dropout applies DropConnect on the vectors in the Word Embeddings, effectively setting some elements in the Embeddings vector to zero.

Input Layer Dropout applies DropConnect in the input vector, effectively setting some elements in the input vector to zero, which means some words are entirely masked.

Weight Dropout applies DropConnect on the weights of the hidden-to-hidden matrices. The hidden-to-hidden matrices are the matrices that are of size $input \times 1150$, where some elements are randomly set to 0.

Hidden Dropout applies DropConnect on the output of a LSTM layer to another. It zeros out the inputs of the stacked LSTM layers. For illustration, as the AWD-LSTM of ULMFiT consists of three layers, for some words the output from layers 1 to 2 or from 2 to 3 is randomly zeroed.

Output Dropout is only applied in the Language Model, where it zeroes final sequence outputs from one layer before feeding it to the classification head. This way the different layers better learn to cooperate.

4.4 AWD-LSTM: ULMFiT

This section focuses on the implementation of AWD-LSTM and its extensions in ULMFiT, being: the applied training cycle, Discriminative Fine-tuning, Gradual Unfreezing, and Concat Pooling.

4.4.1 Optimization: Training Cycle

Fit Flat Cosine schedules the learning rate so that the model is fit at the learning rate set and then applies a cosine annealing near the end of the fitting cycle. The fitting cycle is determined by the number of steps in each epoch multiplied by the number of epochs. The effect of Fit Flat Cosine training on the learning rate over multiple steps is shown in Figure 5, where the $x - axis$ is for 249 batches in 25 epochs, resulting in 6225 steps.

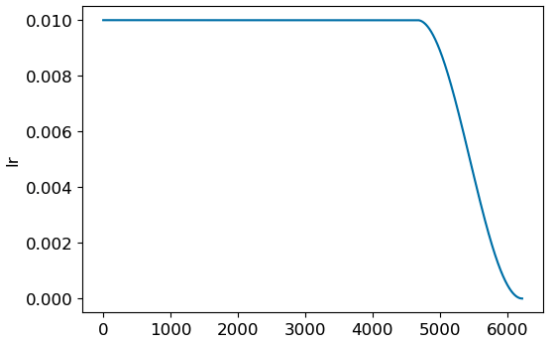


Figure 5: Learning rate adaptation over training steps with Fit Flat Cosine Hyperparameter scheduler

The main advantage of this learning rate schedule is that the increases and decreases in validation loss evolve relatively constant, which is beneficial for the Early Stopping Callback as it makes the improvement of the model more predictable. A downside is that the cosine annealing may not be utilized when the training cycle is stopped too early. The learning rate is decreased manually when the model has stopped because the training loss did not decrease.

Discriminative fine-tuning was introduced by Howard and Ruder (2018) and builds on the idea that different layers capture different types of information and thus should be fine-tuned to different extents. Howard and Ruder (2018) have empirically determined that the learning rate must decrease by a factor of 2.6 for each layer from top to bottom.

`FastAI` lets you pass a Python slice object anywhere that a learning rate is expected. A slice converts a learning rate into a log-uniform collection from its beginning to its end. The first value passed is the learning rate in the earliest layer of the Neural Network, and the second value is the learning rate in the final layer. The layers in between have learning rates that are multiplicatively equidistant throughout that range.

Learning rates are plotted against the losses, as introduced by (Smith, 2015) and further

explained in Appendix B. Practically, this way the learning rate would be set too low, leading to longer training times. For this reason, the procedure as explained in Appendix B is applied with a minimum value for the learning rate, depending on the place in the training cycle.

Gradual Unfreezing consists of freezing all the parameters of the model and unfreezing them layer by layer from last to first. Howard and Ruder (2018) found that this improves model performance, and attributed it to the fact that the last layer contains the least general knowledge and that it reduces the risk of catastrophic forgetting. When all layers are unfrozen, the whole model is optimized altogether.

Model Resetter ensures that the hidden state and the cell states are reset after each epoch through a call back. When fitting the model, the states must be reset to 0 to start with a clean slate. This prevents the model from saturating and subsequently failing to improve.

4.4.2 Architecture

The Word Embeddings contain vectors of 400 numbers for each word (11,008 words on 100% of the data). These vectors pass three stacked LSTMs. The first LSTM transforms the Word Embeddings vector into a vector of size 1150, the second LSTM also outputs a vector of 1150 and the third LSTM transfers it back into a vector of 400. The dimensions of W_* and b_* in equations 9 to 11 is thus 1150 for the first and second LSTM, the dimensions of W_* and b_* in the third LSTM are 400. The total amount of trainable parameters on 100% of the data is $24,625,808$ for the Language Model and $24,680,070$ for the final classification model. This number may seem impressive to the average reader, but not all parameters are moving. As the parameters in the earlier layers of the model contain general knowledge, the parameters that are moving are only in the last layers. Furthermore, given that the chain rule inevitably emphasizes updates in the last layers, the only parameters that should be moving are in the last layers. This further underscores the benefits of pretraining, as the pretrained parameters containing general knowledge do not have to move as they are accurately estimated on large amounts of data.

The encoder component of the AWD-LSTM model used in ULMFiT is visualized in Figure 13 in Appendix C. This encoder is used in both Language Modeling and classification, only the decoder differs. In language modeling, the vector of 400 (the last hidden state) is multiplied by a decoder matrix of $400 \times 11,008$, after which a softmax function transforms all values into probabilities for each word (vector of shape $1 \times 11,008$).

In classification, the last hidden state is concatenated with the max-pooled and the mean-pooled hidden state. The max-pooled hidden state is the vector containing the largest value for each of the 400 features of all hidden states in the sequence. The mean-pooled hidden state contains the average of all features in the hidden states in the sequence. This is called **Concat Pooling**, which prevents losing information(Howard & Ruder, 2018). The result is a vector of 1×1200 features.

This vector is then passed into a layer of 50 ReLU activations, which reduces the dimension to 50, then another layer with 50 ReLU's rescales the output to 68, after which a Softmax layer rescales the outputs to obtain a probability for each class.

4.4.3 Applied Training Procedure

The language model and the classification model both apply all optimization techniques as mentioned in this section (4.4), only the values differ. The AR and the TAR for both models are set to 1 and 2 respectively, which is the default value in ULMFiT (Howard & Ruder, 2018). This section further explains the values used for estimating both models.

4.4.3.1 Language Model

The default dropout probabilities for the Language model are as follows: 0.1 for output dropout, 0.15 for hidden dropout, 0.25 for input dropout, 0.02 for embedding dropout, 0.2 for weight dropout.

First, the pretrained model from van der Burgh and Verberne (2019) is matched against the vocabulary in the training set. In the beginning, the minimum learning rate is set to 0.01 and a dropout multiplier of 0.3 is applied. When the model stops improving, the learning rate is decreased to 0.005 and the weight decay increased to 0.25. Then, the dropout multiplier increases to 0.6 and 1. When the dropout multiplier is equal to 1, the minimum learning rate is also decreased to 0.0001.

4.4.3.2 Classification Model

The default dropout probabilities for the classification model are as follows: 0.4 for output dropout, 0.3 for hidden dropout, 0.4 for input dropout, 0.05 for embedding dropout, 0.5 for weight dropout.

First, the encoder of the Language Model is loaded. The dropout multiplier is set to 0.5 and the minimum learning rate is first set to 0.005. After gradual unfreezing, the learning rate is to 0.001 and a for loop with different values for weight decay is applied. Then the dropout multiplier is set to 1 and gradual unfreezing is applied. When all layers are unfrozen the learning rate is decreased to 0.0001 and the weight decay is again for looped in ascending order to 1.

4.5 Transformers: BERT and RoBERTa

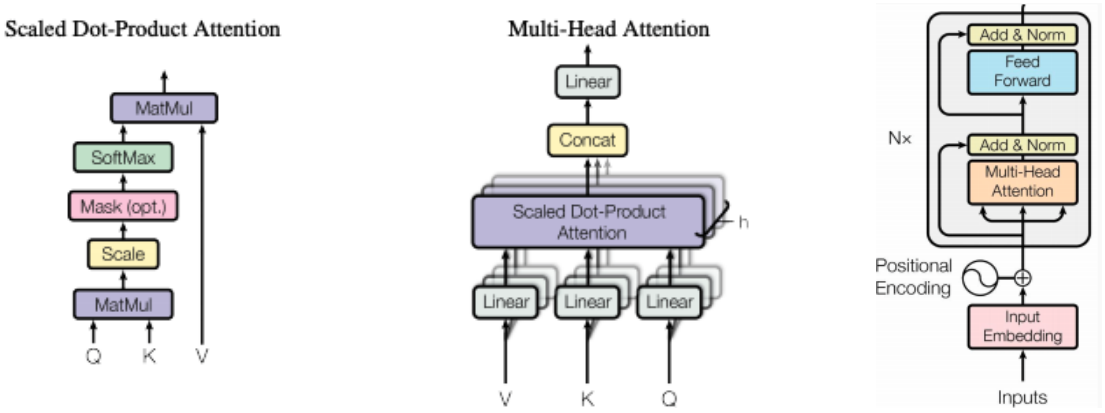
This section is about the architecture and the training cycle after pretraining of BERT and RoBERTa. As mentioned in section 2.5.2, only the pretraining procedure differs between BERT and RoBERTa. Therefore, in this section, the mentioning of BERT also refers to RoBERTa, as it explains both models.

BERT is a bidirectional variation on the Transformer architecture introduced by Vaswani et al. (2017) and only utilizes the encoder part in text classification. This section thus focuses on the foundation of Transformers in BERT, and how BERT extends beyond them.

BERT consists of Positional Encoding (section 4.5.1) and 12 encoder layers. Each encoder layer consists of two sub-layers: one Multi-Head Attention sub-layer (MHA), two Add and Normalisation operations, and one Feed Forward Neural Network (FFNN) sub-layer. MHA is further explained in section 4.5.2. FFNN is further explained in section 4.5.4.

The vector of each word with size d in the Word Embeddings flows through its own path in the encoder. The encoder receives a list of word vectors as input. The word dependencies are incorporated in the Attention layers, but not in the FFNN layers. As word relations are not incorporated in the FFNN layers, multiple words can be processed at the same time, thus allowing for parallelization in those layers

Figure 6: Methodology of the encoder stack in transformers and the attention mechanism. Obtained from Vaswani et al. (2017).

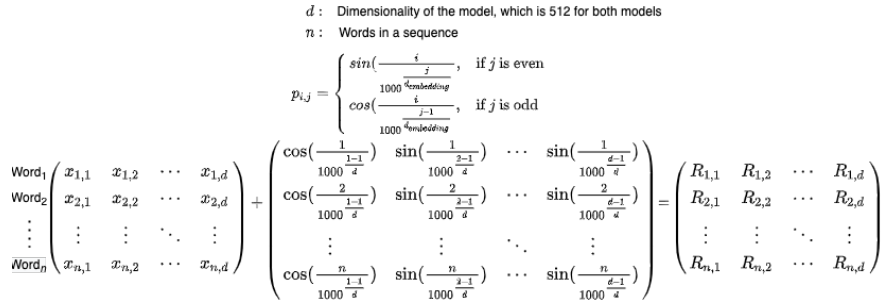


4.5.1 Positional Encoding

Positional Encoding adds a matrix consisting of elements whose values are based on the place of the words in the sequence i and the element in the vector j. The model learns these patterns to

encode word order. Figure 7 visualises positional encoding. After positional encoding, the words are put through Multi-Head Attention.

Figure 7: Positional Enconding in BERT and RoBERTa



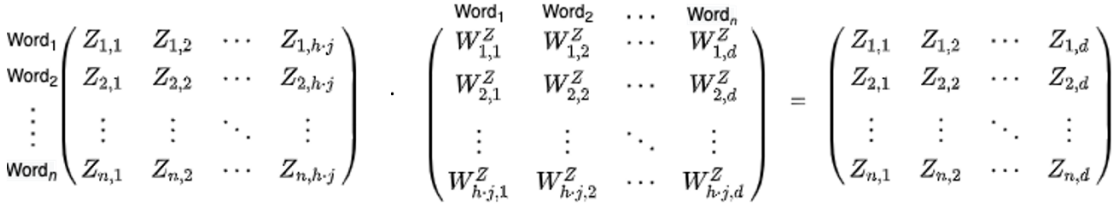
4.5.2 Multi-Head Attention

The Multi-Head Attention (MHA) sub-layer consists of 12 Self-Attention operations. Self-Attention takes in one Query ($\boldsymbol{Q}$), one Key ($\boldsymbol{K}$), and one Value ($\boldsymbol{V}$) matrix, each containing one vector for each word. The $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ matrices are obtained by multiplying the input vector of each word with a weight matrix that is trained. The $\boldsymbol{Q}$ and $\boldsymbol{K}$ matrices are multiplied, divided with $\sqrt{d}$, scaled by a Softmax, and finally multiplied with $\boldsymbol{V}$. The $\boldsymbol{Q}$ and $\boldsymbol{K}$ vectors generate the amount of attention a word obtains in $\boldsymbol{V}$, resulting in a new matrix $\boldsymbol{Z}$. Self-Attention is formulated as:

$$Softmax(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{d}}) \cdot \boldsymbol{V} = \boldsymbol{Z} \tag{13}$$

Self-Attention is further explained and illustrated in Appendix D. The outputs of the 12 Self-Attention operations in Multi-Head Attention are concatenated in one final matrix. This enables the model to focus on multiple positions, as it creates multiple representation subspaces. As a result, $12 \cdot \boldsymbol{Z}$ matrices are obtained. These are concatenated and multiplied with a weight matrix, to obtain a final matrix $\boldsymbol{Z}$ that can be put into the FFNN sub-layer. The mechanism behind Multi-Head Self-Attention is visually represented in Figure 8.

Figure 8: Mathematical representation of Multi-Head Attention



4.5.3 Add & Normalisation

For both MHA and FFNN the original input is added to the output of its sub-layer and then normalized. Before adding the input and output matrices, Dropout is applied on the output matrix. The 'Add & Norm' step is visualized in Figure 9 below.

$$\begin{pmatrix} Z_{1,1} & Z_{1,2} & \cdots & Z_{1,d} \\ Z_{2,1} & Z_{2,2} & \cdots & Z_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n,1} & Z_{n,2} & \cdots & Z_{n,d} \end{pmatrix} \cdot \begin{pmatrix} 1_{1,1} & 0_{1,2} & \cdots & 0_{1,d} \\ 0_{2,1} & 1_{2,2} & \cdots & 1_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1_{n,1} & 0_{n,2} & \cdots & 1_{n,d} \end{pmatrix} = \begin{pmatrix} Z_{1,1} & 0_{1,2} & \cdots & 0_{1,d} \\ 0_{2,1} & Z_{2,2} & \cdots & Z_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n,1} & 0_{n,2} & \cdots & Z_{n,d} \end{pmatrix}$$

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix} + \begin{pmatrix} Z_{1,1} & 0_{1,2} & \cdots & 0_{1,d} \\ 0_{2,1} & Z_{2,2} & \cdots & Z_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n,1} & 0_{n,2} & \cdots & Z_{n,d} \end{pmatrix} = \begin{pmatrix} x + Z_{1,1} & x + 0_{1,2} & \cdots & x + 0_{1,d} \\ x + 0_{2,1} & x + Z_{2,2} & \cdots & x + Z_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x + Z_{n,1} & x + 0_{n,2} & \cdots & x + Z_{n,d} \end{pmatrix}$$

For each element in the final matrix the following normalisation is applied: $\frac{(x - \bar{x})}{(\sigma + \epsilon)}$

Figure 9: Add and Normalisation

The formula of normalization as stated in the bottom of Figure 9 ($\frac{x-\bar{x}}{\sigma+\epsilon}$) shows that for each element in the matrix the average is subtracted and then divided by the standard deviation added up with epsilon. Epsilon is a hyperparameter to tune.

4.5.4 Feed Forward Neural Network

The Feed-Forward Neural Network (FFNN) is visualized in Figure 10. The FFNN follows the structure as explained in section 4.2. The FFNN layers use the ReLU activation. The inner layer has a dimensionality of 768, the dimensionality of the input and output is 512.

Each word in the sequence passes its own FFNN, the FFNN consists of two layers with dropout on the fist layer, coming down to:

$$FFNN(x) = max(0, x \odot dropout(W_1) + b_1)W_2 + b_2$$

Resulting in: $x = x + Z_{1,1},\ x + 0_{1,2} \cdots x + 0_{1,d}$,

$$\boldsymbol{W_1} = \begin{pmatrix} 0^1_{1,1} & W^1_{1,2} & \cdots & 0^1_{1,d_{ff}} \\ W^1_{2,1} & 0^1_{2,2} & \cdots & W^1_{2,d_{ff}} \\ \vdots & \vdots & \ddots & \vdots \\ 0^1_{d,1} & W^1_{d,2} & \cdots & 0^1_{d,d_{ff}} \end{pmatrix}, \quad \boldsymbol{W_2} = \begin{pmatrix} W^2_{1,1} & W^2_{1,2} & \cdots & W^1_{2,d} \\ W^2_{2,1} & W^2_{2,2} & \cdots & W^2_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ W^2_{dd_{ff},1} & W^2_{d_{ff},2} & \cdots & W^2_{d_{ff},d} \end{pmatrix}$$

With: $d_{ff} = 768$

Output: $FFNN(x) = R_{1,1},\ R_{1,2} \cdots R_{1,d}$

The resulting vectors $(R_{1,1},\ R_{1,2} \cdots R_{1,d})$ are then appended to form a matrix, which is passed through an other encoder layer, or the classification part, after another round of adding the input and applying normalisation

Figure 10: Feed Forward Neural Network layer in BERT and RoBERTa

4.6 Pretraining procedure

As mentioned in section 2.5 BERT is pretrained bidirectionally by using masking through auto-encoding. To prevent the model from failing to estimate parameters for words that are masked, the masked words are not always hidden. The tokens are hidden 80% of the time, replaced with a random token 10% of the time, and the actual token 10% of the time (Devlin et al., 2018). 'Next-Sentence Prediction' is done by letting the model must predict whether a sentence (B) follows after another sentence (A), where sentence B is the actual next sentence 50% of the time, which is then labeled as `IsNext` and `NotNext` otherwise.

4.6.1 Learning cycle

The hyperparameters that are tuned for BERT and RoBERTa are the learning rate, dropout, and weight decay. The hyperparameter for epsilon is set to 1^{-8}, the scheduler used is `linear schedule with warmup` with 0 warming up steps. The optimization algorithm is Adam as explained in section 4.2.4. The recommended batch size for both models is 32 (Delobelle et al., 2020; Devlin et al., 2018), due to GPU memory restrictions the batch size is set to 12 as the program otherwise crashes.

4.6.1.1 Learning rate

The default learning rate for BERT is 1^{-5}, which is small. However, considering that the learned contextualized embeddings are not initialized from zero, but pretrained on an enormous amount of data, small adjustments make sense. Setting the learning rate to 1^{-2} or 1^{-3} increased the error rate on the first two epochs. The learning rate is set to 1^{-4} and is divided by a factor of ten when both the training and validation loss did not improve.

4.6.1.2 Dropout & Weight Decay

BERT is prone to overfitting when the data is limited, as empirically found by Devlin et al. (2018). There has been limited work on how to regularize BERT, as there is only one article published that regularises BERT. H. D. Lee, Lee, and Kang (2020b) developed a Neural Network that automatically optimizes BERT. For the sake of this research, the regularisation terms are put to their maximum value, as it otherwise overfits quickly. I refer to H. D. Lee et al. (2020b) for further research.

The dropout values on the attention heads and the hidden states are thus set to 0.5, which is equal to the maximum of regularisation as explained in section 4.2.3 (Baldi & Sadowski, 2013).

The weight decay in BERT and RoBERTa increases by 0.25 if the validation loss failed to improve after an epoch, with a maximum value of 1.

4.7 Rotation Estimation

Stratified K-Fold Rotation Estimation can be applied in statistics to optimize the generalization (i.e. prediction performance). The data is split into $K - folds$, where each of k folds is used for validation once and the other $k-1$ for training. The K resulting models then predict the probability that an observation in the test set belongs to one of the classes, the predicted probabilities are then averaged and the highest resulting probability determines to which class an observation belongs. The advantages are twofold: the randomness of the split is removed and the data is fully used.

For all models, the effect of 5-Fold Stratified Rotation Estimation is considered. The decision about the number of folds is arbitrary but most often set to either 5 or 10. Considering that an epoch with BERTje and RobBERT takes 5 minutes for only 20% of the data, 5-Folds is chosen with the main argument of computational feasibility.

4.8 Model Ensembling

Figure 11 visually represents how Model Ensembling obtains the final predictions. The Model Ensembling method applied uses the predictions of Level 1 Models as the inputs for a Level 2 Model, where the Level 2 Model combines the predictions to obtain the final prediction (Sakkis et al., 2001).

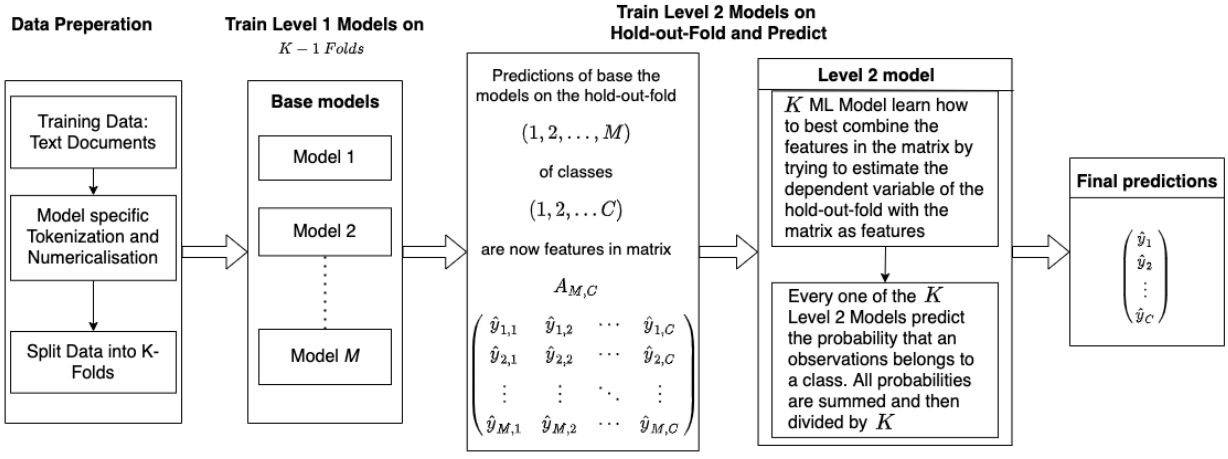


Figure 11: Methodology of Model Ensembling

The Level 1 Models are pretrained on a stratified subset of the data to obtain their parameters. The Level 1 Models then predict the class of every observation in a 'hold-out-fold', which in this case is the validation set. The predicted probabilities are stored in matrix $A_{M,C}$. The Level 2 Model then learns how to best combine these predicted probabilities into a final prediction. The Level 2 Model cannot be learned on observations the Level 1 Models used to estimate their parameters, as the Level 2 Model then solely focuses on the best performing model instead of combining all models.

Model Ensembling can add value when the error terms of the Level 1 Models deviate, as the Level 2 Model can then learn how to best combine the Level 1 Models. Therefore, it is logical to assume that Model Ensembling can increase prediction performance with multiple Neural Networks as Level 1 Models, as different Neural Network architectures approximate a function differently (Goodfellow et al., 2016). The Level 1 and 2 Models can be any Machine Learning model. The ML models considered as Level 2 Models are Multinomial Logistic Regression and Linear Discriminant Analysis, as it is intuitive to linearly combine the predictions of the Level 1 Models into final predictions.

Multinomial Logistic Regression constructs a linear predictor function that gives probability scores for a class dependent on the linearly combined explanatory variables, thus estimating the conditional distribution of the response class given the predictors. Logistic Regression is supposed to converge faster than LDA (James et al., 2021). LDA is often preferred over Multinomial Logistic Regression as it would lead to more stable parameters (James et al., 2021). The loss function as stated in formula 4 is optimized for both LDA and Logistic Regression. For both Logistic Regressions as LDA for $L-1$ classes $p+1$ parameters must be estimated, which is equal to $((68-1)\cdot(68\cdot 3+1) =)$ 18291 parameters. The mathematics behind Multinomial Logistic regression are explained in Appendix E, for LDA in Appendix F.

4.9 Model Evaluation

The prediction performance of the models is evaluated with the macro-weighted F_β-score, macro-weighed refers to the procedure of weighting the F_β-score for each class by its amount of observations. The F_β-score consists of precision and recall. Precision penalizes False Positives and recall penalizes False Negatives. A False Positive occurs when the model falsely predicts a class, a False Negative occurs when the model falsely does not predict a class, alias Type I and Type II errors.

The F_β-score captures both errors and is calculated as follows:

$$F_\beta = \frac{(\beta^2+1)\cdot precision \cdot recall}{(\beta^2 \cdot precision) + recall}, \; where \; Precision = \frac{True\,Positive}{True\,Positive \; + \; False\,Positive}\,, \\ \& \; Recall = \frac{True\,Positive}{True\,Positive \; + \; False\,Negative}\,, \tag{14}$$

The β determines the desired weight distribution between precision and recall, as for $\beta < 1$ recall is favored, for $\beta > 1$ precision is preferred, and with $\beta = 1$ they are equal. $\beta = 1$ is used, as they are considered equally important. Based on a more thorough legal understanding of the classes, the weights can be adjusted for some classes. This is for further research.

In determining the F_1-score, the classes *'considerans-present'* and *'compliance-present'* are disregarded, as they constitute leftover classes that are legally uninteresting. The F_1-score for the whole model is calculated by calculating the macro-weighted mean of the F_1-scores of the other classes, following the formula:

$$macro - weighted\, F_1 = \frac{n_1}{N} \cdot L_1F_1 + \frac{n_2}{N} \cdot L_2F_1 + \cdots + \frac{n_j}{N} \cdot L_jF_1\,, \tag{15}$$

Where L_jF_1 is the F_1-score of the jth class, with observations n_j of total observations N. Calculating the total F_1-score this way accounts for the class imbalance in the data. Again, with more thorough legal knowledge certain classes can be given a larger weight if they are considered to be more important.

The best-performing models are also compared by a Confusion Matrix. A Confusion Matrix lies at the foundation of the F_1-score, as it compares the predicted classes against the actual classes, identifying True Positives, True Negatives, False Positives, and False Negatives. By looking at this Matrix, it is deducible how the model performs for the 68 classes individually.

5 Results

In this section, the performance of the models is compared on different subsets of data, where the differences in performance are explained based on the theory provided in the previous sections. First, the general prediction performance of the models as measured by the macro-weighted F_1-score is compared, then the model performance per class is discussed.

5.1 General Performance

As can be seen in Table 1, all models perform comparably well on different amounts of data. Figure 12 shows that the differences in prediction performance between the models are not large and that

Model Ensembling can enhance prediction performance.

The prediction performance of the models on 20% of the data is visually represented in Figure 12. The importance of pretrained contextualized Word Embeddings for limited data is underscored by the fact that the right-to-left ULMFiT performs the worst. The importance of pretraining does however fade when enough data is available, which is illustrated in Figure 12 as right-to-left ULMFiT overtakes both BERT and RobBERT on 100% of the data.

Rotation Estimation with a majority vote improves prediction performance for all models on all data levels except for BI-LSTM ULMFiT on 100% of the data. This could be due to the amount of averaging required to obtain the final predictions. Figures 20 to 24 in Appendix G and Table 2 furthermore illustrate that Rotation Estimation works the best for the Transformer models as the data increases compared to ULMFiT.

Table 1 shows that 'Next-Sentence Prediction' applied in BERTje is more valuable than the increment in data used for pretraining RobBERT for text classification of DPAs, as BERTje and Rotation Estimated BERTje outperform RobBERT and Rotation estimated RobBERT on all data levels.

The Level 2 Models in Model Ensembling failed to estimate its parameters until 80% of the data is used, as it only improves over Rotation Estimated BERTje after that threshold. Notably, the LDA only outperforms Logistic Regression when 100% of the data is used, meaning that LDA requires more data than Logistic Regression. Simultaneously, it signals that LDA does outperform Logistic Regression when enough data is available, which is further illustrated by Figures 25 and 26 in Appendix G as the slope of the increase in prediction performance for LDA remains the largest compared to Logistic Regression.

Table 1: Macro-weighted F1-score of the Models for different amounts of data (in Percentage (%)

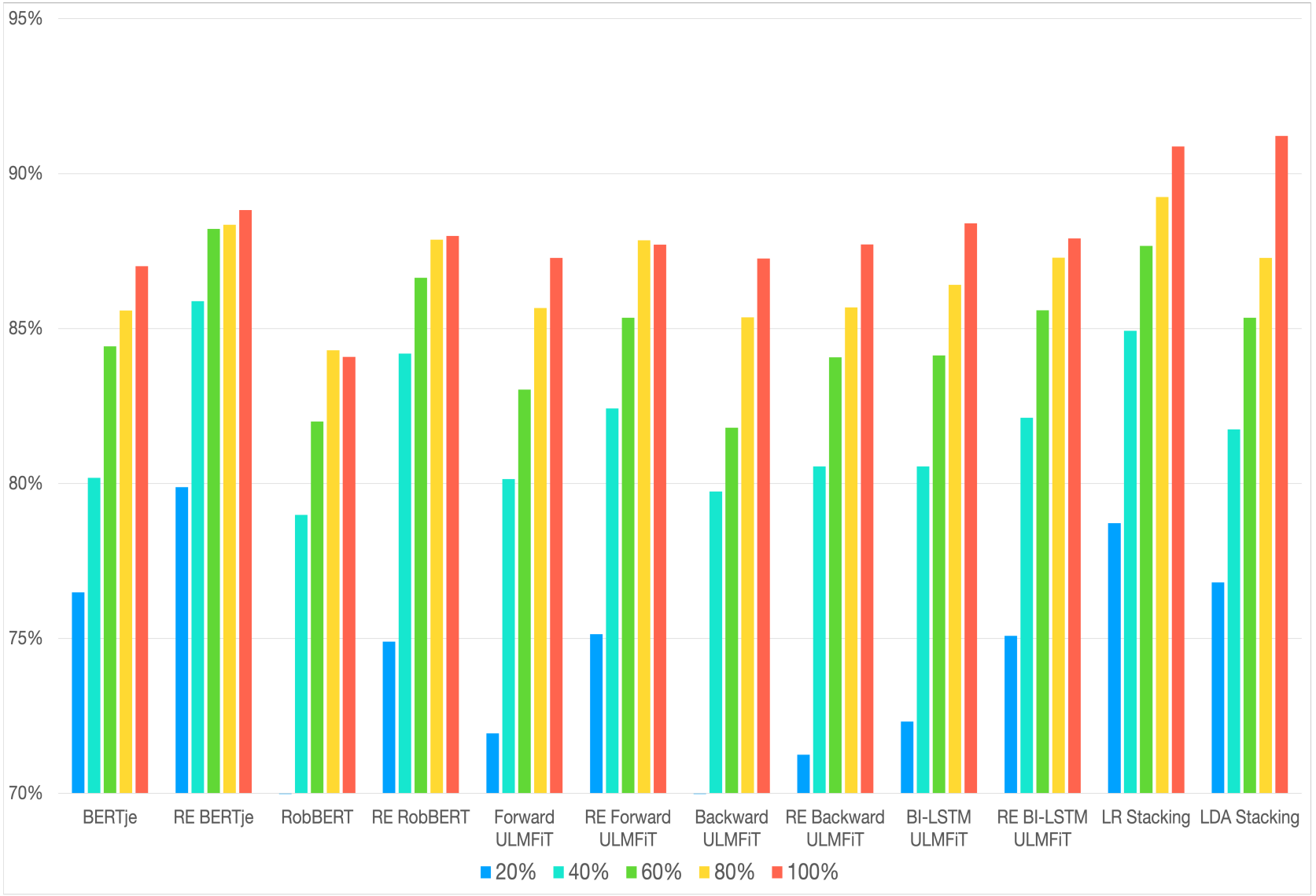
Data Level	BERTje	RE BERTje	RobBERT	RE RobBERT	F-ULMFiT	RE F-ULMFiT	B-ULMFiT	RE B-ULMFiT	BI-ULMFiT	RE BI-ULMFiT	LR Ensemble	LDA Ensemble
20%	76,490	79,882	69,718	74,893	71,934	75,134	65,216	71,246	72,316	75,086	78,720	76,809
40%	80,181	85,883	78,984	84,1912	80,140	82,420	79,740	80,552	80,552	82,119	84,929	81,749
60%	84,428	88,214	81,998	86,636	83,033	85,352	81,797	84,075	84,130	85,587	87,669	85,347
80%	85,585	88,348	84,297	87,871	85,665	87,846	85,366	85,686	86,410	87,284	89,244	87,283
100%	87,011	88,827	84,083	87,991	87,279	87,707	87,262	87,712	88,398	87,907	90,874	91,217

The abbreviations are as follows: 'RE' stands for Rotation Estimated, 'F' for Forward, 'B' for Backwards and 'BI' for Bidirectional

Table 2: Added Value of Prediction Performance of Rotation Estimation compared to the model on its own based on the natural logarithm of change ($D = 100 \cdot \ln(\frac{V_{CV}}{V_{solo}})$)

Data Level	BERTje	RE RobBERT	RE F-ULMFiT	RE B-ULMFiT	BI-ULMFiT
20%	4,339	7,159	4,352	8,842	3,758
40%	6,867	6,384	2,806	1,013	1,926
60%	4,387	5,502	2,755	2,747	1,717
80%	3,177	4,153	2,514	0,375	1,007
100%	2,066	4,544	0,489	0,515	-0,5575

Figure 12: Macro-weighted F1-score for all data levels for all different models



5.2 Summarized Confusion Matrices

As shown in Figure 12 the best performing models as measured by the macro-weighted F_1-score are Ensembling with LDA, Ensembling with Logistic Regression, Rotation Estimated BERTje, BI-LSTM ULMFiT and Rotation Estimated RobBERT. The differences are not large, but Ensembling convincingly adds value. The macro-weighted F_1-score provides no insight into how the models perform in the individual classes, which is why summaries of the confusion matrices of all models are given in Tables 4 to 7 in Appendix H.

Table 4 and 5 in Appendix H shows that BERTje and RobBERT fail to predict the same 11 of 68 classes. They fail to predict *'datalek-12uur', 'datalek-48uur', 'datalek-36uur', 'datalek-72uur', 'datalek-8uur', 'aardsoort-inclausule', 'looptijd-eigen', 'retourkeuze-vw', 'verzoeken-zelf', 'wijzigingen-alleenvv'* and *'wijzigingen-alleenvw'*, which only have 14, 64, 26, 16, 2, 9, 6, 9, 30 and 19 observations, respectively.

Table 6 in Appendix H shows that BI-LSTM ULMFiT fails to predict 5 out of the 68 classes, being *'aardsoort-inclausule', 'datalek-12uur', 'datalek-72uur', 'datalek-8uur'* and *'retour-keuze-vw'*, with, respectively, 9, 14, 16, 2 and 9 observations. The BI-LSTM ULMFiT model needs fewer observations to distinguish between these clauses. Notably, the *'datalek'*-class performed better, which could be explained by the fact that such a clause follows a standard formula where the amount of hours (i.e. 'uur') is explicitly stated in a similar place in a sequence, causing the recurrent structure of ULMFiT to outperform the Positional Encoding used in Transformers.

Table 7 shows that Model Ensembling with Logistic Regression fails to predict 7 classes, whereas Table 8 shows that Model Ensembling with LDA fails to predict 3 classes. LDA thus outperforms Logistic Regression on both macro and micro levels when enough data is available.

6 Conclusion

This thesis tried to answer the question: *"How can we successfully classify clauses in a Dutch Data Processing Agreement with limited data?"*. Previous research showed that BERT, RoBERTa and ULMFiT are the models that provide the potential to yield high performance. For this research, 600 different DPAs were labeled by independent interns and lawyers of ICTRecht. Moreover, five sub-questions were formulated to define the appropriate research.

To answer sub-question 1, in the preprocessing of DPAs there should be no filtering based on the occurrences of words, as typos are far from custom and every word can have meaning. Adding the

headers into the sequence increases model performance. At last, codes based on text conversions can be filtered through regular expressions.

Regarding sub-question 2, which was about the identifying the current state-of-the-art algorithms and evaluating their differences. Previous research found that the current state-of-the-art text classification algorithm available for Dutch texts is ULMFiT, BERT and RoBERTa. The main theoretical differences between ULMFiT, BERT, and RoBERTa, is that BERT and RoBERTa only differ in pretraining while ULMFiT is an entirely different model. BERT and RoBERTa are both Transformers that only contain the encoder part for text classification. They apply Positional Encoding to encode word order and Multi-Head Attention to learn the relations and relative importance among words. They are both prone to overfitting. RoBERTa is pretrained on more textual data and does not use the 'Next-Sentence Prediction'-procedure. ULMFiT, on the other hand, is a Recurrent Neural Network with LSTM cells that is fully optimized according to the AWD-LSTM model and is pretrained on a smaller corpus of data than both BERT and RoBERTa. ULMFiT encodes the word order through its recurrent architecture and learns relations and relative importance among words by using the cell state in the LSTM. Furthermore, ULMFiT uses more regularisation, being Activation Regularisation, Temporal Activation Regularisation, DropConnect, Dropout, and Weight Decay, while BERT and RoBERTa only apply Dropout and Weight Decay. At last, BERT and RoBERTa are inherently bidirectional, while the ULMFiT used has to average the predictions of two separate models to become bidirectional.

To continue to sub-questions 3 and 4, on 20% and 60% of the data BERTje was the best performing model but is overtaken by BI-LSTM ULMFiT on 40%, 80%, and 100%. RobBERT was one of the worst-performing models, indicating that Next-Sentence Prediction adds value to the Contextualized Word Embeddings of the BERT architecture. Rotation Estimated BERTje was the best performing up until 80% of the data, after which it was overtaken by Model Ensembling.

At last, sub-question 5, was about the relevance of Model Ensembling. Model Ensembling only outperformed all models at 80% of the data, implying the validation set did not have enough observations before this threshold to correctly estimate its parameters. Furthermore, it was found that LDA is the best Level 2 Model if there is enough data. LDA had the best macro-weighted F_1-score on 100% of the data and was able to correctly classify classes that all Level 1 Models left out. Logistic Regression outperformed LDA up until 100% of the data, indicating that it is a better model for limited data.

In conclusion, all Figures showed that Rotation Estimated BERTje outperformed every Level

1 Model on every level of data. BERTje outperforming RobBERT signals that the 'Next-Sentence Prediction' is valuable for estimating parameters in text classification of legal documents. BERTje did, however, leave out 11 classes whereas ULMFiT only left out 5, where the deficit in the classes *'datalek'* is attributed to the lesser competence of encoding words positions through Positional Encoding compared to the recurrent structure in RNN's. Furthermore, BERTje is pretrained on a vast amount of data, making it an unfair competition on smaller amounts of data. Only when an ULMFiT model is pretrained on an as large amount of data, a proper comparison can be made. The BERT architecture is however faster in computation time if the GPU memory is large enough, as it allows for more parallelization. Ultimately, prediction performance can be improved by Model Ensembling, where Logistic Regression is a more appropriate Level 2 Model than LDA if the data is limited, but eventually, LDA does obtain better parameters and then outperforms Logistic Regression.

7 Discussion

As this is one of the first researches of NLP in the legal domain, there are still a lot of territories to explore, allowing for suggestions for future research. This section thus discusses the limitations of this research and gives suggestions for future research.

To begin with, some classes have few observations making it more difficult to train and evaluate parameters that distinguish those classes. As a result, BERT might eventually perform comparable or better if there is more data in both the test and training set for those observations. Data augmentation could also enhance model performance as it artificially creates those observations. Data augmentation creates new observations by adjusting existing observations, but only has a minor effect for text classification on consumer reviews (Liesting, Frasincar, & Trusca, 2021).

In addition, the residual classes *'considerans-present'* and *'compliance-present'* have many observations, meaning it heavily influences optimization of the loss function. Model performance could be improved if these classes are weighted less in the loss function, letting the parameters focus less on optimizing for these classes. Also, as the border between these classes is vague, putting them together into one class might also increase model performance as it does not try to learn to distinguish characteristics that are not present.

Furthermore, the regularisation of BERT and RobBERT has been set to its maximum capacity. This was acceptable as the models still overfit the data after around 7 to 8 epochs. Nonetheless, as

mentioned earlier in section 4.6.1.2, applying Automated BERT Regularisation to determine the optimal regularisation may increase the performance (H. D. Lee et al., 2020b).

As mentioned in the conclusion, BERTje and RobBERT are pretrained on much more data, making it an unfair competition for ULMFiT. It could thus be interesting to pretrain ULMFiT on a comparable amount of data and then evaluate the performance on limited data. Furthermore, it could be interesting to train a Bidirectional ULMFiT with masking through auto-encoding as used in BERT, allowing for pretraining a bidirectional language model instead of averaging the predictions of two models. Furthermore, a bidirectional ULMFiT can also be trained purely on the classification task.

A domain-specific legal BERT model could be interesting, as BioBERT has increased model performance for medical texts by 12.24% for question answering (J. Lee et al., 2019). It is however dully noted that English Legal-BERT was trained on 12 GB of Legal texts and only improved slightly on BERT Base on text classification (Chalkidis, Fergadiotis, Malakasiotis, Aletras, & Androutsopoulos, 2020). Nonetheless, the effect on limited data might be interesting for Dutch contracts. Moreover, another advantage of developing a Dutch Legal-BERT model is that it can be applied to any NLP task, as only the classification head must be trained for each specific task. E.g., Named Entity Recognition is interesting to extract party names, contract duration, and amounts of penalty payments.

Another interesting application is multi-label classification, where an observation can belong to multiple classes. This is particularly useful for contracts where a clause can contain multiple statements about different matters, which is often the case for contracts used in the United States. This could more accurately classify the exact description of every clause.

Also, other Transformer architectures could be explored. To date, RoBERT and BERTje are the only Dutch Transformer models available. It could therefore be interesting to train other Transformer models on large amounts of Dutch text data. Moreover, Deep Forests is a new interesting development in the field of NLP that obtained comparable results to Neural Networks while being more interpretable (Zhou & Feng, 2017).

At last, model interpretability is an interesting area to explore. E.g., the importance of certain words in determining a class could be explored, so that a lawyer can inspect the model's decision-making. It is however dully noted that interpretability in NLP shall remain an issue, as humans do not read the text in a particular interpretable way either. Reading texts, among other things, always applies to previous knowledge in different domains, making it difficult to determine what

specifically pinpoints interpretations.

8 Final Remarks

The results have shown that the different state-of-the-art models perform comparably on different amounts of data, where BERTje and its Rotation Estimated version performed the best on smaller amounts of data. Eventually, Model Ensembling enhances prediction performance.

The findings of this research can reduce the time required on reviewing DPAs. Reviewing a DPA by hand takes around an hour, whereas the algorithm is done within a minute. In a well-defined procedure, a lawyer then only has to check clauses that the model was unsure of, as given by its loss value. Also, some sample tests must be performed to ensure the algorithm works right. This research has shown the potential of text classification on contracts, as this field is highly underdigitalized, there is a lot of value to be created.

9 References

Baldi, P., & Sadowski, P. J. (2013). Understanding dropout. *Advances in neural information processing systems*, *26*, 2814–2822.

Bekkerman, R., & Allan, J. (2004). *Using bigrams in text categorization* (Tech. Rep.). Technical Report IR-408, Center of Intelligent Information Retrieval, UMass.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. USA: Oxford University Press, Inc.

Cambria, E., & White, B. (2014). Jumping NLP curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, *9*(2), 48–57.

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2020). LEGAL-BERT: the muppets straight out of law school. *CoRR*, *abs/2010.02559*. Retrieved from `https://arxiv.org/abs/2010.02559`

Dale, R. (2019). Law and word order: NLP in legal tech. *Natural Language Engineering*, *25*(1), 211–217.

Delobelle, P., Winters, T., & Berendt, B. (2020). Robbert: a dutch roberta-based language model. In *Emnlp*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch BERT model. *CoRR*, *abs/1912.09582*. Retrieved from `http://arxiv.org/abs/1912.09582`

Geisser, S. (1993). *Predictive Inference*. Chapman and Hall.

Goldberg, D. (2000, Apr). *What every computer scientist should know about floating-point arithmetic*. Retrieved from `https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html`

Goldberg, Y. (2017). *Neural network methods for natural language processing* (Vol. 37). San Rafael, CA: Morgan & Claypool. doi: doi: 10.2200/S00762ED1V01Y201703HLT037

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved from `https://books.google.co.in/books?id=Np9SDQAAQBAJ`

Grimes, S. (2017). *4 AI Startups that Analyze Customer Reviews.* (retrieved 18 March 2021 from `https://venturebeat.com/2017/01/27/4-ai-startups-that-analyze-customer-reviews/`)

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A field guide to dynamical recurrent neural networks. IEEE Press*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: with applications in r*. Springer.

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization.*

Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview. *arXiv preprint arXiv:2101.09957*.

Lee, H. D., Lee, S., & Kang, U. (2020a). AUBER: automated BERT regularization. *CoRR*, *abs/2009.14409*. Retrieved from `https://arxiv.org/abs/2009.14409`

Lee, H. D., Lee, S., & Kang, U. (2020b). *Auber: Automated bert regularization.*

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2019, Sep). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*. Retrieved from `http://dx.doi.org/10.1093/bioinformatics/btz682` doi: doi: 10.1093/bioinformatics/btz682

Liesting, T., Frasincar, F., & Trusca, M. M. (2021). *Data augmentation in a hybrid approach for aspect-based sentiment analysis.*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). *Roberta: A robustly optimized bert pretraining approach.*

Loshchilov, I., & Hutter, F. (2019). *Decoupled weight decay regularization.*

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, *5*(4), 1093–1113.

Merity, S., Keskar, N. S., & Socher, R. (2017). Regularizing and optimizing LSTM language models. *CoRR*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech*

communication association.

Olah, C. (2015). *Understanding LSTM networks.* GitHub. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Reed, R., & Marks, R. (1999). *Neural smithing: Supervised learning in feedforward artificial neural networks.* MIT Press. Retrieved from https://books.google.nl/books?id=sreDQgAACAAJ

Rich, B. (2020, Oct). *How AI Is Changing Contracts.* Retrieved from https://hbr.org/2018/02/how-ai-is-changing-contracts

Roelants, P. (n.d.). *How to impelement a simple RNN.* GitHub. Retrieved from https://peterroelants.github.io/posts/rnn-implementation-part01/

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., & Stamatopoulos, P. (2001). Stacking classifiers for anti-spam filtering of e-mail. *arXiv preprint cs/0106040*.

Smith, L. N. (2015). No more pesky learning rate guessing games. *CoRR*, *abs/1506.01186*. Retrieved from http://arxiv.org/abs/1506.01186

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929-1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Sussillo, D. (2014). Random walks: Training very deep nonlinear feed-forward networks with smart initialization. *CoRR*, *abs/1412.6558*. Retrieved from http://arxiv.org/abs/1412.6558

Toews, R. (2019, Dec). *AI Will Transform The Field Of Law.* Forbes Magazine. Retrieved from https://www.forbes.com/sites/robtoews/2019/12/19/ai-will-transform-the-field-of-law/?sh=1f78ddff7f01

van der Burgh, B., & Verberne, S. (2019). The merits of universal language model fine-tuning for small datasets - a case with dutch book reviews. *CoRR*, *abs/1910.00896*. Retrieved from http://arxiv.org/abs/1910.00896

Van der Ploeg, T., Austin, P. C., & Steyerberg, E. W. (2014). Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints. *BMC medical research methodology*, *14*(1), 1–13.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.*

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., & Fergus, R. (2013, 17–19 Jun). Regularization of neural networks using dropconnect. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of*

the 30th international conference on machine learning (Vol. 28, pp. 1058–1066). Atlanta, Georgia, USA: PMLR. Retrieved from `http://proceedings.mlr.press/v28/wan13.html`

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.

Zhong, X., & Enke, D. (2019, 12). Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, *5*. doi: doi: 10.1186/s40854-019-0138-0

Zhou, Z.-H., & Feng, J. (2017). Deep forest. *arXiv preprint arXiv:1702.08835*.

Appendix A Data Distribution

Table 3: Data distribution for the DPA's.

	Number of observations	Percentage of total observations
datalek-8uur	2	0,000101286
looptijd-eigen	6	0,000303859
aardsoort-inclausule	9	0,000455789
retour-keuze-vw	9	0,000455789
datalek-12uur	14	0,000709004
wijziging-alleenvw	16	0,000810291
datalek-72uur	16	0,000810291
wijziging-alleenvv	19	0,00096222
looptijd-ontbinden	23	0,00111415
datalek-36uur	26	0,001316722
verzoeken-zelf	30	0,001519295
bijstand-kosten	34	0,001721868
geschillen-rechtbank-recht	46	0,002329586
beveiliging-garantie	50	0,002532158
overdracht-present	54	0,002734731
datalek-48uur	64	0,003241163
aansprakelijkheid-boete	68	0,003443735
wijziging-alswet	83	0,004203383
geschillen-recht	83	0,004203383
looptijd-onbepaald	87	0,004405956
aansprakelijkheid-vrijwaring	87	0,004405956
register-present	91	0,004608528
beveiliging-inspannen	99	0,005013674
aansprakelijkheid-onbeperkt	100	0,005013674
ierechten-present	101	0,00511496
bijstand-present	109	0,005520105
wijziging-samen	113	0,005722678
aansprakelijkheid-beperkt	113	0,005672035
doorgifte-toegestaan	121	0,006127823
audit-zelf	121	0,006127823
aardsoort-generiek	126	0,006381039
datalek-24uur	134	0,006786185
retour-retour	141	0,007140687
instructiestrijd-present	147	0,007444546
subverwerkers-specifiek	156	0,007900334
infoplicht-present	163	0,008254836
retour-present	164	0,00830548
vvgarantie-present	176	0,008913198
geschillen-rechtbank	179	0,009065127
aansprakelijkheid-hoofdovk	182	0,009217057
retour-vernietigen	182	0,009217057
bijstand-dpia	191	0,009672845
retour-keuze-vv	213	0,010786995
verzoeken-doorgeven	233	0,011799858
doorgifte-verboden	234	0,011850501
beveiliging-bijlage	244	0,012356933
datalek-zsm	250	0,012660792
beveiliging-inclausule	255	0,012914008
audit-derde	268	0,013521726
geschillen-present	299	0,015142307
autoriteiten-present	299	0,015142307
looptijd-gelijk-ho	323	0,016357743
aardsoort-bijlage	338	0,017117391
wijziging-present	355	0,017978325
rangorde-present	363	0,01838347
verzoeken-present	404	0,02045984
subverwerkers-algemeen	405	0,020510483
doorgifte-present	417	0,021118201
looptijd-present	424	0,021472703
aansprakelijkheid-present	455	0,023042642
aardsoort-present	480	0,024308721
geheimhouding-present	641	0,032462271
audit-present	881	0,044616631
considerans-present	946	0,047908437
subverwerkers-present	950	0,04811101
datalek-present	1376	0,069684999
beveiliging-present	1746	0,088422972
compliance-present	3209	0,162513927

Appendix B Optimal Learning Rate FastAI

This plot is drawn by taking the bias-corrected exponentially weighted average losses over an interval of the logarithm of learning rates. First, the average loss $\bar{L}$ on index i is calculated:

$$\begin{aligned}\bar{L}_i &= \beta * \bar{L}_{i-1} + (1-\beta) * L_i = \beta^2 * \bar{L}_{i-2} + \beta(1-\beta) * L_{i-1} + \beta L_i = ... \\ &= (1-\beta)\beta^i L_0 + (1-\beta)\beta^{i-1} L_1 + ... + (1-\beta)\beta L_{i-1} + (1-\beta) L_i\end{aligned} \tag{16}$$

Formula 16 shows that the weights are all powers of β, meaning the sum of our weights is equal to:

$$(1-\beta) * \frac{1-\beta^{i+1}}{1-\beta} = 1 - \beta^{i+1} \tag{17}$$

Meaning the plotted average is:

$$smoothed\, \bar{L}_i = \frac{\bar{L}_i}{1-\beta^{i+1}} \tag{18}$$

Finally, the quantity at each step must be calculated, because taking a simple ratio leads to all points being concentrated near the end, as the loss is plotted against the log of the learning rates. The step q is determined as follows:

$$lr_{N-1} = lr_0 \times q^{N-1} \iff q^{N-1} = \frac{lr_{N-1}}{lr_0} \iff q = (\frac{lr_{N-1}}{lr_0})^{\frac{1}{N-1}} \tag{19}$$

Meaning the x-axis value of lr at index i is calculated as follows:

$$\log(lr_i) = \log(lr_0) + i\log(q) = \log(lr_0) + i\frac{\log(lr_{N-1}) - \log(lr_0)}{N-1} \tag{20}$$

Finally, the optimal learning rate can be deduced from the graph. The optimal learning rate is one order of magnitude before the minimum, as that learning rate is on the edge between improving and becoming unsolvable.

Appendix C ULMFiT architecture

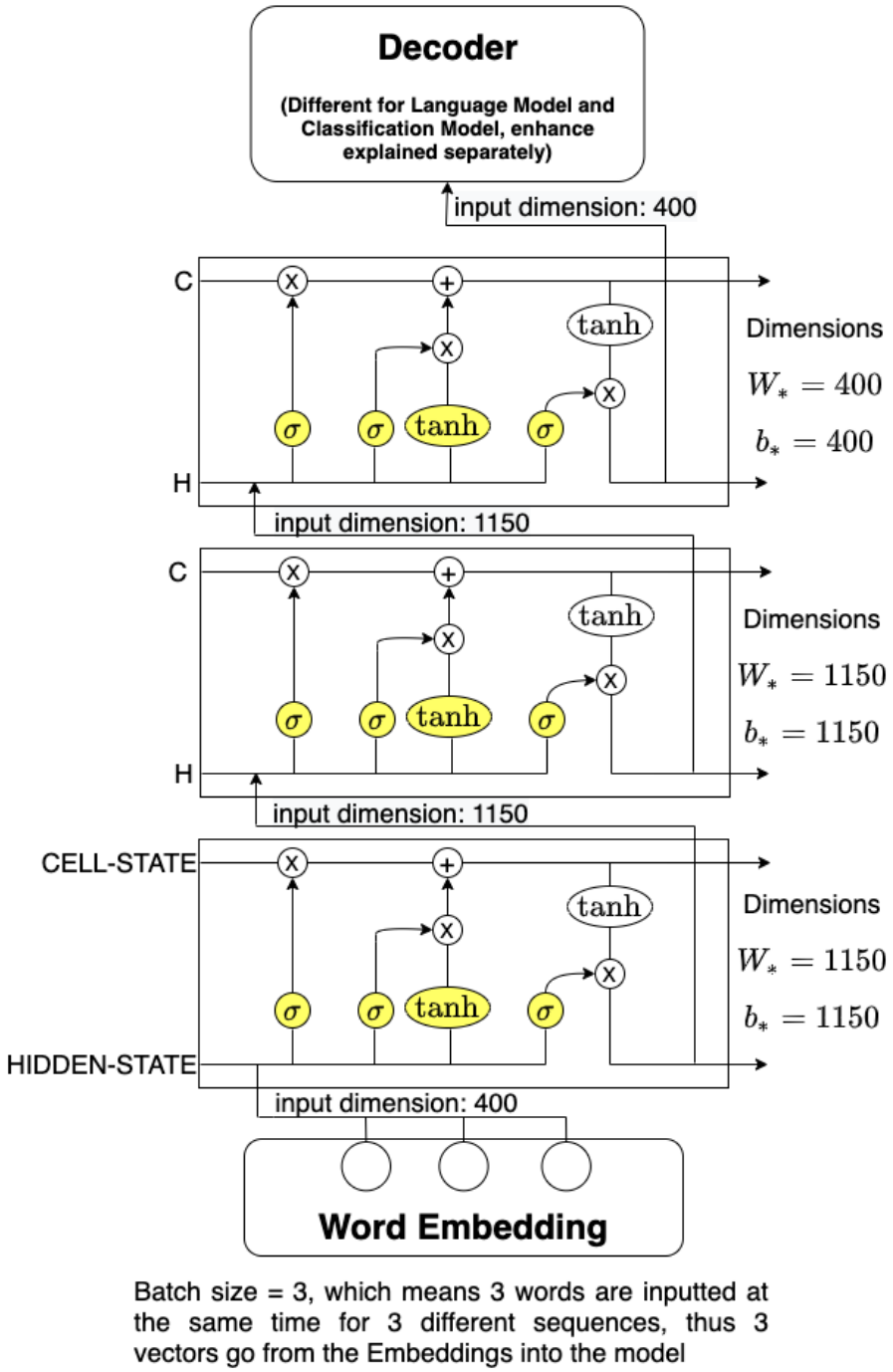


Figure 13: Embeddings and model encoder visualized (the batch size of 3 is merely exemplary)

Appendix D Self-Attention

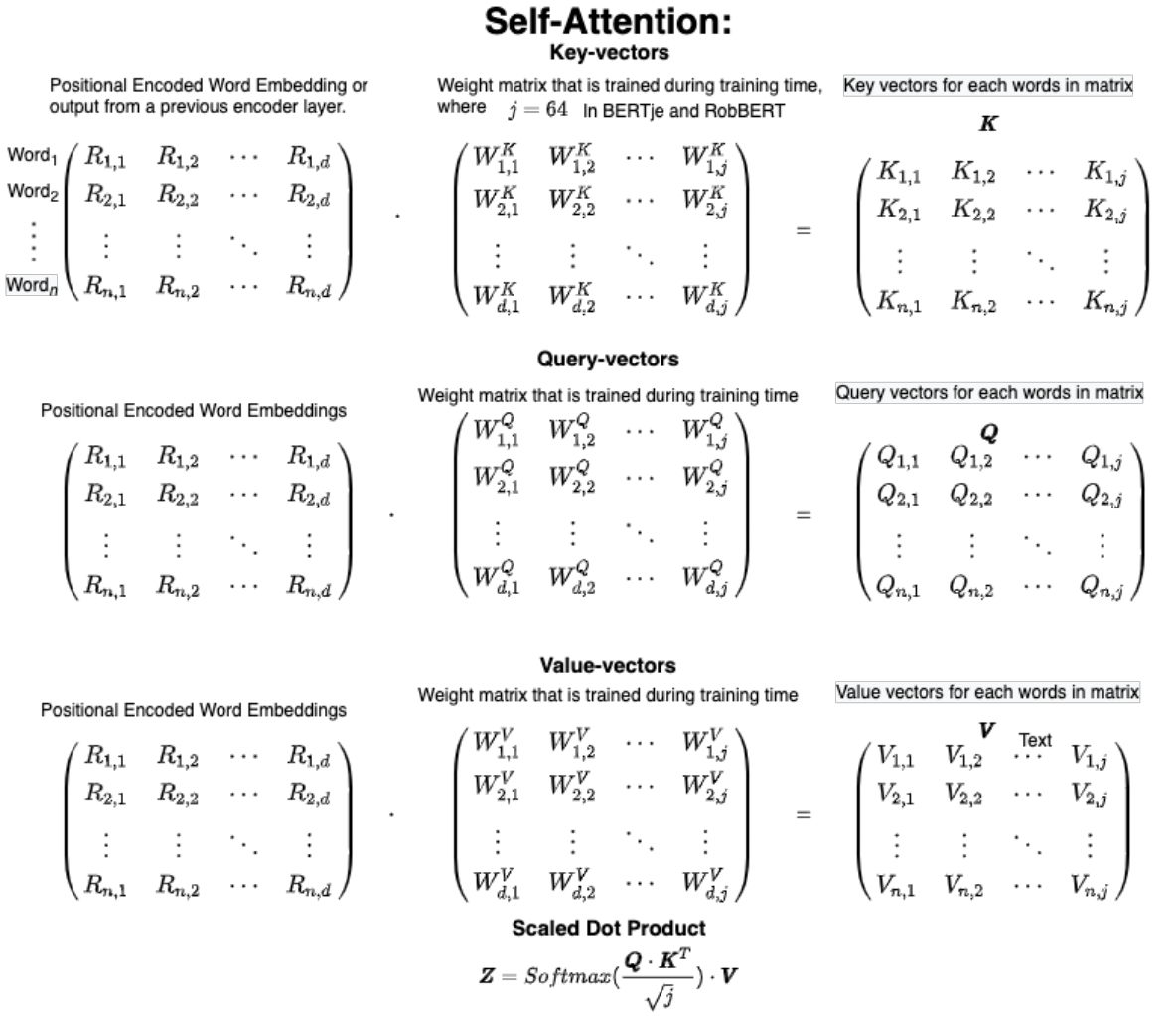


Figure 14: Mathematical representation of Self-Attention

Appendix E Multinomial Logistic Regression

Multinomial Logistic Regression constructs a linear predictor function that gives probability scores for a class dependent on the linearly combined explanatory variables, thus estimating the conditional distribution of the response class given the predictors. For this purpose, it uses the logistic function with maximum likelihood. The logistic function can be estimated as a log-linear function, where the probability that class C is k for observation i is equal to the total of the parameters β_k for features X_i with the logarithm of the partition function Z, coming down to the following:

$$\log P(C_i = k) = \beta_k \cdot X_i - \log(Z) \tag{21}$$

The logarithm of the partition function (Z) is added to ensure that all the probabilities sum to one. Considering that the probabilities sum to 1 and exponentiating both sides turns the additive term into multiplication, that Z can be rewritten as:

$$\begin{aligned} P(C_i = k) = \frac{1}{Z} e^{\beta_k \cdot X_i} \; and \; \sum_{j=1}^{J} P(C_i = j) = 1\,, \\ \sum_{j=1}^{J} \frac{1}{Z} e^{\beta_j \cdot X_i} (=) Z = e^{\beta_j \cdot X_i}\,, \end{aligned} \tag{22}$$

Given formula 21 and formula 22, estimating the probability of class k equals the softmax procedure given in formula 3, where the softmax outputs a vector with probabilities for every class. The Softmax implementation of Multinomial Logistic Regression equals:

$$P(C_i = k) = \frac{e^{\beta_k \cdot X_i}}{\sum_{j=1}^{J} e^{\beta_j \cdot X_i}}\,, \tag{23}$$

As formula 23 is the same as the classification layer in the Neural Networks, the same loss function as stated in formula 4 is optimized. The coefficients are estimated by the gradient-based optimization algorithm Limited-memory Broyden-Fletcher-Foldfarb-Shanno (L-BFGS).

Appendix F Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) estimates the distribution of the predictors X separately for each class C and then uses Bayes' theorem to estimate $P(C = j|X = x)$ (James et al., 2021). LDA thus tries to divide the data space into C separate spaces, which are determined by X. For sake of clarity, Bayes' theorem is stated as the following:

$$P(C = j|x) = \frac{P(x|C = j) \cdot P(C = j)}{P(x)}\,, \tag{24}$$

In LDA $P(x|y)$ is modeled as a multivariate Gaussian distribution with density, being $X \sim N(\mu, \mathbf{\Sigma})$, which is formally defined as (James et al., 2021):

$$P(x|C = j) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_j)\right)\,, \tag{25}$$

where μ_j is a class-specific mean vector and Σ is a covariance matrix that is shared by all classes C. Applying the distribution function to the Bayes theorem results in the following probability estimation for each class j:

$$\delta_j(\mathbf{x}) = \mathbf{x}^T \mathbf{\Sigma}^{-1} \mu_j - \frac{1}{2}\mu_j^T \mathbf{\Sigma}^{-1} \mu_j + \log P(C = j)\,, \tag{26}$$

where $P(C = j)$ is the class prior probability, $\mu_j$the class-wise mean vectors and Σ the covariance matrix. The formula can be solved by applying Singular Value Decomposition (SVD) on the input matrix $\mathbf{X}$ and on the class-wise mean vectors (μ_j), as the covariance matrix is assumed to be constant across classes.

Appendix G Accuracy and macro-weighted F1-score over different data levels

Figure 15: BERTje

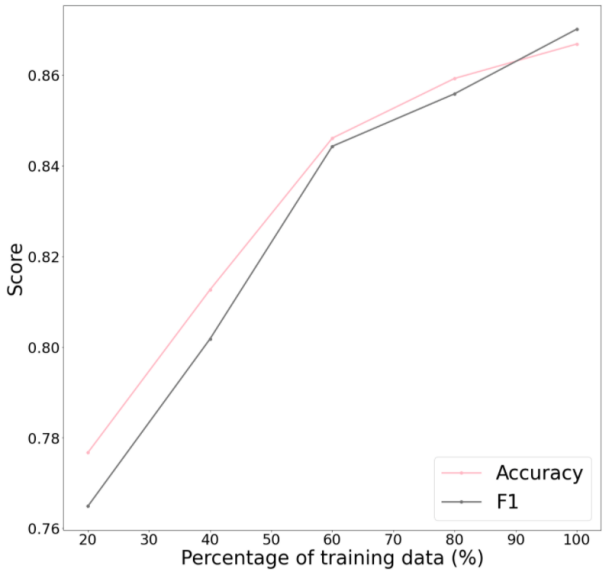


Figure 16: RobBERT

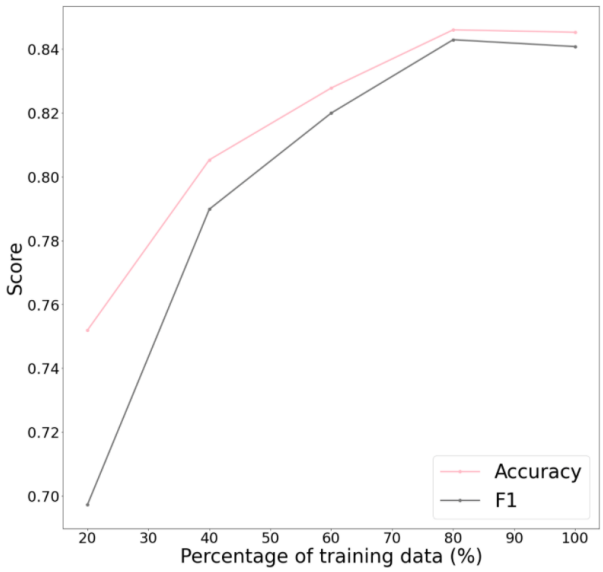


Figure 17: Forward (Left-to-Right) ULMFiT

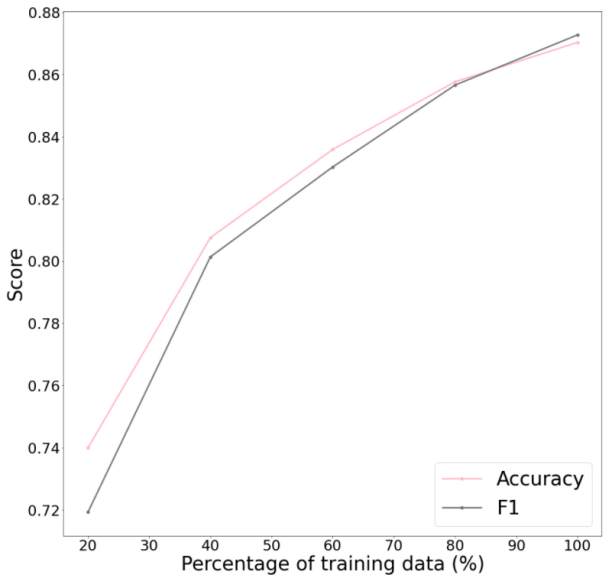


Figure 18: Backward (Right-to-Left) ULMFiT

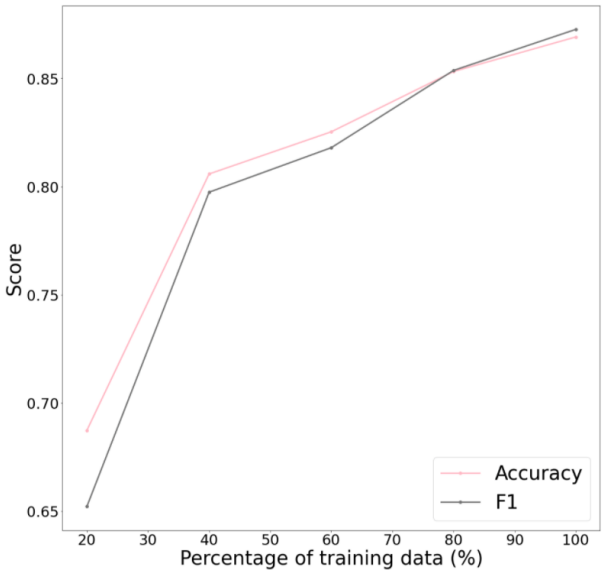


Figure 19: Averaged predictions of Backward and Forward ULMFiT (BI-LSTM)

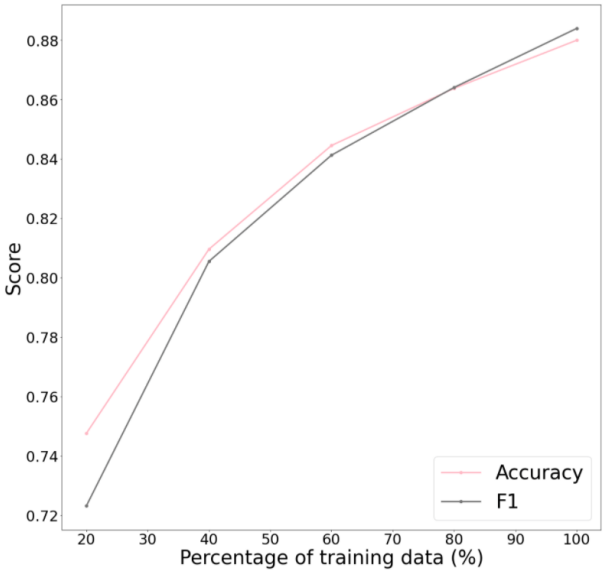


Figure 20: 5-Fold Stratified Rotation Estimated BERTje

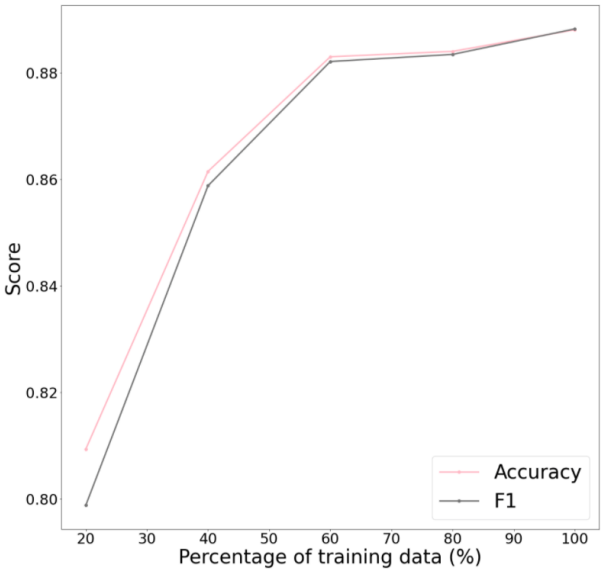


Figure 21: 5-Fold Stratified Rotation Estimated RobBERT

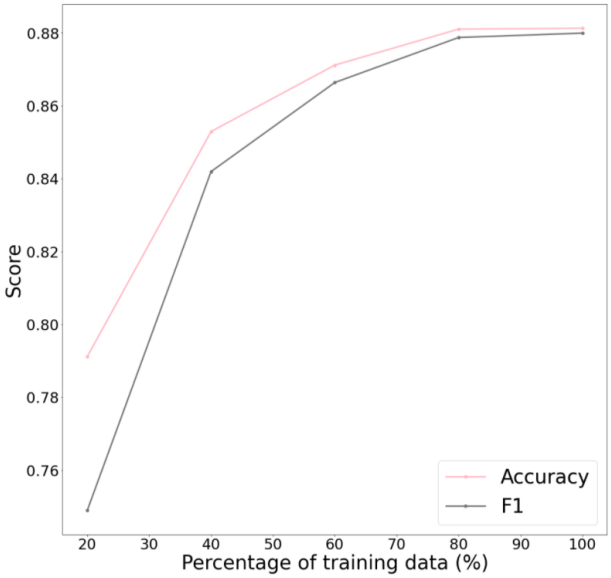


Figure 22: 5-Fold Stratified Rotation Estimated Forward (Left-to-Right) ULMFiT

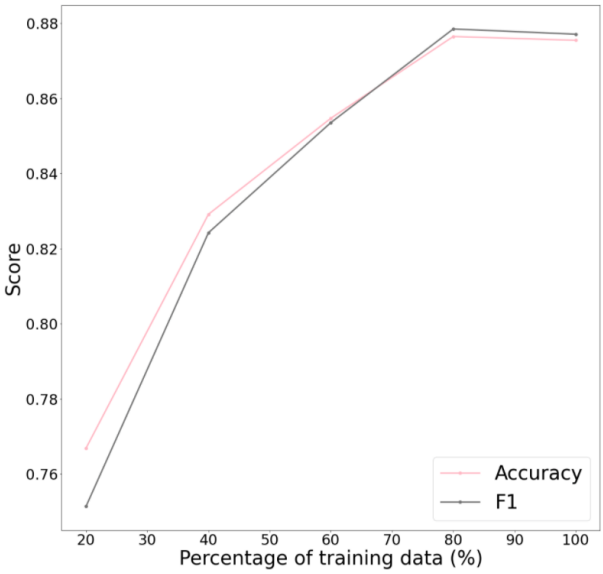


Figure 23: 5-Fold Stratified Rotation Estimated Backward (Right-to-Left) ULMFiT

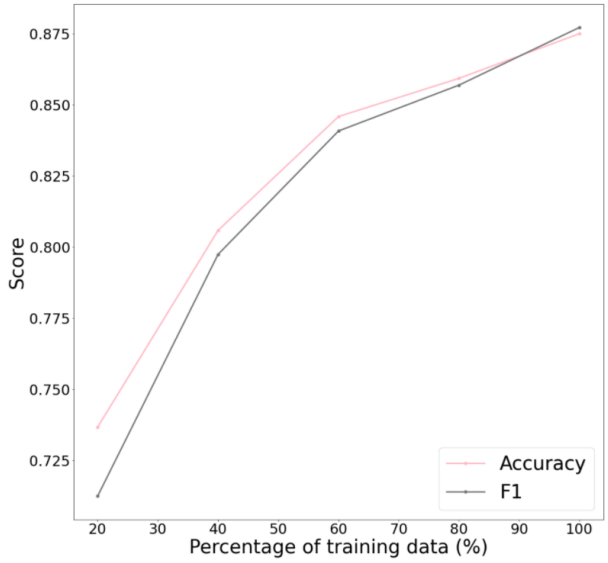


Figure 24: 5-Fold Stratified Rotation Estimated Averaged predictions of Backward and Forward ULMFiT (BI-LSTM)

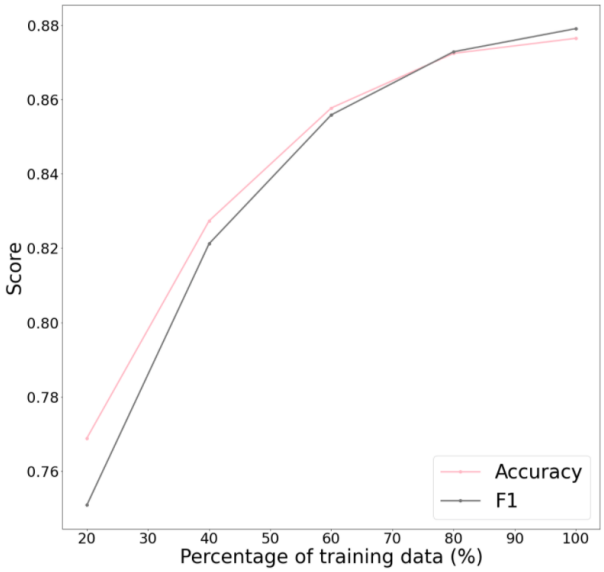


Figure 25: 5-Fold Stratified Rotation Estimated Level 1 Models with Logistic Regression

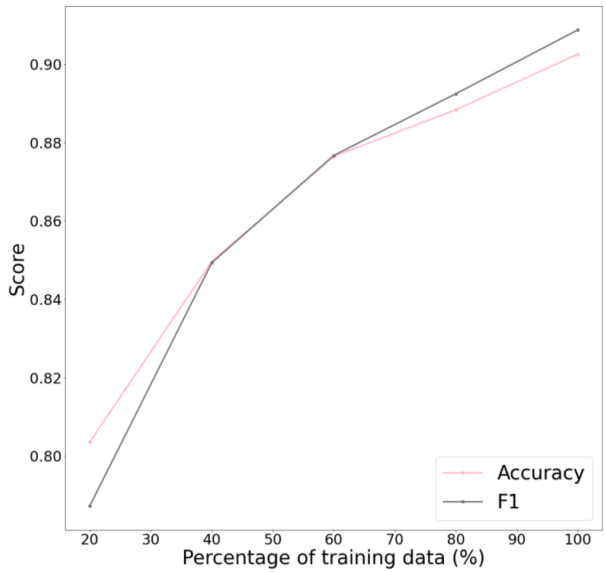
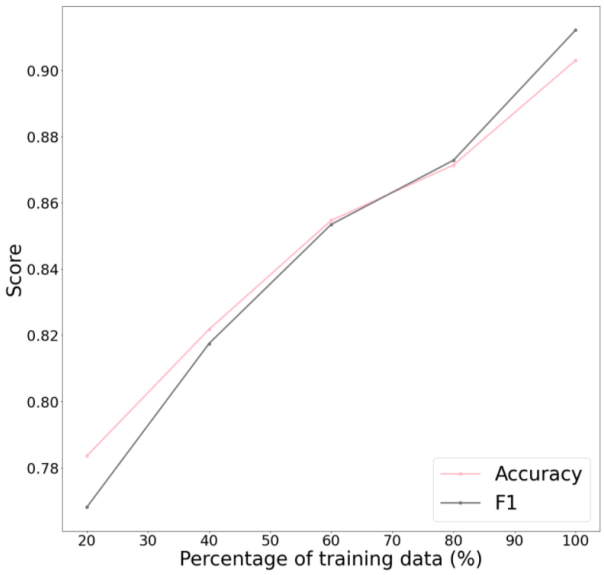


Figure 26: 5-Fold Stratified Rotation Estimated Level 1 Models with Linear Discriminant Analysis



Appendix H Summary Confusion Matrices

Table 4: Summary of the Confusion Matrix of Rotation Estimated BERTje

	Total Predicted	Total Actual	True Positive	False Positive	False Negative	Precision	Recall	F1-score
aansprakelijkheid-present	101	93	87	14	6	0,861386139	0,935483871	0,896907216
aansprakelijkheid-vrijwaring	18	17	16	2	1	0,888888889	0,941176471	0,914285714
aansprakelijkheid-onbeperkt	17	19	14	3	5	0,823529412	0,736842105	0,777777778
aansprakelijkheid-hoofdovk	45	37	32	13	5	0,711111111	0,864864865	0,780487805
aansprakelijkheid-beperkt	7	22	7	0	15	1	0,318181818	0,482758621
aansprakelijkheid-boete	15	14	14	1	0	0,933333333	1	0,965517241
aardsoort-present	94	96	90	4	6	0,957446809	0,9375	0,947368421
aardsoort-bijlage	68	68	66	2	2	0,970588235	0,970588235	0,970588235
aardsoort-generiek	23	25	23	0	2	1	0,92	0,958333333
compliance-present	753	646	624	129	22	0,828685259	0,965944272	0,892065761
aardsoort-inclausule	0	2	0	0	2	0	0	0
doorgifte-present	85	85	74	11	11	0,870588235	0,870588235	0,870588235
audit-present	179	177	167	12	10	0,932960894	0,943502825	0,938202247
audit-zelf	15	24	14	1	10	0,933333333	0,583333333	0,717948718
audit-derde	62	53	52	10	1	0,838709677	0,981132075	0,904347826
beveiliging-present	343	348	324	19	24	0,944606414	0,931034483	0,937771346
infoplicht-present	35	32	29	6	3	0,828571429	0,90625	0,865671642
datalek-zsm	48	50	46	2	4	0,958333333	0,92	0,93877551
autoriteiten-present	61	60	55	6	5	0,901639344	0,916666667	0,909090909
geschillen-rechtbank	36	36	36	0	0	1	1	1
geheimhouding-present	130	128	123	7	5	0,946153846	0,9609375	0,953488372
beveiliging-bijlage	52	49	48	4	1	0,923076923	0,979591837	0,95049505
beveiliging-inclausule	47	51	45	2	6	0,957446809	0,882352941	0,918367347
considerans-present	127	189	114	13	75	0,897637795	0,603174603	0,721518987
beveiliging-inspannen	17	19	17	0	2	1	0,894736842	0,944444444
beveiliging-garantie	12	10	10	2	0	0,833333333	1	0,909090909
subverwerkers-present	189	190	180	9	10	0,952380952	0,947368421	0,949868074
doorgifte-toegestaan	21	24	16	5	8	0,761904762	0,666666667	0,711111111
datalek-present	268	276	265	3	11	0,98880597	0,960144928	0,974264706
bijstand-dpia	39	38	37	2	1	0,948717949	0,973684211	0,961038961
bijstand-present	17	22	15	2	7	0,882352941	0,681818182	0,769230769
bijstand-kosten	2	6	2	0	4	1	0,333333333	0,5
verzoeken-present	86	82	76	10	6	0,88372093	0,926829268	0,904761905
wijziging-present	80	71	60	20	11	0,75	0,845070423	0,794701987
geschillen-present	56	61	48	8	13	0,857142857	0,786885246	0,820512821
datalek-24uur	52	26	25	27	1	0,480769231	0,961538462	0,641025641
datalek-12uur	0	3	0	0	3	0	0	0
datalek-48uur	0	13	0	0	13	0	0	0
datalek-36uur	0	5	0	0	5	0	0	0
datalek-72uur	0	3	0	0	3	0	0	0
datalek-8uur	0	1	0	0	1	0	0	0
doorgifte-verboden	45	45	38	7	7	0,844444444	0,844444444	0,844444444
geheimhouding-present	69	73	62	7	11	0,898550725	0,849315068	0,873239437
geschillen-rechtbank-recht	10	9	9	1	0	0,9	1	0,947368421
geschillen-recht	17	17	17	0	0	1	1	1
wijziging-alswet	20	16	13	7	3	0,65	0,8125	0,722222222
ierechten-present	20	20	19	1	1	0,95	0,95	0,95
instructiestrijd-present	26	29	25	1	4	0,961538462	0,862068966	0,909090909
looptijd-gelijk-ho	73	65	62	11	3	0,849315068	0,953846154	0,898550725
looptijd-present	74	85	69	5	16	0,932432432	0,811764706	0,867924528
looptijd-onbepaald	12	17	12	0	5	1	0,705882353	0,827586207
looptijd-eigen	0	1	0	0	1	0	0	0
looptijd-ontbinden	5	4	4	1	0	0,8	1	0,888888889
overdracht-present	9	11	8	1	3	0,888888889	0,727272727	0,8
register-present	19	18	17	2	1	0,894736842	0,944444444	0,918918919
vvgarantie-present	33	35	32	1	3	0,96969697	0,914285714	0,941176471
retour-present	35	33	31	4	2	0,885714286	0,939393939	0,911764706
retour-vernietigen	38	36	34	4	2	0,894736842	0,944444444	0,918918919
retour-keuze-vv	58	43	37	21	6	0,637931034	0,860465116	0,732673267
retour-retour	9	28	8	1	20	0,888888889	0,285714286	0,432432432
retour-keuze-vw	0	2	0	0	2	0	0	0
subverwerkers-algemeen	72	80	69	3	11	0,958333333	0,8625	0,907894737
subverwerkers-specifiek	37	30	27	10	3	0,72972973	0,9	0,805970149
verzoeken-zelf	0	6	0	0	6	0	0	0
verzoeken-doorgeven	48	46	44	4	2	0,916666667	0,956521739	0,936170213
wijziging-samen	21	23	20	1	3	0,952380952	0,869565217	0,909090909
wijziging-alleenvv	0	4	0	0	4	0	0	0
wijziging-alleenvw	0	3	0	0	3	0	0	0

Table 5: Summary of the Confusion Matrix of Rotation Estimated RobBERT

	Total Predicted	Total Actual	True Positive	False Positive	False Negative	Precision	Recall	F1-score
aansprakelijkheid-present	102	93	85	17	8	0,833333333	0,913978495	0,871794872
aansprakelijkheid-vrijwaring	19	17	16	3	1	0,842105263	0,941176471	0,888888889
aansprakelijkheid-onbeperkt	14	19	13	1	6	0,928571429	0,684210526	0,787878788
aansprakelijkheid-hoofdovk	37	37	26	11	11	0,702702703	0,702702703	0,702702703
aansprakelijkheid-beperkt	18	22	13	5	9	0,722222222	0,590909091	0,65
aansprakelijkheid-boete	14	14	13	1	1	0,928571429	0,928571429	0,928571429
aardsoort-present	99	96	92	7	4	0,929292929	0,958333333	0,943589744
aardsoort-bijlage	69	68	65	4	3	0,942028986	0,955882353	0,948905109
aardsoort-generiek	20	25	20	0	5	1	0,8	0,888888889
compliance-present	747	646	618	129	28	0,827309237	0,956656347	0,887293611
aardsoort-inclausule	0	2	0	0	2	0	0	0
doorgifte-present	102	85	79	23	6	0,774509804	0,929411765	0,844919786
audit-present	174	177	164	10	13	0,942528736	0,926553672	0,934472934
audit-zelf	22	24	19	3	5	0,863636364	0,791666667	0,826086957
audit-derde	57	53	52	5	1	0,912280702	0,981132075	0,945454545
beveiliging-present	359	348	331	28	17	0,922005571	0,951149425	0,936350778
infoplicht-present	36	32	30	6	2	0,833333333	0,9375	0,882352941
datalek-zsm	46	50	45	1	5	0,97826087	0,9	0,9375
autoriteiten-present	60	60	57	3	3	0,95	0,95	0,95
geschillen-rechtbank	40	36	36	4	0	0,9	1	0,947368421
geheimhouding-present	131	128	124	7	4	0,946564885	0,96875	0,957528958
beveiliging-bijlage	51	49	47	4	2	0,921568627	0,959183673	0,94
beveiliging-inclausule	40	51	39	1	12	0,975	0,764705882	0,857142857
considerans-present	134	189	116	18	73	0,865671642	0,613756614	0,718266254
beveiliging-inspannen	17	19	17	0	2	1	0,894736842	0,944444444
beveiliging-garantie	11	10	10	1	0	0,909090909	1	0,952380952
subverwerkers-present	190	190	181	9	9	0,952631579	0,952631579	0,952631579
doorgifte-toegestaan	9	24	9	0	15	1	0,375	0,545454545
datalek-present	271	276	264	7	12	0,974169742	0,956521739	0,965265082
bijstand-dpia	37	38	36	1	2	0,972972973	0,947368421	0,96
bijstand-present	15	22	14	1	8	0,933333333	0,636363636	0,756756757
bijstand-kosten	2	6	2	0	4	1	0,333333333	0,5
verzoeken-present	91	82	78	13	4	0,857142857	0,951219512	0,901734104
wijziging-present	83	71	61	22	10	0,734939759	0,85915493	0,792207792
geschillen-present	54	61	45	9	16	0,833333333	0,737704918	0,782608696
datalek-24uur	50	26	25	25	1	0,5	0,961538462	0,657894737
datalek-12uur	0	3	0	0	3	0	0	0
datalek-48uur	0	13	0	0	13	0	0	0
datalek-36uur	0	5	0	0	5	0	0	0
datalek-72uur	0	3	0	0	3	0	0	0
datalek-8uur	0	1	0	0	1	0	0	0
doorgifte-verboden	43	45	37	6	8	0,860465116	0,822222222	0,840909091
geheimhouding-present	69	73	59	10	14	0,855072464	0,808219178	0,830985915
geschillen-rechtbank-recht	9	9	8	1	1	0,888888889	0,888888889	0,888888889
geschillen-recht	15	17	15	0	2	1	0,882352941	0,9375
wijziging-alswet	17	16	11	6	5	0,647058824	0,6875	0,666666667
ierechten-present	21	20	19	2	1	0,904761905	0,95	0,926829268
instructiestrijd-present	26	29	26	0	3	1	0,896551724	0,945454545
looptijd-gelijk-ho	74	65	62	12	3	0,837837838	0,953846154	0,892086331
looptijd-present	70	85	64	6	21	0,914285714	0,752941176	0,825806452
looptijd-onbepaald	12	17	12	0	5	1	0,705882353	0,827586207
looptijd-eigen	0	1	0	0	1	0	0	0
looptijd-ontbinden	1	4	1	0	3	1	0,25	0,4
overdracht-present	8	11	7	1	4	0,875	0,636363636	0,736842105
register-present	20	18	18	2	0	0,9	1	0,947368421
vvgarantie-present	33	35	32	1	3	0,96969697	0,914285714	0,941176471
retour-present	34	33	31	3	2	0,911764706	0,939393939	0,925373134
retour-vernietigen	40	36	35	5	1	0,875	0,972222222	0,921052632
retour-keuze-vv	60	43	39	21	4	0,65	0,906976744	0,757281553
retour-retour	6	28	6	0	22	1	0,214285714	0,352941176
retour-keuze-vw	0	2	0	0	2	0	0	0
subverwerkers-algemeen	77	80	74	3	6	0,961038961	0,925	0,942675159
subverwerkers-specifiek	30	30	24	6	6	0,8	0,8	0,8
verzoeken-zelf	0	6	0	0	6	0	0	0
verzoeken-doorgeven	46	46	43	3	3	0,934782609	0,934782609	0,934782609
wijziging-samen	16	23	16	0	7	1	0,695652174	0,820512821
wijziging-alleenvv	0	4	0	0	4	0	0	0
wijziging-alleenvw	2	3	0	2	3	0	0	0

Table 6: Summary of the Confusion Matrix of BI-directional ULMFiT (not Rotation Estimated)

	Total Predicted	Total Actual	True Positive	False Positive	False Negative	Precision	Recall	F1-score
aansprakelijkheid-present	93	93	80	13	13	0,860215054	0,860215054	0,860215054
aansprakelijkheid-vrijwaring	17	17	17	0	0	1	1	1
aansprakelijkheid-onbeperkt	25	19	18	7	1	0,72	0,947368421	0,818181818
aansprakelijkheid-hoofdovk	34	37	28	6	9	0,823529412	0,756756757	0,788732394
aansprakelijkheid-beperkt	19	22	16	3	6	0,842105263	0,727272727	0,780487805
aansprakelijkheid-boete	13	14	13	0	1	1	0,928571429	0,962962963
aardsoort-present	98	96	90	8	6	0,918367347	0,9375	0,927835052
aardsoort-bijlage	66	68	63	3	5	0,954545455	0,926470588	0,940298507
aardsoort-generiek	24	25	22	2	3	0,916666667	0,88	0,897959184
compliance-present	746	646	617	129	29	0,827077748	0,955108359	0,886494253
aardsoort-inclausule	0	2	0	0	2	0	0	0
doorgifte-present	90	85	77	13	8	0,855555556	0,905882353	0,88
audit-present	182	177	168	14	9	0,923076923	0,949152542	0,935933148
audit-zelf	22	24	20	2	4	0,909090909	0,833333333	0,869565217
audit-derde	46	53	43	3	10	0,934782609	0,811320755	0,868686869
beveiliging-present	364	348	328	36	20	0,901098901	0,942528736	0,921348315
infoplicht-present	33	32	29	4	3	0,878787879	0,90625	0,892307692
datalek-zsm	46	50	43	3	7	0,934782609	0,86	0,895833333
autoriteiten-present	52	60	50	2	10	0,961538462	0,833333333	0,892857143
geschillen-rechtbank	33	36	33	0	3	1	0,916666667	0,956521739
geheimhouding-present	129	128	122	7	6	0,945736434	0,953125	0,949416342
beveiliging-bijlage	51	49	45	6	4	0,882352941	0,918367347	0,9
beveiliging-inclausule	35	51	33	2	18	0,942857143	0,647058824	0,76744186
considerans-present	151	189	124	27	65	0,821192053	0,656084656	0,729411765
beveiliging-inspannen	17	19	16	1	3	0,941176471	0,842105263	0,888888889
beveiliging-garantie	9	10	7	2	3	0,777777778	0,7	0,736842105
subverwerkers-present	188	190	179	9	11	0,95212766	0,942105263	0,947089947
doorgifte-toegestaan	22	24	17	5	7	0,772727273	0,708333333	0,739130435
datalek-present	264	276	258	6	18	0,977272727	0,934782609	0,955555556
bijstand-dpia	36	38	35	1	3	0,972222222	0,921052632	0,945945946
bijstand-present	18	22	14	4	8	0,777777778	0,636363636	0,7
bijstand-kosten	4	6	3	1	3	0,75	0,5	0,6
verzoeken-present	87	82	75	12	7	0,862068966	0,914634146	0,887573964
wijziging-present	75	71	60	15	11	0,8	0,845070423	0,821917808
geschillen-present	58	61	47	11	14	0,810344828	0,770491803	0,789915966
datalek-24uur	31	26	26	5	0	0,838709677	1	0,912280702
datalek-12uur	0	3	0	0	3	0	0	0
datalek-48uur	17	13	12	5	1	0,705882353	0,923076923	0,8
datalek-36uur	4	5	3	1	2	0,75	0,6	0,666666667
datalek-72uur	1	3	0	1	3	0	0	0
datalek-8uur	0	1	0	0	1	0	0	0
doorgifte-verboden	39	45	37	2	8	0,948717949	0,822222222	0,880952381
geheimhouding-present	75	73	60	15	13	0,8	0,821917808	0,810810811
geschillen-rechtbank-recht	10	9	8	2	1	0,8	0,888888889	0,842105263
geschillen-recht	17	17	16	1	1	0,941176471	0,941176471	0,941176471
wijziging-alswet	16	16	12	4	4	0,75	0,75	0,75
ierechten-present	17	20	17	0	3	1	0,85	0,918918919
instructiestrijd-present	29	29	26	3	3	0,896551724	0,896551724	0,896551724
looptijd-gelijk-ho	60	65	55	5	10	0,916666667	0,846153846	0,88
looptijd-present	82	85	74	8	11	0,902439024	0,870588235	0,886227545
looptijd-onbepaald	22	17	17	5	0	0,772727273	1	0,871794872
looptijd-eigen	1	1	1	0	0	1	1	1
looptijd-ontbinden	2	4	2	0	2	1	0,5	0,666666667
overdracht-present	10	11	9	1	2	0,9	0,818181818	0,857142857
register-present	15	18	15	0	3	1	0,833333333	0,909090909
vvgarantie-present	37	35	34	3	1	0,918918919	0,971428571	0,944444444
retour-present	25	33	20	5	13	0,8	0,606060606	0,689655172
retour-vernietigen	41	36	30	11	6	0,731707317	0,833333333	0,779220779
retour-keuze-vv	52	43	38	14	5	0,730769231	0,88372093	0,8
retour-retour	22	28	13	9	15	0,590909091	0,464285714	0,52
retour-keuze-vw	0	2	0	0	2	0	0	0
subverwerkers-algemeen	66	80	65	1	15	0,984848485	0,8125	0,890410959
subverwerkers-specifiek	39	30	30	9	0	0,769230769	1	0,869565217
verzoeken-zelf	2	6	2	0	4	1	0,333333333	0,5
verzoeken-doorgeven	48	46	43	5	3	0,895833333	0,934782609	0,914893617
wijziging-samen	20	23	18	2	5	0,9	0,782608696	0,837209302
wijziging-alleenvv	2	4	2	0	2	1	0,5	0,666666667
wijziging-alleenvw	1	3	1	0	2	1	0,333333333	0,5

Table 7: Summary of the Confusion Matrix of Model Ensembling with Logistic Regression

	Total Predicted	Total Actual	True Positive	False Positive	False Negative	Precision	Recall	F1-score
aansprakelijkheid-present	97	93	84	13	9	0,865979381	0,903225806	0,884210526
aansprakelijkheid-vrijwaring	16	17	15	1	2	0,9375	0,882352941	0,909090909
aansprakelijkheid-onbeperkt	20	19	17	3	2	0,85	0,894736842	0,871794872
aansprakelijkheid-hoofdovk	33	37	28	5	9	0,848484848	0,756756757	0,8
aansprakelijkheid-beperkt	22	22	16	6	6	0,727272727	0,727272727	0,727272727
aansprakelijkheid-boete	13	14	13	0	1	1	0,928571429	0,962962963
aardsoort-present	97	96	92	5	4	0,948453608	0,958333333	0,953367876
aardsoort-bijlage	66	68	65	1	3	0,984848485	0,955882353	0,970149254
aardsoort-generiek	23	25	22	1	3	0,956521739	0,88	0,916666667
compliance-present	754	646	627	127	19	0,831564987	0,970588235	0,895714286
aardsoort-inclausule	0	2	0	0	2	0	0	0
doorgifte-present	88	85	77	11	8	0,875	0,905882353	0,89017341
audit-present	173	177	166	7	11	0,959537572	0,937853107	0,948571429
audit-zelf	21	24	20	1	4	0,952380952	0,833333333	0,888888889
audit-derde	56	53	52	4	1	0,928571429	0,981132075	0,95412844
beveiliging-present	351	348	328	23	20	0,934472934	0,942528736	0,938483548
infoplicht-present	34	32	30	4	2	0,882352941	0,9375	0,909090909
datalek-zsm	48	50	47	1	3	0,979166667	0,94	0,959183673
autoriteiten-present	58	60	57	1	3	0,982758621	0,95	0,966101695
geschillen-rechtbank	36	36	35	1	1	0,972222222	0,972222222	0,972222222
geheimhouding-present	129	128	124	5	4	0,96124031	0,96875	0,964980545
beveiliging-bijlage	50	49	48	2	1	0,96	0,979591837	0,96969697
beveiliging-inclausule	46	51	43	3	8	0,934782609	0,843137255	0,886597938
considerans-present	139	189	124	15	65	0,892086331	0,656084656	0,756097561
beveiliging-inspannen	16	19	16	0	3	1	0,842105263	0,914285714
beveiliging-garantie	9	10	8	1	2	0,888888889	0,8	0,842105263
subverwerkers-present	189	190	182	7	8	0,962962963	0,957894737	0,960422164
doorgifte-toegestaan	20	24	18	2	6	0,9	0,75	0,818181818
datalek-present	269	276	266	3	10	0,988847584	0,963768116	0,976146789
bijstand-dpia	37	38	37	0	1	1	0,973684211	0,986666667
bijstand-present	21	22	16	5	6	0,761904762	0,727272727	0,744186047
bijstand-kosten	2	6	2	0	4	1	0,333333333	0,5
verzoeken-present	89	82	76	13	6	0,853932584	0,926829268	0,888888889
wijziging-present	80	71	62	18	9	0,775	0,873239437	0,821192053
geschillen-present	57	61	49	8	12	0,859649123	0,803278689	0,830508475
datalek-24uur	33	26	25	8	1	0,757575758	0,961538462	0,847457627
datalek-12uur	0	3	0	0	3	0	0	0
datalek-48uur	15	13	11	4	2	0,733333333	0,846153846	0,785714286
datalek-36uur	3	5	3	0	2	1	0,6	0,75
datalek-72uur	0	3	0	0	3	0	0	0
datalek-8uur	0	1	0	0	1	0	0	0
doorgifte-verboden	43	45	38	5	7	0,88372093	0,844444444	0,863636364
geheimhouding-present	73	73	61	12	12	0,835616438	0,835616438	0,835616438
geschillen-rechtbank-recht	11	9	9	2	0	0,818181818	1	0,9
geschillen-recht	16	17	16	0	1	1	0,941176471	0,96969697
wijziging-alswet	16	16	12	4	4	0,75	0,75	0,75
ierechten-present	18	20	18	0	2	1	0,9	0,947368421
instructiestrijd-present	27	29	26	1	3	0,962962963	0,896551724	0,928571429
looptijd-gelijk-ho	63	65	56	7	9	0,888888889	0,861538462	0,875
looptijd-present	80	85	71	9	14	0,8875	0,835294118	0,860606061
looptijd-onbepaald	18	17	15	3	2	0,833333333	0,882352941	0,857142857
looptijd-eigen	0	1	0	0	1	0	0	0
looptijd-ontbinden	3	4	3	0	1	1	0,75	0,857142857
overdracht-present	11	11	9	2	2	0,818181818	0,818181818	0,818181818
register-present	17	18	17	0	1	1	0,944444444	0,971428571
vvgarantie-present	35	35	34	1	1	0,971428571	0,971428571	0,971428571
retour-present	32	33	31	1	2	0,96875	0,939393939	0,953846154
retour-vernietigen	37	36	34	3	2	0,918918919	0,944444444	0,931506849
retour-keuze-vv	43	43	36	7	7	0,837209302	0,837209302	0,837209302
retour-retour	27	28	20	7	8	0,740740741	0,714285714	0,727272727
retour-keuze-vw	0	2	0	0	2	0	0	0
subverwerkers-algemeen	72	80	70	2	10	0,972222222	0,875	0,921052632
subverwerkers-specifiek	32	30	26	6	4	0,8125	0,866666667	0,838709677
verzoeken-zelf	0	6	0	0	6	0	0	0
verzoeken-doorgeven	46	46	43	3	3	0,934782609	0,934782609	0,934782609
wijziging-samen	17	23	16	1	7	0,941176471	0,695652174	0,8
wijziging-alleenvv	2	4	2	0	2	1	0,5	0,666666667
wijziging-alleenvw	1	3	1	0	2	1	0,333333333	0,5

Table 8: Summary of the Confusion Matrix of Model Ensembling with LDA

	Total Predicted	Total Actual	True Positive	False Positive	False Negative	Precision	Recall	F1-score
aansprakelijkheid-present	92	93	82	10	11	0,891304348	0,88172043	0,886486486
aansprakelijkheid-vrijwaring	17	17	16	1	1	0,941176471	0,941176471	0,941176471
aansprakelijkheid-onbeperkt	24	19	19	5	0	0,791666667	1	0,88372093
aansprakelijkheid-hoofdovk	35	37	28	7	9	0,8	0,756756757	0,777777778
aansprakelijkheid-beperkt	20	22	16	4	6	0,8	0,727272727	0,761904762
aansprakelijkheid-boete	13	14	13	0	1	1	0,928571429	0,962962963
aardsoort-present	96	96	92	4	4	0,958333333	0,958333333	0,958333333
aardsoort-bijlage	67	68	65	2	3	0,970149254	0,955882353	0,962962963
aardsoort-generiek	23	25	22	1	3	0,956521739	0,88	0,916666667
compliance-present	753	646	625	128	21	0,83001328	0,96749226	0,893495354
aardsoort-inclausule	0	2	0	0	2	0	0	0
doorgifte-present	89	85	76	13	9	0,853932584	0,894117647	0,873563218
audit-present	174	177	167	7	10	0,959770115	0,943502825	0,951566952
audit-zelf	21	24	20	1	4	0,952380952	0,833333333	0,888888889
audit-derde	56	53	52	4	1	0,928571429	0,981132075	0,95412844
beveiliging-present	344	348	325	19	23	0,944767442	0,933908046	0,939306358
infoplicht-present	34	32	31	3	1	0,911764706	0,96875	0,939393939
datalek-zsm	48	50	46	2	4	0,958333333	0,92	0,93877551
autoriteiten-present	59	60	56	3	4	0,949152542	0,933333333	0,941176471
geschillen-rechtbank	36	36	35	1	1	0,972222222	0,972222222	0,972222222
geheimhouding-present	126	128	123	3	5	0,976190476	0,9609375	0,968503937
beveiliging-bijlage	49	49	47	2	2	0,959183673	0,959183673	0,959183673
beveiliging-inclausule	48	51	45	3	6	0,9375	0,882352941	0,909090909
considerans-present	143	189	126	17	63	0,881118881	0,666666667	0,759036145
beveiliging-inspannen	17	19	16	1	3	0,941176471	0,842105263	0,888888889
beveiliging-garantie	11	10	10	1	0	0,909090909	1	0,952380952
subverwerkers-present	185	190	179	6	11	0,967567568	0,942105263	0,954666667
doorgifte-toegestaan	19	24	17	2	7	0,894736842	0,708333333	0,790697674
datalek-present	269	276	265	4	11	0,985130112	0,960144928	0,972477064
bijstand-dpia	37	38	37	0	1	1	0,973684211	0,986666667
bijstand-present	22	22	18	4	4	0,818181818	0,818181818	0,818181818
bijstand-kosten	2	6	2	0	4	1	0,333333333	0,5
verzoeken-present	87	82	77	10	5	0,885057471	0,93902439	0,911242604
wijziging-present	78	71	61	17	10	0,782051282	0,85915493	0,818791946
geschillen-present	66	61	50	16	11	0,757575758	0,819672131	0,787401575
datalek-24uur	29	26	26	3	0	0,896551724	1	0,945454545
datalek-12uur	2	3	0	2	3	0	0	0
datalek-48uur	12	13	11	1	2	0,916666667	0,846153846	0,88
datalek-36uur	6	5	4	2	1	0,666666667	0,8	0,727272727
datalek-72uur	3	3	2	1	1	0,666666667	0,666666667	0,666666667
datalek-8uur	1	1	1	0	0	1	1	1
doorgifte-verboden	45	45	39	6	6	0,866666667	0,866666667	0,866666667
geheimhouding-present	67	73	58	9	15	0,865671642	0,794520548	0,828571429
geschillen-rechtbank-recht	9	9	8	1	1	0,888888889	0,888888889	0,888888889
geschillen-recht	16	17	16	0	1	1	0,941176471	0,96969697
wijziging-alswet	16	16	12	4	4	0,75	0,75	0,75
ierechten-present	18	20	18	0	2	1	0,9	0,947368421
instructiestrijd-present	27	29	26	1	3	0,962962963	0,896551724	0,928571429
looptijd-gelijk-ho	65	65	57	8	8	0,876923077	0,876923077	0,876923077
looptijd-present	71	85	68	3	17	0,957746479	0,8	0,871794872
looptijd-onbepaald	21	17	17	4	0	0,80952381	1	0,894736842
looptijd-eigen	4	1	1	3	0	0,25	1	0,4
looptijd-ontbinden	4	4	4	0	0	1	1	1
overdracht-present	10	11	9	1	2	0,9	0,818181818	0,857142857
register-present	18	18	17	1	1	0,944444444	0,944444444	0,944444444
vvgarantie-present	34	35	33	1	2	0,970588235	0,942857143	0,956521739
retour-present	33	33	31	2	2	0,939393939	0,939393939	0,939393939
retour-vernietigen	38	36	35	3	1	0,921052632	0,972222222	0,945945946
retour-keuze-vv	40	43	35	5	8	0,875	0,813953488	0,843373494
retour-retour	25	28	18	7	10	0,72	0,642857143	0,679245283
retour-keuze-vw	3	2	0	3	2	0	0	0
subverwerkers-algemeen	71	80	69	2	11	0,971830986	0,8625	0,913907285
subverwerkers-specifiek	34	30	27	7	3	0,794117647	0,9	0,84375
verzoeken-zelf	2	6	2	0	4	1	0,333333333	0,5
verzoeken-doorgeven	45	46	43	2	3	0,955555556	0,934782609	0,945054945
wijziging-samen	17	23	16	1	7	0,941176471	0,695652174	0,8
wijziging-alleenvv	3	4	2	1	2	0,666666667	0,5	0,571428571
wijziging-alleenvw	1	3	1	0	2	1	0,333333333	0,5