Using Natural Language Processing with Deep Learning to Explore Clinical Notes

Anders Benjamin Grinde
Bendik Mathias Johansen

Master's thesis in Software Engineering at

Department of Computing, Mathematics and Physics,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

June 1, 2021

Acknowledgements

First of all we would like to thank Dr. Alexander Selvikvåg Lundervold for being our supervisor during this project and for introducing us to machine learning. We especially would like to thank him for guidance and support during the whole project, and for always being an inspiring and enthusiastic person in the field.

Thank you to Mohn Medical Imaging and Visualization Centre for always letting us come and work with superior equipment, when available.

Finally we would like to thank our co-students and everyone at MMIV for all the discussions and guidance we have had during our project.

Abstract

In recent years, the deep learning community and technology have grown substantially, both in terms of research and applications. However, some application areas have lagged behind. The medical domain is an example of a field with a lot of untapped potential, partly caused by complex issues related to privacy and ethics. Still, deep learning is a very powerful tool to utilize structured and unstructured data, and could help save lives. In this thesis, we use natural language processing to interpret clinical notes and predict the mortality rate of subjects. We explore if language models trained on a specific domain would become more performant, and we compared them to language models trained on an intermediate data set. We found that our language model trained on an intermediate data set that had some resemblance to our target data set performed slightly better than its counterpart language model. We found that text classifiers built on top of the language models were capable of correctly predicting if a subject would die or not. Furthermore, we extracted the free-text features from the text classifiers and combined them, using stacking, with heterogeneous data as an attempt to increase the efficacy of the classifiers and to explore the relative performance boost gained by including free-text features. We found a correlation between the quality of text classifiers that produced the text features and the stacking classifiers' performances. The classifier that was trained on a data set without text features performed the worst, and the classifier trained on a data set with the best text features performed the best. We also discuss the central concerns that come with applying deep learning in a medical domain with regards to privacy and ethics. It is our intention that this thesis serves as a contribution to the advancement of deep learning within the medical domain, and as a testament as to what can be achieved with today's technology.

Table of contents

I Introduction **1**

II Theoretical Background **4**

1 Natural language processing and deep learning **5**

1.1 Vocabulary, tokenization and numericalization 5
1.2 Transfer learning 6
1.3 Artificial neural networks 7
1.3.1 Layers 7
1.3.2 Weights of activations 8
1.3.3 Recurrent Neural Networks 8
1.3.4 ASGD Weight-Dropped LSTM 9
1.3.5 Language models 10
1.3.6 Classification models 10
1.4 Fastai: a deep learning library 10
1.5 The architecture of ULMFiT 11
1.5.1 General-domain pre-trained language models 11
1.5.2 Target task LM fine-tuning 12
1.5.3 Target task classifier fine-tuning 12

2 Evaluating and understanding model predictions **13**

2.1 Language model metrics 13
2.1.1 Accuracy 13
2.1.2 Perplexity 13
2.1.3 Loss 14
2.2 Binary classification model metrics 14
2.2.1 Recall and precision 15
2.2.2 F_1-score 15
2.2.3 Accuracy 15
2.2.4 Matthews correlation coefficient 15
2.3 Confusion Matrix 16
2.4 Feature Importance 16

3 Privacy and ethics in machine learning **17**

3.1 Privacy 17
3.2 Ethics 17

III Research questions **20**

4 Research questions and hypotheses for the experimental work **21**

4.1 Examples of related work 21
4.2 Can natural language processing with deep learning pick up on linguistic features in clinical notes? 22
4.3 Can a text classifier built from the language model extract useful information from clinical text? 23
4.4 Can the extracted information be used in combination with other data? 23
4.5 Can the deep learning models be explored and be made sense of? 24

IV Experimental work **25**

5 A language model trained on free-text clinical notes **26**

5.1 The MIMIC-III data set 26
5.2 Training a language model 26
5.2.1 Vocabulary and tokenization 27
5.2.2 Metrics 27
5.3 Structure data and gather the clinical notes 27
5.3.1 MIMIC extract 27
5.4 Experiments 28
5.4.1 Training a language model from Wikipedia 28
5.4.2 Training a langauge model from PubMed Medline 33
5.5 Conclusion 35

6 Making predictions on clinical free-text **36**

6.1 Training a Text Classifier 36
6.1.1 Metrics 36
6.1.2 Cross validation 36
6.2 Aggregating clinical notes 36
6.3 Experiment 37
6.3.1 In-hospital mortality 37
6.3.2 Results 38
6.4 Conclusion 41

7 Stacking free-text features with heterogeneous data **43**

7.1 Extracting activations from the text classifier 43
7.2 Aggregating heterogeneous clinical data 43
7.3 Combining heterogeneous clinical data and free-text features . . 44
7.4 Finding a suitable classifier 44
7.5 Experiments 44
7.5.1 In-hospital mortality without free-text features 45
7.5.2 In-hospital mortality with free-text features 46
7.6 Conclusion 50

V Conclusive work **51**

8 Discussion **52**

8.1 Language models on free-text 52
8.1.1 Sources of error 53
8.2 Text classifier on clinical notes 53
8.3 Stacking free-text features with heterogeneous data 56

9 Conclusion **58**

9.1 Can natural language processing with deep learning pick up on linguistic features in clinical notes? 58
9.2 Can a text classifier built from the language model extract useful information from clinical text 58
9.3 Can the extracted information be used in combination with other data? 58
9.4 Can the deep learning models be explored and be made sense of? 59

10 Further work **60**

VI Literature and References **62**

References **63**

List of Figures

1.1 A taxonomy for transfer learning (Ruder, 2019) 6
1.2 A simple neural network. Neurons in each layer are depicted as circles and arrows are signaling the connections between them. . 8
1.3 A recurrent neural network neuron with output given back to itself. 9
1.4 The 3 stages of the ULMFiT technique on a neural network using transfer learning to fine tune a text based model. From ULMFiT paper (Howard and Ruder, 2018) 11

2.1 Predicted values by actual values in a two-by-two contingency table. FN is short for false negative, TP is short for true positive, TN is short for true negative and FP is short for false positive. . 14

5.5 Transfer learning through Medline to further improve the language model towards the medical field. 33

6.1 Mortality rate class distribution 38
6.2 Confusion matrices for the text classifiers without a random seed after evaluation of the test data set. The test data set accounts for 20% of the whole data set, and is split using stratified sampling to ensure the same proportion of classes. 40
6.3 Confusion matrices for the text classifiers without a random seed after evaluation of the test data set. The test data set accounts for 20% of the whole data set, and is split using stratified sampling to ensure the same proportion of classes. 40

7.1 The most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set does not contain any extracted text features. 46
7.2 The most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set contains text features extracted from the best performing text classifier. 47
7.3 A graphical representation of the most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set contains text features extracted from the worst performing text classifier. 49

7.4 The confusion matrices for the best classifiers evaluated on the test data sets. 50

List of Tables

5.1 Results from training on MIMIC with a language model with a size 30,000 for vocabulary. 29
5.2 Results from training on MIMIC with a language model with 60,000 vocabulary. Until the model converged towards a number. 29
5.3 Result from predictions. Text given to the models are marked in bold and with the color red. The models predicted 100 succeeding tokens. Newlines are depicted as "\n". 32
5.4 Best results from the medline and 60,00 mimic models with same train and validation sets. 33
5.5 Medline predictions on the same text as in Table 5.3, predicting 100 succeeding tokens. The results shows an understanding of the structure of the MIMIC clinical notes, with slightly less predictions of special tokens. 34

6.1 Text classifier using language model with a vocabulary of 60,000 words without a fixed random state seed. This is the first text classifier in the list of text classifier versions in section 6.3.1. . . . 39
6.2 Text classifier using language model with a vocabulary of 30,000 words without a random state seed. This is the second text classifier in the list of text classifier versions in section 6.3.1. 39
6.3 Text classifier using language model trained on Medline with a vocabulary of 60,000 words and a seed for random state 41
6.4 Text classifier using language model with a vocabulary of 60,000 words and a seed for random state 41
6.5 Results of the text classifiers evaluation on the test set. The first text classifier refers to the one based on a language model with a vocabulary size of 60,000 trained through 15 epochs. The second refers to the text classifier is based on a language model with a vocabulary size of 30,000 trained though 21 epochs. The third and fourth text classifiers are based on language models with a vocabulary size of 60,000 trained through 17 and 10 epochs respectively. Additionally, the third text classifier is trained on an intermediate data set, Medline, before being trained on MIMIC-III. There seems to be something wrong with the fourth text classifier because it was trained very similar to the other text classifiers, but performed much worse 42

7.1 The top five models' mean metric scores without free-text features for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. ET is short for Extra Trees Classifier, and RF is the Random Forest Classifier. 45
7.2 The top three tuned models' mean metric scores without free-text features for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. 45
7.3 The top five model mean metric scores with free-text features for ten stratified sample folds. The free-text features are extracted from a text classifier built on a language model with a vocabulary size of 60,000 and without a seed, which performed the best out of the trained text classifiers. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. ET is short for Extra Trees Classifier, and RF is the Random Forest Classifier. 46
7.4 The top three tuned models' mean metric scores with the free-text features extracted from the best performing text classifier for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. 47
7.5 The top five model mean metric scores with free-text features for ten stratified sample folds. The free-text features were extracted from a text classifier built on a language model with a vocabulary size of 30,000, which performed the best worst out of the trained text classifiers. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. DT is short for Decision Tree Classifier, and RF is the Random Forest Classifier. 48
7.6 The top three tuned models' mean metric scores with the free-text features extracted from the worst performing text classifier for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. 48
7.7 The results for each of the best classifiers evaluated on the test set, all of which are gradient boosting classifiers. The first classifier is trained on a data set with text features extracted from the best performing text classifier. The second classifier is trained on a data set with text features extracted from the worst performing text classifier. The third classifier is trained on a data set without text features. 49

Abbreviations

AI - **A**rtificial **I**ntelligence
Ada - **Ada** Boost Classifier
ANN - **A**rtificial **N**eural **N**etworks
ASGD - **A**synchronous **S**tochastic **G**radient **D**escent
AWD-LSTM - **A**SGD **W**eight-**D**ropped **LSTM**
BERT - **B**idirectional **E**ncoder **R**epresentations from **T**ransformer
CAPTCHA - **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
CLAMP - **C**linical **L**anguage **A**nnotation, **M**odeling, and **P**rocessing
DT - **D**ecision **T**rees Classifier
ET - **E**xtra **T**rees Classifier
GBC - **G**radient **B**oosting **C**lassifier
GUI - **G**raphical **U**ser **I**nterface
ICD - **I**nternational **C**lassification of **D**iseases
LSTM - **L**ong **S**hort-**T**erm **M**emory
MCC - **M**atthews **C**orrelation **C**oefficient
MIMIC - **M**edical **I**nformation **M**art for **I**ntensive **C**are
ML - **M**achine **L**earning
NLP - **N**atural **L**anguage **P**rocessing
RF - **R**andom **F**orest Classifier
Ridge - **Ridge** Classifier
RNN - **R**ecurrent **N**eural **N**etwork
ULMFiT - **U**niversal **L**anguage **M**odel **Fi**ne-tuning for **T**ext Classification

Part I

Introduction

In the 1950s, Arthur Samuel created a small computer program for playing checkers. The program used a scoring mechanism and measure each side's chance of winning. He then developed it further so that the scoring was updated based on past results. Arthur Samuel came up with the phrase "Machine Learning" in 1952 (Foote, 2019). Alan Turing wrote a paper in the 1950s about "thinking" machines (Turing, 1950). In the paper, he discusses the topic of learning a machine to be indistinguishable from a human. Additionally, he discusses how to teach the machines small topics, such as chess. The paper ends with the quote "We can only see a short distance ahead, but we can see plenty there that needs to be done." (Turing, 1950), which indicates how ahead of his time he was. Two years later the Hodgkin-Huxley model, which shows how the brain uses neurons to form an electrical network, was presented. This mathematical model gave scientists a greater understanding of how the brain neurons worked before they had a detailed understanding of what the membrane of a nerve cell looked like (Swarthmore Edu, n. d.). These are some of the events that transpired and became inspirations to Natural Language Processing (NLP) and Artificial Intelligence (AI) (Foote, 2019). AI has been heavily debated over the years and from it, different questions arise, such as how great of an understanding can a computer attain? Can we create an AI indistinguishable from humans, and is it safe to do so? How well can computers understand texts today? A wide large collection of articles and experiments discuss these problems.

A recent study shows that NLP is a promising method for extracting information from text, using language models such as BERT. (Xue et al., 2019). The study uses text written in Chinese, which is one of the most advanced languages. This indicates the potential of NLP, which is likely to be greater for simpler languages such as Indo-European languages. The intention behind the research in this paper is to use NLP to interpret medical records in such a way that classifications can be performed on the data. The hope is to demonstrate how NLP can act as a powerful tool in the medical field. Using these techniques, a computer can reveal patterns that are virtually undiscoverable by humans (Malmasi and Turchin, 2019). In addition, if such documents are formatted by a fixed set of rules, they become more maintainable and easier to analyze.

Natural language processing is a domain within AI which is about interpreting human languages. It has many relevant applications today, such as understanding the sentiment behind pieces of language, advanced language translation, text summarization and speech recognition (Shaalan and Tolba, 2018, pp. 3, 101, 435) (Howard and Gugger, 2020).

As of today, there exist several language models with a number of techniques used trained for different languages. One such technique, called ULMFiT, recently achieved state-of-the-art results with their transfer learning technique (Howard and Ruder, 2018).

On the matter of clinical data and machine learning, the computer science part has been lagging behind for many years for at least two reasons. First of which is the need for data. There are very strict privacy concerns for sharing medical records outside of a medical institution. Second, since medical records are written by several institutions, they do not follow a strict format. This makes it harder for a computer to parse and use (Wang et al., 2018).

Among data sources that will be explored, is the set of clinical notes in the MIMIC-III database (Johnson et al., 2016), (Goldberger et al., 2000). Based on the data in MIMIC-III, we will attempt to predict whether a person will die at the hospital.

Part II

Theoretical Background

1. Natural language processing and deep learning

We assume that the reader has a basic understanding of machine learning. Rather than explaining the basics of machine learning we will focus on the parts that are most relevant to the present work: deep learning and its application to natural language processing. For readers unfamiliar with machine learning, we recommend reading at least the first chapter of "Foundations of machine learning" by Mohri, Rostamizadeh and Talwalkar (Mohri et al., 2018). Other useful books include "Hands on machine learning" (Géron, 2017), "Deep learning" (Goodfellow et al., 2016), "Natural language processing with Python" (Bird et al., 2009).

A useful point of view on deep learning is that it "is a computer technique to extract and transform data—with use cases ranging from human speech recognition to animal imagery classification—by using multiple layers of neural networks" (Howard and Ruder, 2018). Each layer of an artificial neural network processes the inputs from the previous one(s) and progresses them on down the hierarchy of layers. Deep learning has caused several fields of computing to move forward in a rapid pace.

Natural Language Processing, or NLP, is the science of making the computer understand human language, commonly in the form of written text (Ventsislav, 2018). There are many different approaches and techniques to NLP, we will focus on the techniques applied in the experiments. Words and sentences are translated into numbers so the machine can compute and find connections between them. Some well-known technologies relying on deep learning for NLP are Alexa from Amazon, Google Assistant or Siri from Apple (Tableau, n. d.). Among many other things, these systems can translate spoken sentences into text written in a different language. First, speech recognition is used to extract the text. The text is then sent to an algorithm that uses NLP to translate the words into the target language and returns a written result. Another great example one might have encountered are chat bots that return automatic response based on a query input from the user. There are many different techniques used when working with NLP and the ones used in thesis will be explained in this chapter.

1.1 Vocabulary, tokenization and numericalization

Tokenization is the process of converting the text to a list of words. In general there are three ways of tokenizing the data:

1. Word-based
2. Subword-based

3. Character based

The first approach consists of splitting the sentence on its spaces, and then it applies language-specific rules. The application of language-specific rules tries to separate parts of meaning even when there are no spaces. One example of this is turning "it's" into "it 's". Usually, the punctuation is also split into separate tokens. The subword based approach will split the words into smaller parts, this is done based on the most commonly occurring substring in the text. An example might be "beginning" be tokenized as "begin n ing". Finally, character based splits the text on each individual character in the text. See (Howard and Gugger, 2020) for further details and examples.

The vocabulary takes a subset of all the tokens to give to the model, so that the model does not guess from every word that exist in the language. This subset is the tokens that are the most frequent in the given texts. This is what a model will use to make predictions in text problems. The size of a model's vocabulary determines how many tokens it can store and how well it can predict the next token in a text.

For the computer to understand written text, it has to be translated to numbers, this is called numericalization. Numericalization can be done in different ways on a technical level, but the essential part is having something that maps a number to a token.

1.2 Transfer learning

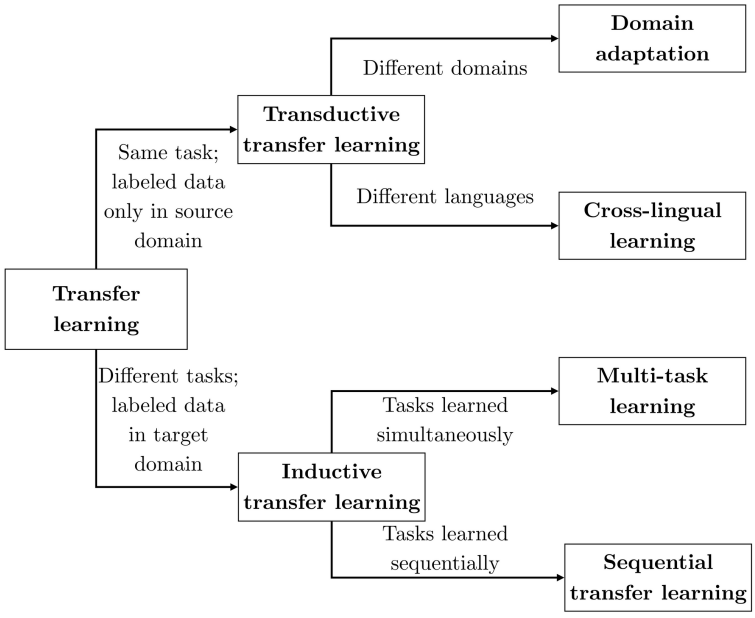


Figure 1.1: A taxonomy for transfer learning (Ruder, 2019)

Transfer learning is the process by which a model is pre-trained on a data-rich task before being fine-tuned on a specific task (Raffel et al., 2019). This is the simple concept of transfer learning. The art of pre-training a model is when the weights in each layer is fit to a large model, which will have a deeper understanding for a language. Pre-training makes the model work better for a set language domain. Transfer learning is essential in creating a state-of-the-art NLP model, since the technique makes the training of a model much faster. For example, by using a model trained on understanding the English language, that model can be trained further to predict the author of a given text. This process reduces the time to train significantly. In text cases the pre-trained models are already trained on great amounts of texts and they have a fair grasp on linguistic concepts and grammar. However, language varies much between domains, such as informal direct messages between friends and formal subject notes in a hospital. It would be nearly impossible for a model today to consider every domain of language while maintaining accuracy and efficiency. In order to make good use of the pre-trained model, it is very common to re-train the last few layers, as to tailor it for a specific purpose. There exists many different models, especially for broader subjects, and by utilizing transfer learning there is no need to train all the layers and weights each time.

Figure 1.1 shows a taxonomy for transfer learning in NLP from a PhD thesis written on the subject of transfer learning in NLP (Ruder, 2019). The two main distinctions is between inductive transfer learning and transductive transfer learning. Inductive transfer learning is to train a model on a different task than what the language model was originally trained on. Transductive transfer learning is to train the language model on the same task. The figure then depicts various ways those can be used to transfer what has been learned to even more specific tasks.

1.3 Artificial neural networks

Artificial neural networks (ANNs) are a class of machine learning algorithms. An ANN takes an input and uses multiple layers and activation functions to generate a prediction from that input. It is trained in iterations and uses an error metric, calculated after each iteration, to update its layers to achieve more accurate result. It is built up of neurons which takes a set of weighted inputs and then applies an activation function to them, and returns an output. Neural networks are vaguely modeled after the human brain, and are in that way designed to recognize patterns.

1.3.1 Layers

An artificial neural network consist of multiple layers with different usage. The **input layer** contains the data inputs that your model uses to train. **Hidden layers** are found between the input and output layers. Such layers applies an activation function before they pass on their results. There are usually multiple hidden layers, which is what “deep” in “deep learning” refers to. ANNs are called fully connected if each neuron receives all the outputs from every neuron in the previous layer, and in turn sends the results to all the neurons in the next layer. The last layer in the model is the **output layer**, and this is the layer

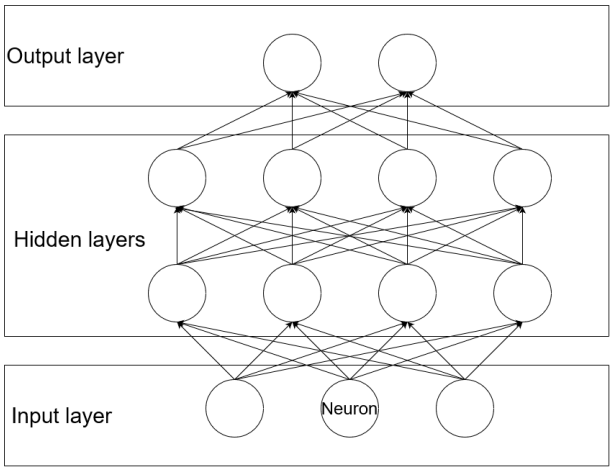


Figure 1.2: A simple neural network. Neurons in each layer are depicted as circles and arrows are signaling the connections between them.

that returns an output representing the prediction of the model.

1.3.2 Weights of activations

The strength of the connection between two neurons are controlled by the **weights**. The inputs are multiplied by the weights when going from one neuron to the next. This in turn will determine how much influence the inputs will have on the output of that neuron. Inside the layers are **activation functions** that modify the data before sending it to the next layer. This is what makes the model able to create complex relationships between the features.

1.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural networks which work on sequences of arbitrary lengths, unlike many other neural network types which rely on a fixed-sized input (Géron, 2017, pp. 379-410). For instance they can take sentences as inputs of varying lengths, making them especially useful for NLP tasks. A RNN is very similar to a normal neural network, the main difference being that a neuron both have connections pointing forward in the network, and connections pointing backwards (see Fig. 1.3). A layer of neurons receives in turn two sets of weights, one new set as the input, and the set it produced as output in the last step. This is why we say that these neurons have a form of memory. A part of the network that maintain some state across time steps is called a "memory cell", this can be whole sections not just a single neuron. These are the hidden units of the RNN.

Long Short-Term Memory (LSTM) is one example of an RNN. It is considered more advanced than a basic RNN architecure. It replaces every hidden unit in the neural network with a LSTM cell and adds a new connection between each cell called cell state (Hochreiter and Schmidhuber, 1997). This was created as an attempt to fix one of the major flaws of RNNs: the vanishing/exploding

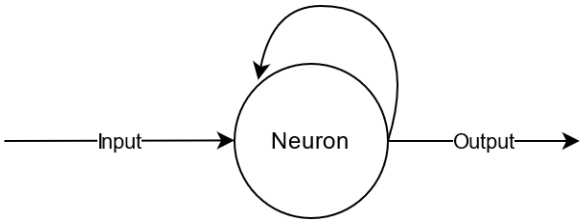


Figure 1.3: A recurrent neural network neuron with output given back to itself.

gradients. An exploding gradient happens when the derivatives are multiplied together for each hidden layer, and the derivatives are larger than the gradient, the gradient will eventually explode as it increases exponentially while traversing down the model. A vanishing gradient on the other hand is when the derivative is smaller than the gradient and eventually vanishes. The special cell in the LSTM network that replaces the hidden units in the RNN is a "memory cell", which can store information in memory for an extended period of time. How long this information is stored and how it is transferred are controlled by three gates in the cell. The input gate decides how much information passed from the last sample will be kept in the memory. The output gate control what amount of information is sent to the next layer. Finally, the forget gate determine what part of the cell state is to be thrown away or rather, forgotten.

1.3.4 ASGD Weight-Dropped LSTM

ASGD Weight-Dropped LSTM (AWD-LSTM) is an improved RNN architecture for training a language model and builds upon the LSTM cell. AWD-LSTM was presented as a strategy to regularize and optimize LSTM language models in 2017 (Merity et al., 2017).

ASGD stands for Averaged Stochastic Gradient Descent and are similar to stochastic gradient descent (Zheng et al., 2016). The difference being that it returns an average of the iterates, instead of only the last iterate. It is determined as follows:

$$\frac{1}{K-T+1}\sum_{i=T}^{K} w_i, \quad where \;\; T < K,$$

where K is the total number of iterations and T is the user-specified threshold. One could also set the threshold based on the models performance over several cycles, setting it non-monotonically (NT-ASGD). In this case the threshold will be changed if the validation metrics does not improve for multiple cycles, to remove randomness in training it is not changed as soon as the metric doesn't improve.

AWD-LSTM combine this with Weight-Dropped LSTM, which uses the theory of DropConnect (Wan et al., 2013) on the hidden-to-hidden weight matrices. Which helps to prevent overfitting over the recurrent weight matrices of the other RNN cells. This is not at the cost of training speed, as the dropout operation is only applied once to the weight matrices. Unlike normal dropout where a random subset of activations are set to zero, the DropConnect approach sets a random subset of weights to be zero, all within the network.

1.3.5 Language models

A language model is a probability distribution over a sequence of tokens. When predicting with a language model the token that the model assigns the highest probability is the one that is chosen. Therefore it may be lost whether or not the model was really confident or not in that prediction. Most language models prediction is calculated as a product of the preceding tokens (Huyen, 2019):

$$PP(w_1, w_2, ..., w_n) = \Pi_{i=1}^{n} p(w_i | w_1, ..., w_{i-1})$$

Language models can be set up as a self-supervised problem, because a text of arbitrary length can use the next word as the label for the model. Neural language models use word embedding, a continuous representation of words, to make their predictions. Word embedding is a technique for NLP which maps words to vectors in such a way that their semantic relationship is captured. This is a very useful property that allows the model to understand the language better. Adding word embeddings as an extra feature is likely to improve the performance of any NLP task (Wang et al., 2018).

1.3.6 Classification models

Similar to the language model, a classification model has a set of labels it can make predictions from, language model having the vocabulary. Classification problems falls within supervised learning because they compare their predictions with given labels. With the given labels the model can find and learn a correlation between the inputs. The labels for a classification problem can i.e. be the author of a book, from a library of a finite number of known authors. One way to create a classification model is by removing the last layer of the language model, add some layers so the output becomes a vector of equal size to the number of labels, and then tune it to create predictions on a set of labels. Unlike a regression task, a classification task is discrete and rely on a finite number of output.

1.4 Fastai: a deep learning library

Fast.ai was founded in 2016 by Jeremy Howard and Rachel Thomas. Fastai is a deep learning library developed by fast.ai that, among other things, provides tools for easily training text classifiers. It is a leading force within enabling people with relatively little technical background and expertise to create close to state-of-the-art machine learning systems. Jeremy Howard and Rachel Thomas have also created multiple free and open courses based on the library that they, and many other contributors, have built together (Howard and Gugger, 2020) (fast.ai course[1]).

For tokenization and building the vocabulary, fastai relies on spaCy (Honnibal et al., 2020), an open-source library designed for NLP. The spaCy based tokenizer class in fastai uses word-based tokenization and has a vocabulary size of 60,000 tokens by default. The least frequent words in the text corpus will be replaced with "xxunk", a special fastai token. There are multiple different

[1] https://course.fast.ai/

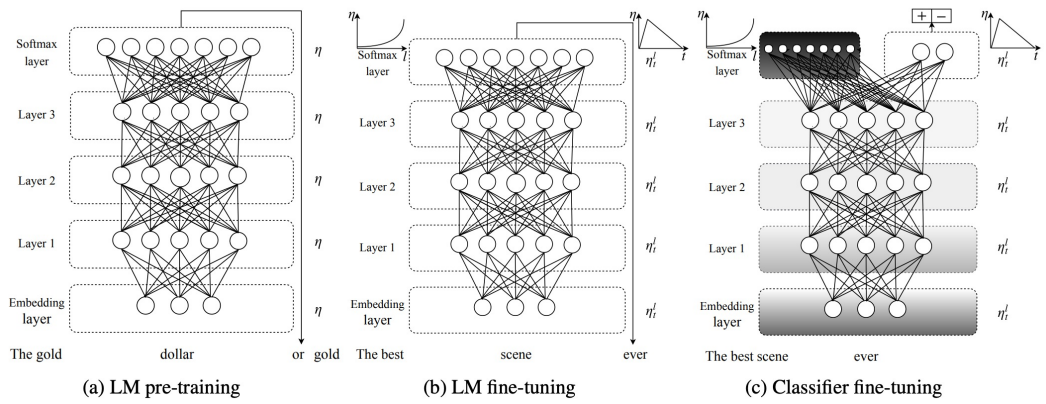


Figure 1.4: The 3 stages of the ULMFiT technique on a neural network using transfer learning to fine tune a text based model. From ULMFiT paper (Howard and Ruder, 2018)

fastai tokens and all of them start with "xx" so they are easy to spot, the "unk" in "xxunk" is short for "unknown".

Fastai makes heavy use of transfer learning to improve their models and results, in both its computer vision module and its NLP module. In the fastai course they show both how to build the model from scratch and how to use their pre-trained models to even better results while saving time. One of the fast.ai founders, Jeremy Howard, took part in the creation of ULMFiT (Howard and Ruder, 2018), a transfer learning-based approach to NLP further described in the next section.

1.5 The architecture of ULMFiT

In 2018, a language model technique called ULMFiT (Universal Language Model Fine-tuning) based on inductive transfer learning was proposed by Howard and Ruder (Howard and Ruder, 2018). ULMFiT outperformed existing NLP techniques, becoming the state-of-the-art model for multiple challenges related to sentiment analysis, question classification and topic classification. It achieved such results by being pre-trained on a big, generalized corpora, and then fine-tuning its word embeddings on a more specific language domain (Howard and Ruder, 2018). It was the first successful use of transfer learning in deep learning models for NLP. Computer vision (CV) had been using transfer learning heavily and successfully for several years, but it wasn't until ULMFiT appeared that transfer learning revolutionized NLP. Figure 1.4 shows the three stages of the ULMFiT technique.

1.5.1 General-domain pre-trained language models

First, the authors developed a pre-trained language model that is focused on the bigger picture which is understanding how the language itself is written. They created this model based on Wikipedia articles, more specific 28,595 of them, the so-called WikiText-103 data set (Stephen et al., 2016). The idea is

to create a model that understands the language that then can be trained on a more specific task.

1.5.2 Target task LM fine-tuning

Using transfer learning the language model that has been pre-trained on the bigger picture will then be fine-tuned to a more specific problem. This problem should have a smaller size than the original and be center around a smaller task, otherwise the usage of a pre-trained model is not that necessary. They further propose two ways of tuning the model using **discriminative fine-tuning** and **slanted triangular learning rates**, learning rates dictates how much a layer is changed each iteration. Discriminative fine-tuning is the process of having different learning rates for different layers, by having high learning rate for the last layer, and lower learning rate for the earlier ones as they may already have a good tuning for the task. Slanted triangular learning rates is a technique to help the model adapt and converge quicker to the specific task at hand. The idea is to start at low learning rate, and then increase it linearly for each iteration. And after a number of iterations the learning rate is linearly lowered again at a slower rate.

1.5.3 Target task classifier fine-tuning

The last stage in this method is to create a classifier augmented from the language model that has been targeted towards the same task as the classifier. There are several different techniques presented for this approach. Including **concat pooling**, **BPTT for text Classification**, **Bidirectional language model** and the last one, **gradual unfreezing**, will be described in minor detail. Gradual unfreezing is the process of not training every layer for each iteration. It is in a sense similar to discriminative fine-tuning as it does the biggest changes in the last layer. It starts by unfreezing the last layer and keeps the remaining frozen (i.e. not updating the parameters of these layers during training), and then train the model on all unfrozen layers. Then gradually start unfreezing until one iteration it unfreezes the whole model and fine tune all of them. This process will minimize the risk of "memory loss" loosing information already gathered.

This is a brief explanation of the stages, and the full explanation can be found in their paper ULMFiT (Howard and Ruder, 2018). This method will be used in this paper to create the models. So having an understanding of this process is essential.

2. Evaluating and understanding model predictions

The term "Black box" is often used in the context of AI to say that the computations the models do are incomprehensible to humans due to their magnitude. However, there exists different methods and techniques to understand the model better and shine a light on the black box to expose some of its contents. While this will not contribute directly to the model's performance, understanding a model can be a very powerful debugging tool. In certain contexts, it is essential to know why and how a model arrived at its conclusion. More on that in Chapter 3. With this in mind it is also very helpful to track the performance of the model. These are called metrics which is quantitative assessment commonly used for comparing, and tracking performance or production (Young, 2020). Displaying the metrics during training will not only help understand the model, but it will help keeping track of what changes made the biggest impact on the predictions. To not get fooled by how well one metric performs, even though it might be a good indication, it is suggested to look at multiple metrics. This will help paint a more accurate picture of the overall performance.

2.1 Language model metrics

The appropriate metrics to consider depend on what kind of models that are being evaluated. In the context of language models the metrics we will look at for evaluation are accuracy, perplexity and loss.

2.1.1 Accuracy

The accuracy is measured by how often the model can predict the correct next token in a sentence. If the model has an accuracy of 20% that means that when given a new sentence, the model can predict the next word one fifth of the time. Which, considering how many words there typically are in a vocabulary, is pretty impressive. This metric is very common for people to look at, and it gives a good impression of whether the model will perform well or not. It can also tell us if a model has been overfitted on the training data, but that is a bigger problem when creating the classifier.

2.1.2 Perplexity

Perplexity is the exponential of cross entropy loss when calculated on a probability distribution. This is a measurement based on the probability distribution and is mathematically defined as:

$$PPL(P,Q) = 2^{H(P,Q)}$$

Where H(P,Q) is cross entropy loss, Q is the distribution learned close to the empirical distribution P for the language. The lower the number the better the

score (Huyen, 2019). As a language model is a probability distribution over a given text or entire sentences. The perplexity tells us how confused the model is. For example a perplexity of 10 would mean that the model is as confused as to choose freely and evenly between 10 different choices for each word. Thus a lower score would mean that the model is better and predicts more accurately for each word evaluated.

2.1.3 Loss

Loss, also known as error or cost function, differs a bit from metrics, and there is an important distinction to be made. Because the loss' sole purpose is to be a function that the system can use to measure performance and update its weights automatically. Loss is used in gradient descent and backpropagation to update the weights and therefore has to, unlike the other metrics, be differentiable. A metric is mostly defined for human consumption, as in being easy to understand for a human, while a loss forms a suitable metric only in some cases (Howard and Gugger, 2020), for example the mean squared error loss in the case of regression.

2.2 Binary classification model metrics

The performance of a classifier can be difficult to interpret because there are many ways of looking at the result. This is why it is very common to consider multiple metrics to attain a more accurate understanding of the model's efficacy. All metrics for binary classification problems can be derived from a simple two-by-two contingency table, as seen in Figure 2.1. Each prediction falls into one of four categories, each with their own quadrant in the figure. FN is short for false negative and means the model predicted negative, but the actual label was positive. TP is short for true positive and means the model was correct in predicting positive. TN is short for true negative and means the model was correct in predicting negative. Finally, FP is short for false positive and means the model predicted positive, but the actual value was negative.

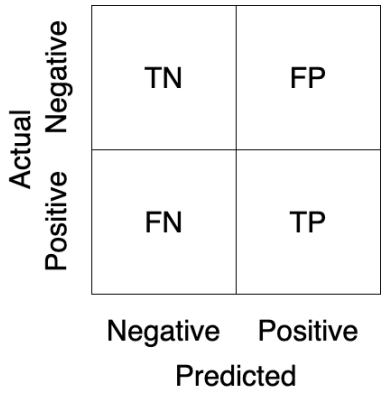


Figure 2.1: Predicted values by actual values in a two-by-two contingency table. FN is short for false negative, TP is short for true positive, TN is short for true negative and FP is short for false positive.

When looking at a classifier we will consider the four metrics recall, precision, F_1 score and accuracy to evaluate the model.

2.2.1 Recall and precision

The **recall** is the proportion of true positive labels out of all the actual positive labels, whether they are true positives or false negatives. It is expressed as:

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

The **precision** reveals the proportion of true positives of all the positive predictions.

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

Precision and recall are best intuitively understood through an example. Consider a skewed data set in the favor of negative labels. If a model predicts all labels to be positive, it will score a great recall because all the actual positives were predicted to be positive. However, the precision will suffer because out of all the predicted positives, most of them are actually negative. On the other hand, if a model is very conservative in predicting positive labels, it will score a high precision at the cost of a terrible recall. The precision will be high because the few predicted positives are very likely to be actual positives and the recall will be low because there are many false negatives (Shung, 2018).

2.2.2 F_1-score

This is why F_1-score is useful. The recall and the precision are indications of a model's efficacy, but they are not comprehensive in isolation. The F_1-score is simply the mean of the recall and the precision (Shung, 2018).

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

2.2.3 Accuracy

When looking at accuracy on a classification model, the idea and calculations are quite simple. The accuracy reveals the proportion of correctly predicted labels (Ekanayake, 2019).

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

2.2.4 Matthews correlation coefficient

Matthews correlation coefficient (MCC), also known as phi coefficient, is a way to measure the quality of a binary classification model (Boughorbel et al., 2017). MCC computes the correlation between the two variables, the higher the correlation between true and predicted values, the better the predictions are. The

computation of MCC is given by:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

This results in a score ranging from between 1 and -1, where 0 is signaling that the classifier is no better than a random one. A score of 1 would indicate a perfect correlation between the positives (FP = FN = 0), and -1 showing that the negative correlation is perfect (TP = TN = 0) and a simple reverse of the classifier would give a perfect result.

The advantage of MCC to F_1-score and accuracy is that MCC takes into account an imbalanced class distribution. This provides a more comprehensive measurement of a model's performance (Chicco and Jurman, 2020).

2.3 Confusion Matrix

The confusion matrix is a representation of the four outcomes of the model, TP, TN, FP and FN. As seen in Figure 2.1, the confusion matrix displays a plot with numbers in each of the four cells indicating the number of predictions that fall under that category. This is a quick and simple way to see how "confused" a binary classification model is when it is predicting. Consider an imbalanced data set where 5% of the cases are positive, and the rest are negative. If a model only predicted negatives, it would have an accuracy of 95%. This matrix would quickly reveal why the model scored so high and show that the model in some sense did not find any relations.

2.4 Feature Importance

Feature importance reveals which features are the most useful to the model. There are many ways to calculate feature importance, and they may reflect different sides of the importance of a feature. Feature importance can reveal many things about a model and its features, one of which being if a feature is not being used, or barely considered at all. This may be fine, and it can help to remove the amount of features used and in turn then reduce the time a model uses to train or predict. In other cases this can reveal that the model does not use a feature that has been scientifically proven to affect the outcome of the task it is given. This gives a good indication that something needs to be done to the model, and figure out why the model does not use that feature as much as it should.

3. Privacy and ethics in machine learning

When working with sensitive data, it is always important to consider the privacy and ethics of the subject. Medical data is very personal and should always be handled with utmost care and privacy.

3.1 Privacy

Medical data is especially sensitive and are protected by a variety of different laws, and therefore it is both difficult to get data, and important to take good care of it when you do. Doctors and nurses that work with the patient have a medical confidentiality, therefore available data should always be anonymized. Working with anonymized data can in some cases prove difficult, it really depends on how it is done. But in most cases it is only the sections that can lead back to that person that has been removed, or replaced, so that the other features are easily accessible.

Privacy can unfortunately take a toll on progress, it can be hard to get access to data, and in many cases they do not have the data available for processing. In some way the struggle to create and make anonymized datasets available punishes advancement within the medical field. Working with machine learning we are reliant on having quite a big amount of data to generate good results. Therefore further development are dependent on that new anonymized data will keep being made semi-publicly available. One of these major datasets are MIMIC, further explained later in Section 5.1, where one are required to go through a preliminary course to get access to the data. This course explains all the details in how to handle the data and what precautions to take when working with them (The CITI Program, 2017). MIMIC is therefore publicly available for everyone, but having the course ensures that the users have been explained in detail what it is and how to use it, so that they can at any time refer to the agreement if the privacy has been breached by a user.

3.2 Ethics

Computer parts are in constant development and they can compute quicker and quicker, and as the computers get better there will unfortunately always be someone out there that wish to use that power with malicious intent. Some might try to use the unlocked power to create serious scams. Like Alan Turing asks in his article imitation game (Turing, 1950), can we create a machine that is indistinguishable from a human being. Take for instance this article from BBC "Fake" Amazon workers defend company on Twitter" (BBC, 2021), where it was uncovered that Amazon has been creating fake accounts for employees to try spread positive information about the workplace and their wages which may or may not be true. Imagine if they could create hundreds of these accounts

with an AI model, and it was close to impossible to determine if it was created from a program or typed by a person. The impact this could have, and the false information that could be spread is massive. Fortunately this is not the case as of yet, but the more we move into everything being on the internet, and the creation of quantum computers it is very important to always take what you read on with a pinch of salt. This could also unfortunately be used to a more serious degree as in corruption or criminal work. The difficulty of detecting this is a huge problem, if we create a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) to automatically detect it, the "offender" could keep training the model to learn how to not get detected. It is an endless loop and it is something that will become more and more relevant as the computers get better and better.

When working in the field of AI and text there are multiple precautions to take, and the need for explainable AI is there. Being able to understand what the model looks at, and how it works to always know that what it does is not out of line is important. What if we some time in the future are creating a model that in the end decides whether or not a person who is on life support will make it. We should be able to understand why that machine predicts so, and be able come closer to a conclusion in whether or not this is the case. Even though the machine makes good predictions, the usefulness in understanding why, is definitely there. And the question stands in can we ever trust only the machine on this, or should there always be human input in making the decisions? With big decisions that for example involve a human life this question is more difficult than maybe for a somewhat smaller decision like whether or not a bone in the arm is broken and need a cast. If your models predict a patient to die, should the patient know this or should this be hidden from them? Some might want to know and others not.

Another thing to always keep in mind is that data contains error, it is nearly impossible to have a perfect data set. The machine learning algorithms are prone to bias. And there are several different biases that can occur, some are more common than others. Humans are also biased, so why would we care about this when working with machines? We are always trying to create the best product with the tools that are available, and algorithms have different use cases than that of a human. Representation bias is for example when an algorithm is trained on mostly white males, and therefore yield a 90% result for white males, while the same product has a 60% accuracy for woman of darker skin colour. Racial bias, if an algorithm discriminates based on racial appearance, for example if a person with a African American name apply for a house loan and the algorithm gives them a worse interest rate than one with an American name with the same data otherwise (Howard and Gugger, 2020). These are just some example of biases that can occur. Measurement bias is when the algorithm is affected by peoples behaviours creating unlikely connections. For example a person who goes a lot to the doctor will have a bigger effect on how an algorithm learn what symptoms can indicate early stages of a certain decease. Because these people will go in for anything, a lot of "false" symptoms that do not have anything to do with that decease, will be weighted as important for the algorithm, or as patterns. These are only some of the different biases that

can happen when training and gathering data. A machine learning algorithm is very likely to have bias, and it is therefore important to be aware of this and be able to understand why the model predicts the way it does. Adding the best of both worlds and helping the model yield better results based of research is a good way to remove some of these biases.

Part III

Research questions

4. Research questions and hypotheses for the experimental work

Natural language processing and deep learning is already a widespread technology and one of the biggest components that propels us into the information age. As discussed, the medical domain often lags behind mainly due to the severity and sensitivity of the domain.

4.1 Examples of related work

First we will look at some related work within the field that answer some of the questions we could have for this topic. However, they do not have an answer for the questions we want to explore, they only indicate the existing possibilities.

Evaluating MIMIC-III clinical notes. The article *"An Empirical Evaluation of Deep Learning for ICD-9 Code Assignment using MIMIC-III Clinical Notes"* (Huang et al., 2019) takes on the clinical notes from the MIMIC-III dataset that also will be used in this thesis. Here they aim to create a deep learning system that can automatically map the clinical notes to the ICD-9 medical codes. To then in turn be able to bill the patients accurately and with the right medical record.

Another article that aims to report the performance of NLP on mapping clinical free-text notes to medical codes is *"Natural language processing of MIMIC-III clinical notes for identifying diagnosis and procedures with neural networks"* (Nuthakki et al., 2019). They employ the deep learning method ULMFiT on the MIMIC-III data set, to compare with conventional machine learning models. Their models achieved accuracies from 60 to 80 percent, when predicting on ICD-9 codes. They hope that the model can assist humans by saving time, minimize cost and eliminate errors.

Transfer learning in medical NLP. In their article *"Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets"* (Peng et al., 2019) they create a set of benchmarks for language models trained within the biomedical field. They call it the Biomedical Language Understading Evaluation (BLUE) benchmark. Here they compare and benchmark five tasks with ten datasets to see the results of the different types of language models trained on the medical field.

Clinical Language Annotation, Modeling, and Processing. Machine learning is becoming more important to the progress within medicine. For this to become useful to doctors worldwide the programs has to be easy and simple for them to use. There has been created a GUI using NLP to make it easy to use for people with a less technical background. This system is called CLAMP (*Clinical Language Annotation, Modeling, and Processing*). CLAMP is

described as a toolkit providing state-of-the-art NLP components, with a user-friendly graphic user interface that can help users quickly build customized NLP pipelines for their individual applications (Soysal et al., 2017). They display a couple of tests they ran using this system, and the results are in the 90% accuracy range, which is promising for a program created for the "everyday human".

Improved classification with language model pre-training. In the article *"Improved Hierarchical Patient Classification with Language Model Pre-training over Clinical Notes"* (Kemp et al., 2019), they propose a pre-trained hierarchical recurrent neural network that utilizes information found in minimally processed clinical notes. They found their RNN improves performance for discharge diagnosis classification tasks compared to models that have no pre-training and that treat the clinical notes as unordered collections of terms. In addition, they apply an attribution technique that reveals what words were key to a prediction and uncovers the importance of the words' nearby context.

4.2 Can natural language processing with deep learning pick up on linguistic features in clinical notes?

Motivation. This research question has been addressed before in other research papers (Nuthakki et al., 2019) (Kemp et al., 2019). Still, it is important to raise the question once again and give it extra attention because it is the foundation for the other experiments in this paper. If the language model does not perform well by its metrics, it will likely become a bottleneck for the other experiments as well. By the same token, if it performs excellently, it may increase the performance of the other models and contribute to separate the results even more from baseline results. Because the language model's predictions are so fundamental to the other models, natural language processing with deep learning is considered to be extracting useful information if it can be used to accurately predict the domain language.

In addition, an estimated amount of 80% of all clinical data is unstructured (Martin-Sanchez and Verspoor, 2014.). Prior to recent advancements in deep learning, computers have not been very efficient in putting unstructured data to use. Finally, this will also serve as an example for one of many cases where language models can attain feasible results without high-end equipment.

Hypothesis 1

A language model trained on clinical notes will be able to predict the next word most of the time and generate similar texts to that of the notes.

4.3 Can a text classifier built from the language model extract useful information from clinical text?

Motivation. In order for the research question to be well defined, it is necessary to define what constitutes as capable of extracting useful information. It stands to reason that if a text classifier performs better than a random classifier that uses a stochastic model. The specifics of each experiment will be discussed in their separate sections.

Although a language model capable of picking up and generating the patterns of language is an autotelic feat, it is not immediately practical. The language model is still outputting a single word at a time, which is not even very difficult for humans. The challenge is to extract and make sense of the information embedded in the text. Many subtle patterns and tendencies may very well go over our head, because we are incapable of processing the amount of variables and possibilities present in the texts. Classifiers seem like a natural solution for these sort of problems.

To summarize, because a language model in of itself is hard to put to practice, it is helpful to compliment it with a text classifier. As mentioned, 80% of clinical data is unstructured (Martin-Sanchez and Verspoor, 2014) and this is a vital part for computers to use that data. A text classifier can be trained to predict a wide array of classes, such as ICD codes and procedures (Nuthakki et al., 2019).

Hypothesis 2

The hypothesis is that the text classifier will be able to make predictions that performs better than a random classifier that uses a stochastic model.

4.4 Can the extracted information be used in combination with other data?

Motivation Data will very often not exclusively be structured or unstructured. Structured is naturally preferred because it is concise, clear and accessible, both to humans and to computers. Nonetheless, some data such as answers to open-ended questions or movie reviews, cannot be easily structured. To make use of free text data is very valuable. However, extracting the most impactful features from free text and combining them with structured data can take the performance of a model even further.

Hypothesis 3

The hypothesis is that combining free text data with structured data will result in better predictions compared to each of them on their own.

4.5 Can the deep learning models be explored and be made sense of?

Motivation One of most crucial parts of applying deep learning of any kind in healthcare and medicine involves understanding the model. It is critical to be able to understand why a model produced its prediction. Contrary to common belief, deep learning need not be a black box for which things outside our comprehension take place. The reason for this question is to able to explain what features had the most impact on the predictions, and translate the features for humans. This might involve trying to discover what exact words or phrasing the model picked up on and weighted as important.

Hypothesis 4

The hypothesis is that the most important features will stand out and it will be possible to see how much they weigh in. However, these are still only pieces of the puzzle, and it will be hard to get anything more than a strong impression.

Part IV

Experimental work

5. A language model trained on free-text clinical notes

To discover whether we can extract useful information from clinical notes, we wish to pursue training and using a tailored language model as a backbone for text classification. Natural language text is considered an unstructured source of data, which means it is not easily accessible for computations done by a computer. It is estimated that 80% of all currently available data falls under the category of unstructured data, which emphasises the importance of being able to make use of unstructured data. For example, electronic health records are intended for humans and not computers, but still contain a plethora of information useful to make predictions (Martin-Sanchez and Verspoor, 2014.)

5.1 The MIMIC-III data set

The results of (Huang et al., 2019) acts as a motivation for the approach taken in this chapter. The authors were able to construct a deep learning-based language model that led to models that could outperform state-of-the-art traditional machine learning models when tasked with predicting the top ten ICD-9 codes on text in the MIMIC-III ("Medical Information Mart for Intensive Care") data set.

MIMIC-III is a database containing clinical information from a large number of patients admitted to a tertiary academic medical center in Boston, MA, USA (Johnson et al., 2016), (Goldberger et al., 2000). This collection of data is groundbreaking and important for two reasons. First, clinical data is very difficult to obtain and make use of, due to its sensitive nature. Second, a shared data set provides common ground for researchers to compare results and discuss different approaches, a key aspect of reproducible research.

Extracting the data from MIMIC-III is an intricate task, as it is a large amount of data structured in a very specific way. We used the *MIMIC extract tool* of (Wang et al., 2020), which is an open source project and the result of a research paper that can be used to extract and perform groupings of values with sensible default while still being configurable.

Note that at the time of writing, the most recent MIMIC database is MIMIC-IV(Johnson and Mark, 2021). However, using MIMIC-III made things easier for this project as support for MIMIC-IV had not yet been added to the MIMIC extract tool. There are also currently few published papers based on the very recent MIMIC-IV, making comparisons with the literature harder.

5.2 Training a language model

Our goal is to construct text classifiers based on the clinical notes in the MIMIC-III database. An approach that has shown to be quite powerful is to use deep learning-based classifiers constructed from *language models*, see Section 1.3.5, using the *AWD-LSTM* architecture, see Section 1.3.4. After designing and

training a language model, i.e. a model tasked with predicting the next word from a text sequence, one can use the learned weights in parts of the model architecture as a basis for a text classifier. Creating a language model is a long process and will be described more in detail in this chapter.

5.2.1 Vocabulary and tokenization

As described in Section 1.1 there is a need to decide the size of the vocabulary. And to decide what is a reasonable size is very dependant on the text that is to be used in prediction. The size of the dataset used for fine-tuning has an impact on how small of a vocabulary we can get away with and still get good results.

5.2.2 Metrics

Metrics are an important part in training, and in the assessment of how well the model both performs and improves over time. The metrics used when evaluating the language model are *accuracy*, *perplexity* and *loss*, which are described in detail in Section 2.1.

5.3 Structure data and gather the clinical notes

The MIMIC data set contains a lot of data (6.5 GB compressed). To use the data there is a need to extract exactly what is needed to the project at hand. This is a huge task and as stated earlier this has already been done with the MIMIC-III data set. MIMIC extract is one of them (Wang et al., 2020).

5.3.1 MIMIC extract

The objective here was to use MIMIC extract to structure the data, to be able to create a language model that was trained on the medical language used by doctors and nurses in the clinical notes. MIMIC extract is an open-source software that can be ran to structure the data from MIMIC-III, running this program took approximately 16 hours when it ran all the way through. [1]

Running MIMIC extract structured the MIMIC-III data set, so it is split into multiple files with different data. One of the files named "notes.hdf" contains all medical notes written in the set. This is the file used for training the language model in the experiments written about in the next sections. To further prepare the data and for easier use with the fastai library, the notes were extracted from this file and put into folders. The folders created represent each stay of a patient, every time a patient is hospitalized their notes are stored in a new folder. The name of the folders are constructed from three ID columns and every entry from their stay is structured in a file which is named after the time the entry was written.

[1] Getting MIMIC extract to run proved to be more difficult than first anticipated, as it was supposed to work out-of-the-box. When running it the first time, the program stopped after 2 hours with an error. After debugging the extraction code for a while we found an error in the code, some of the input data had been accidentally duplicated. Creating a solution for this in the GitHub repository (`https://github.com/MLforHealth/MIMIC_Extract`, 2020), we are now one of, as of 2020, four contributors to this project as the solution for this bug was accepted.

This structure is made because the fastai library can create a language model based on folders and folder names. This is also helpful when creating a classifier for the experiments described later on.

5.4 Experiments

To get the language model more fine-tuned within the field of medical data, all the clinical notes that had been extracted from the MIMIC database. The experimental work will contain different pre-trained models, to see the effect of having a model pre-trained on medical text.

5.4.1 Training a language model from Wikipedia

First the MIMIC data will be used to train a model from fastai that has been pre-trained on the Wikipedia text. *Transfer learning*, as described in Section 1.2, is the big motivator for using this pre-trained model. It has many advantages, such as being a time saver. During the first training cycles the clinical notes has been structured after the stay IDs and the subject IDs, and the model gets all the clinical notes concatenated. It is trained on all the text written in the clinical notes, except for a small validation set used automatically when training. This might seem a bit odd considering the usual approach to machine learning is to set aside a test set that the model has not seen. Since this is not the final model used this is not a problem, and actually it is preferable. One of the reasons we find this preferable is because we want the model to know the whole vocabulary, to get every bit of information that it can. Since the model is to be used in a classifier that uses the same vocabulary we are not worried that it will be overfitted towards the data in this case.

First training the model with 60,000 vocabulary was a process taking seven to eight hours per training cycle and the result from training was within the expected range from a model trained on a whole language. Ranging between 30-40% accuracy we believed it could be better as this is a more specific task. Taking a look into the predictions that the model did, it predicted line breaks almost every other word. This is most likely caused by the structure of how the notes are written. We discovered a line break to represent a shift in what was written about when we examined a small sample of the notes. For example, there could be a date, then line break, then a blood pressure measurement followed by another line break. Also two line breaks were used when concatenating the files before the model read them. Therefore some changes had to be done. The notes was structured into folders as described in Section 5.3.1 and this structure was now used to get the clinical notes. Therefore the model will no longer get big concatenated files with a lot of line breaks to signal a new entry. This will reduce the amount of line breaks the model reads, and hopefully help the model in not predicting as many line breaks. Then the vocabulary was reduced to half the size in hopes of shorter training, and thus the training took four to five hours per cycle. We also use gradual unfreezing, see Section 1.5.3, to yield better result from transfer learning. Both of these adjustments contributed to the accuracy of 63%, see Table 5.1. Looking at the table we can see that the model start to converge towards 63% accuracy and a perplexity of 5.9. Once again some further improvements would have to be made in order for this model to produce

Number of Epochs	Train loss	Validation loss	Accuracy	Perplexity
1	2.034	1.879	0.621	6.549
2	3.125	3.073	0.477	21.607
3	1.940	1.814	0.627	6.137
...	...	...	...	...
6	2.437	2.400	0.570	11.024
7	1.969	1.803	0.629	6.069
...	...	...	...	...
18	1.902	1.792	0.630	6.002
19	1.880	1.791	0.630	5.997
20	1.900	1.789	0.630	5.989
21	1.885	1.789	0.630	5.985

Table 5.1: Results from training on MIMIC with a language model with a size 30,000 for vocabulary.

better outcomes. Since the main goal is the classifier, we will have a greater focus on creating a good classifier and test the effect of having a better language model later. Since the topic is pretty specific, we figured that 30,000 could be an acceptable size for the vocabulary. The model also stopped predicting line breaks every other word. There were still plenty of line breaks, but it looked to be less random.

Number of Epochs	Train loss	Validation loss	Accuracy	Perplexity
1	1.875	1.765	0.637	5.846
2	2.007	1.925	0.616	6.857
3	1.940	1.814	0.627	6.137
...	...	...	...	...
12	1.698	1.587	0.664	4.891
13	1.683	1.587	0.664	4.889
14	1.715	1.586	0.664	4.885
15	1.709	1.585	0.665	4.883

Table 5.2: Results from training on MIMIC with a language model with 60,000 vocabulary. Until the model converged towards a number.

Earlier we changed the vocabulary in hopes of shortening the time it took to train a cycle, hopefully not at the cost of reduced accuracy. We also changed the file structure to reduce the model's tendency to predict line breaks too often, since it was previously concatenated with newlines. We decided to spend the time and resources needed to train a model with vocabulary of size 60,000 to see if the vocabulary change had affected the model accuracy. When cutting the vocabulary in half it is expected to get a result that is almost twice as good (where that is reasonable), as the model will have half the options to guess from. Since our first test got in the 30% range, getting 63% when using 30,000 size for the vocabulary would support getting almost twice the accuracy. Still we there was a need to see what caused that change, whether it was the change

of vocabulary size or the change of file structure. The results from the 60,000 model can be seen in Table 5.2, and here we actually get a slightly better result than the one using 30,000. From the results we got it would be favourable to use the 60,000 model, solely because of it yields slightly better results in all the metrics. Therefore it is useful to take a look into how the models predict, to see if there are any discernible variations in how well the model understands the construction of the clinical notes.

To test these two models we found a random note in the validation data set and gave each model the first four words in the note, to see which one would predict the most reasonable. The note used in this test can be seen in Fig. 5.1.

Fig. 5.1: The text from the random note to predict on

Respiratory Care Note: Patient weaned on high flow t- \n
piece to trach this am for app 2 hours with hypercapnea -PCO2 of 119. \n
She rested on PSV of [**10- \n
18**]% and was weaned back to PSV of [**5- \n
18**]. \n
This afternoon she appears tachypneic and PCO2 is elevated in the 80s. \n
PSV increased to 10 with resulting tidal volumes of only 220- \n
280cc. \n
BS coarse bilat. \n
Received MDIs Q4 with some effect. \n
Plan to support at this time with t- \n
piece trials again in am if tolerated. \n

The models will both predict 100 tokens after the first four words (Respiratory Care Note: Patient), and their result will be compared. First we will look at the model trained once with a vocabulary of 60,000 (Fig. 5.2).

Fig. 5.2: Predictions from 60,000 vocabulary model

Respiratory Care Note : Patient remains intubated and mechanically vented . \n
Vent checked and alarms functioning throughout . \n
Sx for small amount of tan , yellow sputum . \n
Bilateral breath sounds auscultated fairly as per MDI resident . \n
RSBI dropped to 80 \n
this am , both FIO2 increased to 40 % , due to decreased oxygen integrity . \n
[* * first Name11 (name Pattern1) 209 * *] \n
[* * last Name (namepattern4) 70.[**telephone * *] \n
, RRT \n

Even though this prediction might not seem to impressive, the model has changed from generating full sentences in English to generate more line breaks and some dates, even abbreviations and focus on medical terms. This result is not the best but still quite impressive. Moving on to the second model trained on the data set with 30,000 vocabulary, this model has been trained 19 times and on all weights. The results are in Fig. 5.3.

Fig. 5.3: Predictions from 30,000 vocabulary model

Respiratory Care Note : Patient received for bipap x4 hours , was weaned generally to c a 22- \n
45 % . \n
Breath sounds are diminished bilaterally , diminished lower lobes . \n
Last exp 11l thick plugs since tube could not be advanced beyond past few days . \n
ET tube achieved from anesthesia just before breathing . \n
b / s s were coarse \n
p = \n
All > small thick white and tan fluid . \n
Sxn sm amts thin bld secretion . \n
Plan : Continue mech vent as ordered . \n

Note: Due to limitations of the paper, the line breaks have been marked with "\n". An interesting first observation based on this, is that the last model has found that the end of a note has a "plan" on what to do further, signaling that the model might have a better understanding of the bigger picture. We can not say for certain that the other model does not have this understanding, it might just need more than 100 tokens to predict it. It is difficult to say exactly how much better a model is, other than comparing the metrics and the predictions. Creating a conclusion from just one note, would not be very scientific. So generating more results will give a better understanding of whether or not the model is actually better. Out of curiosity lets also see what a model that has not been trained on MIMIC and only Wikipedia will predict. As shown in Figure 5.4 the model predicts more full sentences, and not in the style nor with the vocabulary of the MIMIC texts.

Fig. 5.4: Wikipedia model

Respiratory Care Note : Patient Analysis and Recovery : Subsequent Survival , Health and Education Book : Eleven 69(77 Weather Diagnosis Study Test : An Impact Incident on Steroids of Physicians and Physicians in Connective Tissue Physicians and Medicine Specialists for Research and Research Institutions University of New Jersey Writing . The episode first appears in a panel on the National Health Services Organization 's Psuedocyst Report

Further tests are presented in Table 5.3 and gives a little insight in how the model predicts from real notes, but it is difficult to compare them side by side. One of the reasons being that after the model gets the initial data it continues predicting based on earlier predictions, and when it first trails off, it can really trail off. A thing to take notice of is that even though the models trails off, they both stay within predicting what could be wrong with a patient. A further difficulty is that a lot of the values in the notes are numbers. The numbers are often associated with a measurement or a date, and it is important to see that the model has caught on to some of these associations. Examples being multiple places in the predictions they predict a number, then "year" or "y/o" then a gender for the patient. Also phrases like "REASON FOR THIS EXAMINATION" shows a bigger understanding of how the notes have been written. Both these model seem to be decent enough to be used in the classifier

Text	60000 model	30000 model
TITLE:\n Cardiology fellow CCU admit note 65 yo man with idiopathic dilated cardiomyopathy (EF 35%, likely at least partially ischemic), CAD s/p LAD stent, HL presented for elective ICD implantation for primary prevention of SCD. \n Had likely profound vagal episode during establishment of venous access requiring brief period of chest compressions due to asystole.\n Temporary wire placed femorally, intubated. -\n wean from ventilator - \n maintain temp wire overnight	**TITLE:\n Cardiology fellow CCU admit** note Ms . \n [* * known lastname 2470 * *] \n is a 71 year- \n old female with a history of morbid obesity , DM2 , CRI in " 25 lb , LE chills " , who presents from \n [* * hospital3 svce * *] \n with c / o during neck pain radiating to back , aprox 6 days ago , relieved with q 4 / minutes . \n Symptoms improved with last BM to approx 3 days . \n The patient was symptoms relieved	**TITLE:\n Cardiology fellow CCU admit** nodule \n HISTORY OF PRESENT ILLNESS : \n This is a 53 y / o male with an extensive cardiac history , who initially had significant congestive heart failure with right ventricular dilatation , a she has had runs of an VT with atrial wire that to the \n [* * location (un) 1492 * *] \n type II year) with increased orthopnea . \n On coumadin for blood pressure and COPD , neosynephrine added to antibiotic hard 4:39 . \n AIRBAG extraction revealed symmetrical
TITLE: \n Respiratory Care:\n Rec d\n pt intubated with ett #7 taped @\n 21 lip transport from\n pacu/[**Hospital Ward Name **]\n and placed on\n [**Last Name (un) **]\n (see metavision).\n Pt was weaned to psv, notable cuff leak, then extubated without incident.\n Vent removed.\n	**TITLE: \n Respiratory Care:\n Rec d\n pt intubated** with # 8.0 ETT 8.0 , 23 23 @ lip . \n BS bilat wheezes , cont to receive mdi . \n Plan to wean as tol [* * sx " [* * 8 - 14 * *] \n " b \n Neuro- \n intact . \n FC . \n HOH during this admission . \n Decreased Perla r / t cataract surgery . \n Good movement of r leg . \n AEROSOL- \n prone . \n CV- \n SR \n 80's-100 's ,	**TITLE: \n Respiratory Care:\n Rec d\n pt intubated** , awake , alert , tachypnic / biting . \n r / o \n cholecystectomy with several periods of RR and restless bathroom very small @ rehab (30 minutes) . \n mdi 's given with some effect she has effect . \n pt should have worsening asthma \n once to turn when mouthing up pt is unable to understand . \n She can contact w / wife at bedside , and several role of care . \n Minimal comfort and w / neosynephrine but was not hard to communicate . \n airbag test is symmetrical
[2112-8-2**]\n 7:51 PM CHEST** (PORTABLE AP)\n Clip # [**Clip Number (Radiology) 77110**]\n Reason: S/P CVL\n _____\n [**Hospital 4**]\n MEDICAL CONDITION: 37 year old man with Crohn's disease, s/p Ex-\n Lap REASON FOR THIS EXAMINATION: S/P CVL\n _____\n FINAL REPORT\n INDICATION:\n Crohn's disease.\n Status post exploratory laparoscopy.\n Status post central venous line placement.\n Check position.\n FINDINGS:\n A single AP supine view.\n An ET tube is noted, its tip in good position just below the thoracic inlet.\n A new left subclavian central line is also present with its tip in the upper SVC, in satisfactory position.\n The heart is at the upper limits of normal.\n Some atelectatic changes are noted at the left lung base behind the heart.\n No other significant pulmonary abnormality can be identified.\n No evidence of pleural effusion is detected.\n IMPRESSION:\n 1.\n Satisfactory placement of ET tube and left subclavian central line.\n 2.\n Some basilar atelectasis on the left side.\n	**[**2112-8-2**]\n 7:51 PM CHEST**(portable AP) ; −77 BY DIFFERENT PHYSICIAN \n [* * name 7 - 2 * *] \n _ # [* * clip Number (radiology) 25 888 *** ***]]] \n _\n _\n _ Reason : strength in l chest \n _ Admitting Diagnosis : INFECTED ULCER during OR \n _ 555 ___ \n \n \n [[[*** *** hospitalhospitalhospital 222 *** ***]]] \n \n \n MEDICAL CONDITION : 57 year old man with osteomyelitis , s \n _ / p debridement REASON FOR	**[**2112-8-2**]\n 7:51 PM CHEST** (portable AP) \n Clip # [* * clip Number (radiology) 12 666 414141 *** ***]]] \n _ \n _ \n _ Reason : Assess progression of pulmonary edema \n _ Admitting Diagnosis : LIVER FAILURE \n _ 555 ___ \n \n \n [[[*** *** hospitalhospitalhospital 222 *** ***]]] \n \n \n MEDICAL CONDITION : 77 year old man \n s / p resp arrest REASON FOR THIS EXAMINATION : \n w / neosynephrine \n _ 555 ___ \n \n \n FINAL REPORT \n _ INDICATION \n

Table 5.3: Result from predictions. Text given to the models are marked in bold and with the color red. The models predicted 100 succeeding tokens. Newlines are depicted as "\n".

and the 60,000 model has a slight edge purely on it having better scores in the measurements (Table 5.2). We will proceed with creating a model using PubMed Medline, and see if that is an even better fit for the classifier than the 60,000 model.

5.4.2 Training a langauge model from PubMed Medline

Whilst starting the training with a pre-trained model on Wikipedia is a decent base for the model, using a pre-trained model from within the medical field could yield better results. The Wikipedia model continue to be the starting point of the training, but a intermediate data set in Medline will be added as a step in between the wikipedia model and the finished model, as shown in Figure 5.5.

Figure 5.5: Transfer learning through Medline to further improve the language model towards the medical field.

MEDLINE is the National Library of Medicine's premier bibliographic database that contains more than 27 million references to journal articles in life sciences with a concentration on biomedicine (National Library of Medicine, n. d.). It is formatted with XML and therefore some pre-processing has to be done. The archive contains 1065 files, each then containing 30,000 articles. It does not contain the whole article for each entry, but it contains the title and abstract.

We start by trying to train the model on all the text from title and abstracts found in the articles. This proved to be difficult with the technology we had available, as we could only get it down to 200 hours of run time. Trial and error led us to use 50 random files for this test, just so the experiment could be done in the span of time and on the technology that we had available. Each file contains 30,000 titles and a little fewer abstracts as not all the articles are stored with an abstract. This results in around 1,500,000 abstract and titles, and counting the words it ends up with 185 million words. Which is almost twice the number of words used in the original creation of the Wikipedia model (The Wikipedia model is trained on 103 million words). This could lead to some error in this experiment and will be further discussed in the end of this section.

Train loss	Validation loss	Accuracy	Perplexity
Best result from 60,000 model			
1.843	1.708	0.645	5.522
Best result from medline model			
1.768	1.646	0.643	5.189

Table 5.4: Best results from the medline and 60,00 mimic models with same train and validation sets.

Discussed later on in Section 8.1.1, the 60,000 model in Table 5.2 are not trained on the same validation set as the 30,000 model in Table 5.1. Therefore the 60,000 model presented in Table 5.4 have a slightly different result as this one is trained on the same validation set as all the other models. So these experiments are slightly more accurate than the ones presented in Section 5.4.1. The difference in the two models seem minimal (Table 5.4); the train and valid loss is about 3% better for the Medline model and the perplexity is around 6% better than the 60,000 model. This is not a large, but perhaps with more training it would be more significant. Note that these numbers can be better than at first glance, as we do not know how low the numbers can go in this case. If it is not possible to get a perplexity lower than 5, then this difference is huge, but if it is possible to achieve a perplexity of 2, the change is not that big. Another reason for this result might be that since we only used 50 of the files from Medline database, we could have been unlucky with the 50 random files, not getting any useful information from them. Unfortunately this is only speculation. With the amount of text used for training this should be significant enough to give an idea of whether or not training with Medline will yield a better result. The result give the impression that the model will be better and by using the whole lot of 1065 files it would probably be even better than the 60,000 model. The facts of the matter is that these results are going to be a little speculative, but seeing an improvement is what was to be expected.

first text	second text	third text
TITLE : Cardiology fellow CCU admit note Pt had \n [* * age over 90 * *] f with hypotension to suggest 's pneumonia due to rapidly , sepsis , Failure to thrive . \n On day shift she went to OSH for exploratory lap , pelvic ischemia , melanoma and bilateral pleural effusions . \n ECG notable for rate improved 70s , interval improvement . \n She underwent Monitoring 24 Hour Events : \n INVASIVE VENTILATION - \n STOP \n [* * 2100 - 9 - 19 * *]\n	**TITLE : \n** **Respiratory Care : \n** **Rec d \n** **pt intubated** for airway protection , \n Weaned vent to psv / full support , tolerating well . \n 's 18- \n 24 bpm , intubated for airway protection . \n Lung sounds coarse bilaterally . \n Failed PS due to impaired cough . \n Was weaned successfully and placed on PS 5 / 400 / improved . \n SPO2 dropped to 50 % , Comfortable . \n Stable on Fio2 100 % . \n Has brought small partially thin white secretions . \n O : Please see\n	**[* * xxunk - 8 - 2 * *] \n** **xxunk PM CHEST** (portable AP) \n Clip # [* * clip Number (radiology) nitro 's * *] \n _ _ Reason : Please evaluate for interval change in appearance of left pleural effusion or other interval improvement . \n _ _ Admitting Diagnosis : CONGESTIVE HEART \n FAILURE \n _ _ _ _ _ \n \n \n \n \n [[[[[***** ***** hospitalhospitalhospitalhospitalhospital 22222 ***** *****]]]]]\n \n \n \n \n \n MEDICAL CONDITION : 77 year old woman with PNA , tachycardia , brought in \n [* * hospital mail * *] \n HOSPITAL\n

Table 5.5: Medline predictions on the same text as in Table 5.3, predicting 100 succeeding tokens. The results shows an understanding of the structure of the MIMIC clinical notes, with slightly less predictions of special tokens.

Now lets see if we can spot Medline in the predictions the model does. The experiment in Table 5.5 are based of the same texts as in Table 5.3. The first thing we see is that thee number "2112" has been replaced with the "xxunk"

token. This is interesting because it might suggest that the model have a vocabulary that is better centered around text and doesn't have space for all the numbers that the other models have. The rest of the texts predicted seem to be more in line with what the other two models predicted, so it is difficult to spot a severe difference in using Medline as a intermediate data set.

5.5 Conclusion

The results from these experiments are a little speculative, and it is difficult to draw a good conclusion from them. The hypothesis where that the model that had been trained through Medline would perform better than those who where not, which was the case by a small margin. This result could have been better if the 1,5 million articles we used were picked out only from relevant articles, and not at random. It might also be better if the model was trained on all the 27 million articles available in the data set. Considering the results and the sources of error it was decided that these models where good enough, and that it had little purpose to try and improve them more, considering the time and resources we had available. The experiments presented in Section 6.3 gives a better understanding in how well the models can create predictions on a real problem.

6. Making predictions on clinical free-text

6.1 Training a Text Classifier

A text classifier is dependent on a language model and data in order to train. The language models were produced in the previous chapter and will be used in this experiment. Before delving into training, it is necessary to know how to measure the classifier.

6.1.1 Metrics

Similar to the language model, it is necessary to evaluate how well the classifier models perform and observe how they improve or deteriorate during training. For the following experiments, these few select metrics are chosen to best represent the models' efficacy; F_1 *score*, *accuracy*, *recall*, *precision* and *MCC*. All of which are described in detail in section 2.2.

6.1.2 Cross validation

Cross validation is a very useful technique when training models that splits the training data set into subsets, called folds. It reduces the luck (or bad luck) associated with predictions on a data set. The reason to use cross validation instead of a validation set, in this case, is to make sure it can reproduce similar results many times. The likelihood of the model's predictions not being average greatly reduces. In addition, it is much harder to overfit a model on a cross validation set because it needs to satisfy a number of folds.

6.2 Aggregating clinical notes

A clinical note can be seen as a snapshot of a subject's current situation at a particular moment in time. Since the subject's situation is likely to change, the clinical note will become outdated. Still, it contains historical information which provides important context to the patient's current situation. It is worth considering the context a model should be provided with. Should the model be given one note at a time? Or maybe all the notes for each subject? Perhaps all the notes prior to and including a note, so that the previous recordings are captured too? The latter would be most interesting in real life, where the model is fed information in real time. However, this would raise more issues such as how one can be certain the model will not be over-fit and how to label each entry.

To reiterate: the intention of these experiments are not to be as applicable to real life as possible, but rather to research if free-text data can complement fixed features. For this reason, the data can be simplified without compromising

viability by aggregating notes that account for the same instance. Determining what constitutes an instance is most appropriately done initially in each experiment. One disadvantage to this is that there will be fewer predictions and, consequently, less feedback. On the same note, the opportunity to plot the progression of predictions for the same instance over a timeline, which could reveal what notes have the greatest impact, is lost.

6.3 Experiment

These experiments serve two purposes. The first to address the research question in section 4.3; can the text classifier extract useful information from clinical medical text? If so, then the text classifier should be able to make predictions that are better than random. Second, it is a precursor for the research question in section 4.4; can the extracted information be used in combination with other data? A capable text classifier is required if the answer to the research question should be yes, because only a text classifier whose features are better than random can contribute to a greater context of information.

The experiment will consist of three parts. First, is data preparation, second is training the classifier, and finally the classifier's performance is measured on a test set.

6.3.1 In-hospital mortality

The objective of the experiment is to predict whether a subject will die either at the hospital or the intensive care unit. As previously discussed, it is necessary to determine what should constitute an instance before aggregating any notes. This particular case is a binary classification problem, which means a subject either lives or dies. It is worth noting that a subject cannot recover from death and that the subject's health at any point will either hasten or delay the time of death. Thus, it stands to reason all of a subject's notes should be taken into consideration when attempting to predict the fate of the subject.

After labeling the data with whether the subject dies in the hospital or the intensive care unit, it is time to inspect the class distribution. The distribution is depicted in figure 6.1 There are 31,159 negative cases and 3,313 positive cases, which means the mortality rate is 9.6%, rounded off to the nearest decimal. It is important to keep the same ratio between the classes when training the classifier, so the classifier has a precise impression of how skewed the data really is. This is done by using stratified sampling. The data is split into two strata, where one contains one fifth of the data and is reserved for testing and the other contains the rest and is meant for training. Stratified K-fold from Sklearn was used to create ten cross-validation folds while preserving the mortality rate of the training set. Afterwards, the model was trained on the whole training set.

There are four text classifiers, one for each of the language models in section 5.4. Thus, the text classifiers are characterized by their underlying language model and the text classifiers will be referred to in the following order:

1. The text classifier that uses the language model with a vocabulary size of 60,000 without a fixed random state seed.

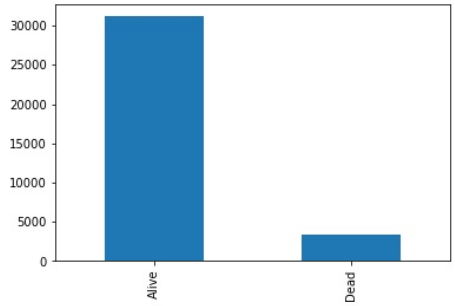


Figure 6.1: Mortality rate class distribution

2. The text classifier that uses the language model with a vocabulary size of 30,000 without a fixed random state seed.

3. The text classifier that uses the language model with a vocabulary size of 60,000, trained on the Medline data set with a fixed random state seed.

4. The text classifier that uses the language model with a vocabulary size of 60,000 with a fixed random state seed.

The first two models are best compared to each other because they are both trained on a random selection of the data. The remaining two are trained on the same selection of data, which means the results will be reproducible and making a comparison will be less speculative.

6.3.2 Results

Unfortunately, precision, recall and MCC are not listed in the result tables for all the folds. The F_1-score is listed, but it still does not provide a full picture. However, the F_1-score is sufficient in our experiment to determine whether a model is better than random.

By applying *gradual unfreezing*, described in section 1.5.3, the text classifier only trained the last layer for the first two epochs, then the two last layers, and finally, every layer was unfrozen and mutated during training. A suggested learning rate was found using Fastai's lr_find tool (Howard and Gugger, 2020), which was based on the work of (Smith, 2017). The second text classifier trained through a total of five epochs with fixed and steadily decreasing learning rates, but without gradual unfreezing. This may have impacted the model's performance negatively, since it no longer found a tailored learning rate. The third classifier was trained exactly the same as the second text classifier. The fourth classifier used the already mentioned lr_find tool to find the steepest point, where the error rate decreased the most. It then applied gradual unfreezing over a total of five epochs. Next, the four text classifiers were evaluated on a separate test set. Since they all have been trained through ten cross validation sets, they are expected to perform somewhat similarly on the test set. It seems the fourth text classifier performs much worse than the other text classifiers. This is likely due to a bug and not any kind of indication of the potential of the method. The other three text classifiers performed in somewhat the same proximity as they did while training. As seen in Table 6.5, the first text classifier

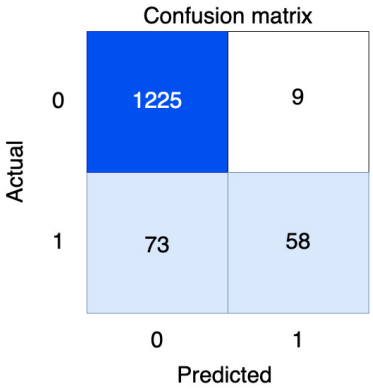
Fold	Train loss	Validation loss	Accuracy	F_1 Score
1	0.245	0.223	0.937	0.531
2	0.258	0.165	0.940	0.613
3	0.240	0.171	0.938	0.556
4	0.268	0.181	0.939	0.565
5	0.243	0.180	0.936	0.542
6	0.203	0.172	0.944	0.615
7	0.242	0.164	0.944	0.607
8	0.233	0.185	0.933	0.520
9	0.235	0.160	0.936	0.582
10	0.231	0.171	0.944	0.601
Mean	0.240	0.177	0.939	0.573
Std	0.016	0.017	0.004	0.034

Table 6.1: Text classifier using language model with a vocabulary of 60,000 words without a fixed random state seed. This is the first text classifier in the list of text classifier versions in section 6.3.1.

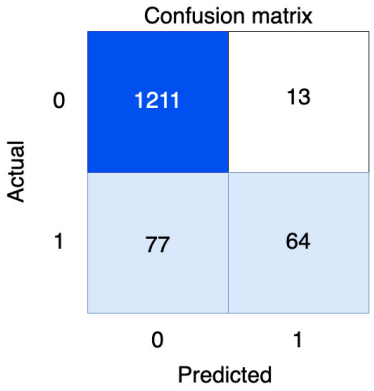
Fold	Train loss	Validation loss	Accuracy	F_1 Score
1	0.239	0.198	0.930	0.495
2	0.255	0.186	0.930	0.528
3	0.254	0.187	0.935	0.539
4	0.272	0.180	0.936	0.536
5	0.251	0.207	0.925	0.510
6	0.254	0.200	0.934	0.540
7	0.238	0.191	0.933	0.520
8	0.230	0.201	0.922	0.456
9	0.246	0.191	0.931	0.529
10	0.226	0.189	0.935	0.508
Mean	0.247	0.193	0.931	0.516
Std	0.013	0.008	0.004	0.024

Table 6.2: Text classifier using language model with a vocabulary of 30,000 words without a random state seed. This is the second text classifier in the list of text classifier versions in section 6.3.1.

has an F_1-score about just as good as the second text classifier, however, their Matthew correlation coefficient varies. This is verified by the confusion matrices. As depicted in Figure 6.2a, the first classifier is more careful in predicting positive labels. This causes it to predict more negatives, and since the data set contains more negative labels, they are more likely to be true negatives. In effect, there are less incorrect predictions. In Figure **??**, there are more true positives, but at the cost of less true negatives and more false predictions. This means the first text classifier will have a higher precision at the cost of a lower recall, because it was more careful than the second text classifier.

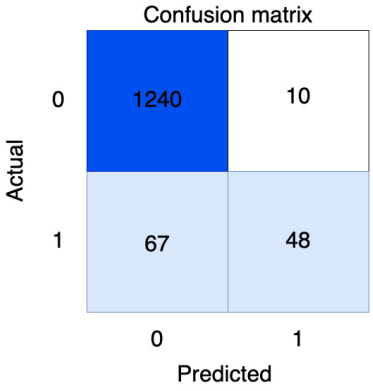


(a) The confusion matrix for the first text classifier's predictions on the test set. The first text classifier is described in more detail in the list in Section 6.3.1. Its metrics are shown in the first row in Table 6.5.

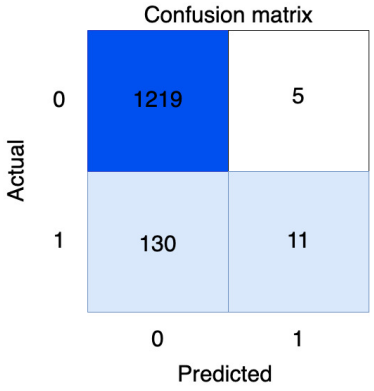


(b) The confusion matrix for the second text classifier's predictions on the test set. The second text classifier is described in more detail in the list in Section 6.3.1. Its metrics are shown in the second row in Table 6.5.

Figure 6.2: Confusion matrices for the text classifiers without a random seed after evaluation of the test data set. The test data set accounts for 20% of the whole data set, and is split using stratified sampling to ensure the same proportion of classes.



(a) The confusion matrix for the third text classifier's predictions on the test set. The third text classifier is described in more detail in the list in Section 6.3.1. Its metrics are shown in the third row in Table 6.5.



(b) The confusion matrix for the fourth text classifier's predictions on the test set. The fourth text classifier is described in more detail in the list in Section 6.3.1. Its metrics are shown in the second row in Table 6.5.

Figure 6.3: Confusion matrices for the text classifiers without a random seed after evaluation of the test data set. The test data set accounts for 20% of the whole data set, and is split using stratified sampling to ensure the same proportion of classes.

Fold	Train loss	Validation loss	Accuracy	F_1 Score
1	0.247	0.190	0.933	0.483
2	0.251	0.168	0.939	0.539
3	0.253	0.183	0.934	0.496
4	0.261	0.171	0.935	0.525
5	0.276	0.190	0.933	0.487
6	0.272	0.184	0.936	0.533
7	0.245	0.183	0.936	0.522
8	0.286	0.207	0.926	0.446
9	0.290	0.198	0.929	0.514
10	0.304	0.207	0.933	0.533
Mean	0.269	0.188	0.935	0.520
Std	0.019	0.013	0.003	0.028

Table 6.3: Text classifier using language model trained on Medline with a vocabulary of 60,000 words and a seed for random state

Fold	Train loss	Validation loss	Accuracy	F_1 Score
1	0.296	0.219	0.921	0.503
2	0.274	0.169	0.939	0.539
3	0.296	0.196	0.931	0.510
4	0.290	0.190	0.936	0.564
5	0.250	0.188	0.934	0.507
6	0.259	0.174	0.941	0.603
7	0.247	0.181	0.937	0.517
8	0.249	0.187	0.929	0.447
9	0.280	0.179	0.935	0.504
10	0.251	0.179	0.937	0.500
Mean	0.269	0.186	0.934	0.519
Std	0.019	0.013	0.005	0.040

Table 6.4: Text classifier using language model with a vocabulary of 60,000 words and a seed for random state

6.4 Conclusion

The second text classifier ever so slightly beats the first classifier with a higher F_1-score, as seen in Table 6.5. However, the first text classifier seems like the superior text classifier when considering the Matthew correlation coefficient. This means in respect to both the positive and negative cases with regards to their disproportional class distribution, the first text classifier outperforms the second text classifier.

Unfortunately, there was some trouble when evaluating the fourth text classifier on the test data set, as seen in Table 6.5. The low scores, that are barely better than random, suggests that the text classifier was not trained with the provided encoder. However, by considering the cross validation mean scores of the third text classifier and the fourth text classifier, as seen in Tables 6.3 and

Model	Accuracy	F_1-score	Recall	Precision	MCC
1	0.940	0.586	0.443	0.866	0.594
2	0.934	0.587	0.454	0.831	0.585
3	0.944	0.555	0.417	0.828	0.564
4	0.901	0.140	0.078	0.688	0.209

Table 6.5: Results of the text classifiers evaluation on the test set. The first text classifier refers to the one based on a language model with a vocabulary size of 60,000 trained through 15 epochs. The second refers to the text classifier is based on a language model with a vocabulary size of 30,000 trained though 21 epochs. The third and fourth text classifiers are based on language models with a vocabulary size of 60,000 trained through 17 and 10 epochs respectively. Additionally, the third text classifier is trained on an intermediate data set, Medline, before being trained on MIMIC-III. There seems to be something wrong with the fourth text classifier because it was trained very similar to the other text classifiers, but performed much worse

6.4 respectively, there is little difference between them. This is not unexpected because of the similar performances of the underlying language models. However, the third text classifier seemed more consistent in its performance. Its median F_1 score is greater than the fourth model (0.522 versus 0.510). This also means the fourth text classifier has a greater variance; the max F_1 score is 0.603 compared to the third model's 0.539.

7. Stacking free-text features with heterogeneous data

Stacking is a technique used to combine models, where activations in a model is extracted and used in another model. This can be useful if, for example, the objective is to use unstructured data alongside structured data. The unstructured data could first be handled by a model dedicated for that specific task, and then the output could be fed into another model that only expects structured data. In addition, this can be used as an ensemble method, where a model accepts the output of at least two other models as input.

The objective for the following experiments is to convert free text data to numerical values. This is done by extracting the activations of the penultimate layer and complementing it with features found in structured data sets. Unless the additional data serves as mostly noise, it should help to improve the performance of the model since it only has more data. It will also be interesting to see how much the text features attribute to the final prediction compared to the structured data. This can be explored even further by looking at the feature importance of every feature, including the text activations.

7.1 Extracting activations from the text classifier

Fastai provides useful tools for retrieving information from the model between the moment of input and output. They are called 'Hooks', and they are callbacks that are hooked onto specified layers in the model. When the layer is invoked with some input and has produced some output, the hook is invoked with three arguments: the layer that is being invoked, the input and the output. The input is the activations.

The last layer in the text classifier is a linear transformation that accepts 50 features and outputs a binary prediction. By hooking onto this layer it is possible to save the activations every time the layer is invoked. The activations need to be saved alongside their respecting subject ID, which can be difficult since the subject ID is not available to the layer. However, by using the model to evaluate the notes for a single subject at a time, the subject ID is known and the hook can mutate a global object, which then saves the text activations.

7.2 Aggregating heterogeneous clinical data

There is a practical limitation to the number of features a model can utilize. If this limit is exceeded, the model may not find the optimal weights and prioritize the "wrong" features. Consequently, the model's performance is unlikely to increase. In fact, it may very well decline. There is a number of factors that play a role in what this limit is, such as the classifier and the feature-label

distribution (Hua et al., 2004). In practice, this restricts us from using every available feature.

The collected features are vitality measurements and information about the subject on admission, such as the gender, ethnicity, age and diagnosis on admission. This data set is then exported to a binary pickle file.

7.3 Combining heterogeneous clinical data and free-text features

As mentioned in the previous section, too many features can hurt the model's efficiency rather than boost it. If this is the case, the free-text features might not be used to an optimal extent. This can be monitored using feature importance.

All the extracted text activations are stored in a binary python file as a Data Frame, which make them easily accessible from anywhere. The same holds for the heterogeneous clinical data. Both data sets have the same set of subject IDs as indices, which makes it very easy to combine them using Pandas functionality.

7.4 Finding a suitable classifier

Finding a suitable classifier can be tedious work because there are many models to compare and there are many different options that can be tuned for the best performance. These problems are not inherently interesting to the experiments, and can be abstracted away using a library called PyCaret (Ali, 2020). PyCaret is a low-code library on top of Sklearn that greatly simplifies this process. In effect, finding five suitable classifiers can be done in roughly two lines of code. In addition, these can be effortlessly fine-tuned and explored.

7.5 Experiments

From the conclusion in Chapter 6.4, the text classifiers were significantly better than models that produce a single prediction or random predictions, even with respect to skewness. The next step is to investigate the utility of the free-text activations from a text classifier which will be done in three ways.

The first is to carry out the experiment with and without free-text features and compare the performance of the models. The top three models for each data set should then be optimized. The idea is to minimize what is left to chance by maximizing the performance of the models, ideally to a point where they stagnate. Any differences in performance then will be significant, even if they are small. Unfortunately, the models are not likely to reach their highest potential because it would be too challenging, however, they will be tuned. After being tuned, the best model trained on each of the data sets will be compared to each other.

Second, carrying out the experiments with the worst and the best performing text classifiers from the previous experiment will indicate the relative performance impact of different text classifiers. The best performing text classifier is the one with a language model that has a vocabulary of 60,000 and trained through 15 epochs. The worst performing text classifier was the text classifier

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.952	0.675	0.542	0.867	0.675
ada	0.948	0.664	0.564	0.807	0.649
ridge	0.944	0.589	0.438	0.903	0.606
et	0.940	0.531	0.371	0.937	0.569
rf	0.937	0.479	0.320	0.957	0.533

Table 7.1: The top five models' mean metric scores without free-text features for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. ET is short for Extra Trees Classifier, and RF is the Random Forest Classifier.

whose language model only had a vocabulary size of 30,000 and trained through 21 epochs.

Finally, how much a model uses the free-text features can be found by using feature importance. This can reveal how much the classifier is dependent on the free-text features.

Given the first and second point, the first experiment will be to find a model without text-features and then with free-text features. The heterogeneous data should be combined with the free-text features two times; one for each of text classifiers' activations. Finally, the feature importance will be discussed for each model.

7.5.1 In-hospital mortality without free-text features

For this experiment, only a subject's vitality data were present in the data set. Table 7.1 displays the training results for classification without any free-text features. The gradient booster classifier beat the other models, followed by the Ada Boost Classifier. The ridge classifier had a slightly worse F_1 score than the best text classifier, however the top two models performed better than the text classifier did.

For the next step, Gradient Boosting classifier, Ada Boost classifier and Ridge classifier was tuned to optimize the F_1-score across all ten folds in the cross validation training set. After their optimal hyper parameters had been found, the models were finalized by being trained on the whole training set.

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.952 $\pm$0.004	0.688 $\pm$0.026	0.580 $\pm$0.031	0.847 $\pm$0.030	0.677 $\pm$0.027
ada	0.948 $\pm$0.003	0.614 $\pm$0.032	0.459 $\pm$0.035	0.930 $\pm$0.023	0.632 $\pm$0.027
ridge	0.945 $\pm$0.003	0.593 $\pm$0.032	0.439 $\pm$0.033	0.917 $\pm$0.019	0.613 $\pm$0.027

Table 7.2: The top three tuned models' mean metric scores without free-text features for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier.

Surprisingly, the Ada boost classifier performed worse after being tuned in Table 7.2 compared to the base classifiers in Table 7.1. This indicates that it

is hard to optimize this model and if its base version was close to its optimal, it still does not outperform the gradient boosting classifier. After having tuned the gradient boosting classifier, its most important features can reveal what it considers to achieve its result. As seen in Figure 7.1, it heavily prioritizes cmo_1.0, and the second place scores half the variable importance.

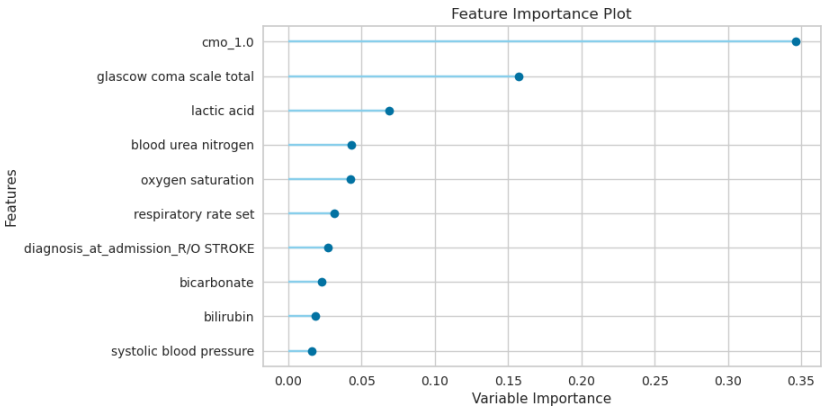


Figure 7.1: The most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set does not contain any extracted text features.

Finally, the gradient boosting classifier was evaluated on the test set and can be viewed in Figure 7.4a.

7.5.2 In-hospital mortality with free-text features

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.957	0.734	0.622	0.868	0.713
ada	0.953	0.710	0.636	0.807	0.691
ridge	0.952	0.666	0.525	0.913	0.671
rf	0.951	0.653	0.506	0.925	0.663
et	0.951	0.649	0.499	0.930	0.660

Table 7.3: The top five model mean metric scores with free-text features for ten stratified sample folds. The free-text features are extracted from a text classifier built on a language model with a vocabulary size of 60,000 and without a seed, which performed the best out of the trained text classifiers. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. ET is short for Extra Trees Classifier, and RF is the Random Forest Classifier.

Table 7.3 displays the results for classifiers using free-text features outputted from the best text classifier. Every model performs better than the best text classifier alone. Also, every model has a relatively high F_1-score. This may indicate that a free-text yielded a high correlation with the label to predict.

The top two models outperform all the models trained without free-text features. The top two models score better in both recall and precision compared to the top two classifiers without free-text features, which suggests they are all-over better classifiers. Ridge, random forest and extra trees classifiers score very high in precision and low in recall. A recall of 0.5 means only half of all actual positives are true positives, while the other half is false negative. It is probable that the models are a bit too conservative, and perhaps a lower threshold for predicting positively would increase their score. The top three classifiers in table 7.3, gradient boosting classifier, Ada boost classifiers and ridge classifer, are then tuned to optimize the F_1-score across the ten stratified folds.

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.958 ±0.003	0.740 ±0.017	0.665 ±0.023	0.835 ±0.021	0.723 ±0.018
ada	0.955 ±0.003	0.700 ±0.021	0.579 ±0.028	0.886 ±0.026	0.695 ±0.021
ridge	0.951 ±0.003	0.651 ±0.025	0.501 ±0.028	0.930 ±0.017	0.662 ±0.023

Table 7.4: The top three tuned models' mean metric scores with the free-text features extracted from the best performing text classifier for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier.

The Ada boost classifier and the ridge classifier in Table 7.4 seems to perform slightly worse than their base counterparts in Table 7.3. The gradient boosting classifier has improved slightly and is ahead of the other classifiers with some margin. Since it has the best F_1-score, it will be trained on the rest of the data set with the hyper parameters found by PyCaret.

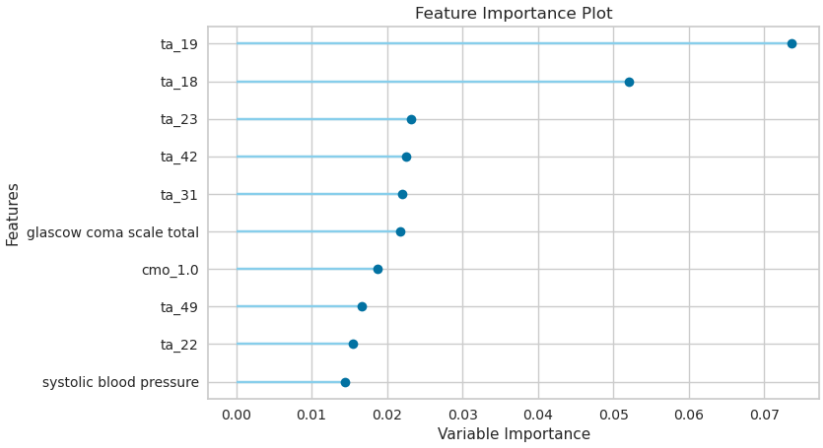


Figure 7.2: The most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set contains text features extracted from the best performing text classifier.

Its most important features, as seen in Figure 7.2 are overwhelmingly text features. It seems not to consider cmo_0.1 much, unlike the gradient boosting classifier that did not have text features, as seen in Figure 7.1. Glascow coma scale total is, however, important to both of the classifiers.

As seen in in Figure 7.4b, the gradient boosting classifier scored an F_1-score of 0.739, which is very close to its performance on the cross validation set. It significantly outperforms the gradient boosting classifier without text features which can be seen in Figure 7.4a.

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.954	0.702	0.584	0.882	0.696
ada	0.951	0.703	0.628	0.800	0.683
ridge	0.948	0.632	0.486	0.904	0.641
rf	0.945	0.593	0.434	0.944	0.618
dt	0.927	0.598	0.588	0.609	0.558

Table 7.5: The top five model mean metric scores with free-text features for ten stratified sample folds. The free-text features were extracted from a text classifier built on a language model with a vocabulary size of 30,000, which performed the best worst out of the trained text classifiers. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier. DT is short for Decision Tree Classifier, and RF is the Random Forest Classifier.

The final data set to evaluate models on, is the data set with vitality measurements and text features extracted from the worst performing text classifier. The base models' results are shown in Table 7.5. As expected, the classifiers here perform slightly worse than the classifiers trained on the data set with free-text features extracted from the best text classifier, seen in Table 7.3. In addition, the top two classifiers with the text features extracted from the worst text classifier outperform the top three in table 7.1.

Model	Accuracy	F_1	Recall	Precision	MCC
gbc	0.957 $\pm$0.002	0.728 $\pm$0.015	0.633 $\pm$0.018	0.857 $\pm$0.026	0.715 $\pm$0.017
ada	0.952 $\pm$0.003	0.664 $\pm$0.025	0.524 $\pm$0.028	0.908 $\pm$0.022	0.668 $\pm$0.023
ridge	0.951 $\pm$0.003	0.651 $\pm$0.025	0.501 $\pm$0.028	0.930 $\pm$0.017	0.662 $\pm$0.023

Table 7.6: The top three tuned models' mean metric scores with the free-text features extracted from the worst performing text classifier for ten stratified sample folds. GBC is short for Gradient Boosting Classifier. Ada is the Ada Boost Classifier. Ridge is the Ridge Classifier.

After fine-tuning the top three classifiers, gradient boosting classifier, Ada boost classifier and ridge classifier, the Ada boost and ridge classifier seems to deteriorate unlike gradient boosting classifier, which improves considerably, as seen in Table 7.6. As seen in Figure 7.3, the most important feature is glascow coma scale total, which also appears in Figure 7.1 and Figure 7.2. There are

also many text features present.

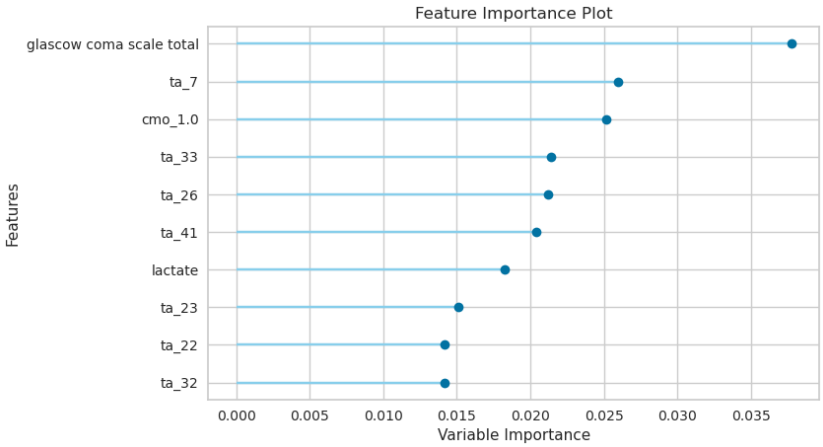
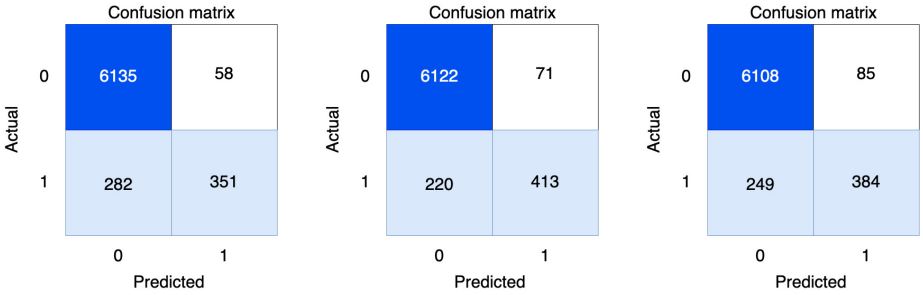


Figure 7.3: A graphical representation of the most impactful features of the fine-tuned gradient boosting classifier after being trained on the training set using ten stratified sampled folds of cross validation. The training set contains text features extracted from the worst performing text classifier.

Since the gradient boosting classifier outperformed the two others in Table 7.6, it was trained on the rest of the training set and evaluated on the test data set. The confusion matrix in Figure 7.4c shows the results. The model scored an F_1-score of 0.697, which is worse than the classifier in Figure 7.4b with the text features extracted from the best text classifier. Additionally, it performs slightly better than the classifier without text features in Figure 7.4a.

Classifier	Accuracy	F_1	Recall	Precision	MCC
1	0.957	0.739	0.652	0.853	0.724
2	0.951	0.697	0.607	0.819	0.680
3	0.950	0.674	0.555	0.858	0.666

Table 7.7: The results for each of the best classifiers evaluated on the test set, all of which are gradient boosting classifiers. The first classifier is trained on a data set with text features extracted from the best performing text classifier. The second classifier is trained on a data set with text features extracted from the worst performing text classifier. The third classifier is trained on a data set without text features.



(a) A depiction of the confusion matrix for the tuned gradient boost classifier's predictions on the test data set without text features.

(b) A depiction of the confusion matrix for the tuned gradient boost classifier's predictions on the test data set with text features extracted from the best text classifier.

(c) The confusion matrix for the tuned gradient boost classifier's predictions on the test data set with text features extracted from the worst text classifier.

Figure 7.4: The confusion matrices for the best classifiers evaluated on the test data sets.

7.6 Conclusion

Based on the evaluation results on the test set, it is clear that the classifier without text features, seen in Figure 7.4a, is worse than the classifiers with text features in Figures 7.4b and 7.4c. Furthermore, the classifier trained on the data set with text features extracted from the best text classifier performed considerably better than the classifier trained on a data set with text features extracted from the worst text classifier.

To reiterate, the research question in focus is from Section 4.4 and it poses the topic "can the extracted information be used in combination with other data?". The conclusion is yes; all evidence points towards a performance boost with text features. The base models without text features had a strong tendency to be better than their counterparts without text features, and the classifiers that were evaluated on the test set stayed true to this tendency. Additionally, it seems as if the better the text features, the better the classifier. This is also confirmed by looking at which features are the most important to the classifiers.

Part V

Conclusive work

8. Discussion

Before conducting our experiments we could only presume what the results would be. But considering the rapid growth and many successes of machine learning in the recent years, we had high hopes. We assumed that we would be able to create a language model for clinical notes with about 30% accuracy, and classification models for useful tasks that could score well above the chance level. The goal was to uncover indications that this approach could help medical staff foresee some events, improving both the accuracy and speed of their decision-making.

In this chapter we summarize and discuss our results.

8.1 Language models on free-text

Creating a language model and using it in the real medical world is a challenge, since the free-text journals have a specific subset of words used in them. They may contain many Latin phrases and abbreviated words. This leads back to the first research question, see Section 4.2, can the a language model pick up on these linguistic features. The results from the experiments in Section 5.4 indicate that the models created all picked up on some linguistic features during their training. The models stopped writing sentences and started to construct notes, they started predicting with abbreviation and and line breaks signaling a new informative entry in that note. We ended up creating four different language models for testing some of the larger changes and their affect on the predictions. They were compared in groups of two, a model with 60,000 vocabulary and one with 30,000 trained directly from Wikipedia to MIMIC. The other comparison being between one 60,000 and a model trained through an intermediate data set, Medline, to try and fine-tune the language better towards the medical field.

We chose to use ULMFiT when developing this language model, which had the best models only a few years ago. However, recently the transformer based models have been achieving the state-of-the-art results when tackling NLP problems. i.e. hugging face Inc. (Hugging Face, n. d.) where the AI community, including big companies as Microsoft, post some of their models. The main reason to use ULMFiT was to use the fastai library, which we find to be easy to use and set up. We considered using the BERT model (Devlin et al., 2018) but found that model needing more time and computing power to train. Also it was achieving some of the best results when we started planning for this project.

One problem with such notes is that there could be a substantial difference between how two doctors write these notes. As discussed earlier the results from the language models are a bit speculative as they are complex and difficult to understand why the prediction has been made. But looking purely at the metric the results, especially in accuracy, exceeded our expectations. The best language model obtained an accuracy in the 60% range (Table 5.1). There can be several reasons for this, one of the reasons being that the vocabulary used by

doctors are smaller than that of the whole English language and therefore the model does not run into words it does not recognize as often. Another reason is if the model predicts the same token a lot of the time, and that has a high representation in the set given, it can score high in accuracy. In our case that was line breaks. Even though it should predict them in some cases as it signals new information, it should not predict it every other word. Whilst predicting it often gives a good result, that would overshadow the other tokens so the model would not have been understanding the language as well as intended.

The model trained through Medline had several problems, most revolving around the technology we had available. We ended up using only 50 of the 1065 files available, in order to save training time and storage capacity. The number of words were still around 185 million, which we assumed would be more than enough. The result from this experiment was a little underwhelming, we saw a between 3% to 6% change for the better in the metrics (Table 5.5). we can not say for certain if this is a big change, if the perplexity cannot get lower than 5 the impact is of higher significance than if perplexity can be as low as 2. We assume it can get better than 5, and we also saw a 4.8 in the first 60,000 model (Table 5.2). Therefore we expected these results to be better than they were, as the model would be fine-tuned towards the medical field. One reason for this could be that the 50 files chosen at random were "bad" files and did not contain a better language nor a more specific one. This leads to our next point, 185 million words may be too many, and the model do not get more tuned to the medical field from it. While the results was not as good as hoped, the results were still better than that of the other models when comparing to the same validation set.

Overall the creation of the language model was a time consuming process, and there are probably some modifications that could be done to improve that time, and also to improve the models further. However, we feel that the model worked well and gained good results with the time used.

8.1.1 Sources of error

The first language model trained on 60,000 presented in Table 5.2 has a different split in the train/validation set than the other 3 models created. This means that the results gotten from comparing 30,000 and 60,000 are not has good as they could be. This was unfortunately discovered too late and corrected in the later experiments.

8.2 Text classifier on clinical notes

To summarize and expand upon Section 6.1.1, the metrics chosen for the experiments in Section 6.3 are accuracy, F_1-score, precision, recall and MCC. These metrics are commonly used to interpret the performance of a model. The precision and recall can give away some interesting insights to how the text classifier could improve. For example, a low recall and a high precision means the model may have been too conservative, and perhaps lowering its threshold for predicting a positive label would increase its overall performance. The F_1-score is the mean of the precision and recall, and it reveals the model's performance with

respect to the positive label. While the positive label may be of most interest, it is still worth considering the overall performance of the model too, using MCC. MCC measures the model's performance with respect to the disproportionate class distribution. The text classifiers was measured across ten stratified sampled folds, to ensure there was no lucky or unlucky fluke. In addition, it prevents over-fitting because the text classifier is measured across ten smaller subsets of the data set, which means any volatility in the performance is likely to be uncovered by inspecting standard deviation.

A defining part of the experiments in Section 6.3 and also later in Section 7.5 was how to structure the data set. As discussed in Section 6.3.1, a clinical note describes a subject's situation at a particular moment. Thus, a string of clinical notes tells a story and provides context to a subject's well-being. This is one way to structure the data set; by concatenating all the clinical notes and assign them to a single label. The benefit of this approach is the simplicity of it. There is no need to consider a time frame, which may complicate the code. There is no need to consider when a subject should be considered to be dead, since all the notes are put together as one. It is also sufficient to answer the research question stated in Section 4.3; can a text classifier built from the language model extract useful information from clinical notes? The downside of joining all the notes and predicting once per subject is that the final notes may be very revealing, perhaps to the point were the fate of a subject is literally spelled out. This could cause the text classifier to, in effect, have the label as a feature which would render the experiments much less useful. Though this may be the case for some subjects, it is unlikely to be the case for all subjects because the models would then score much higher on the test set than they do in Table 6.5. This issue also distances the experiment from real life, because it would be much more useful to evaluate subjects continuously rather than after all the notes had been written. Another limitation of this approach is the number of predictions will greatly decrease, which means the number of entries in the data set decreases. Effectively, the text classifier will not be able to adjust it weights as many times.

Regardless of the structure of the data set, MIMIC-III is not aware of the future for the living subjects it contains. This means some predictions may have been false positives, because the subject has not passed on yet. This issue cannot be avoided or improved, because the data set must be immutable for two reasons. First, if the data set was updated continuously, it could violate a person's privacy. Secondly, the data set must remain unchanged for it to be a common ground for research and experiments, which is its purpose. Nevertheless, it should not unfairly affect the results, in fact, it is very improbable that it will damage the integrity of the conclusions.

As a final note on the data set, is has a very disproportionate class distribution, as seen in Figure 6.1. In order to take this into account, stratified sampling was used when generating ten folds for the cross validation. This way, the model will have a realistic impression of the skewed data set.

For the experiments, a text classifier was trained for each of the four language models in Section 5.4. To mirror what is stated in Section 6.3.1, the text classifiers are characterized by their underlying language model and the text

classifiers will be referred to in the following order:

1. The text classifier that uses the language model with a vocabulary size of 60,000 without a fixed random state seed.

2. The text classifier that uses the language model with a vocabulary size of 30,000 without a fixed random state seed.

3. The text classifier that uses the language model with a vocabulary size of 60,000, trained on the Medline data set with a fixed random state seed.

4. The text classifier that uses the language model with a vocabulary size of 60,000 with a fixed random state seed.

Text classifiers three and four are based on language models that share the same random seed, which makes it more natural to compare them to each other. The first and second text classifiers' language models do not have a specified random seeds.

The first text classifier was trained differently from the three other text classifiers. They all were trained using gradual unfreezing over the same amount of epochs, and their learning rates were found using tooling provided by Fastai. The first classifier, unlike the other three, was trained with a learning rate that started from the steepest point for the loss function to the minimum point for the loss function. This was changed for the other models to experiment different methods. First, an attempt was made to write the learning rates manually, which did not provide great results. The final revision to their training was to use the lr_find function from Fastai to find a suggested learning rate. This time, a new learning rate was found very often and it trained only for one epoch. It The training results for the three similarly trained text classifiers were oddly similar, while the first text classifier performed substantially better. However, on the test set, the best and the worst performing classifiers were extremely similar.

Additionally, Fastai's function, called lr_find, is useful and convenient, it is only a suggestion. It finds a suggestion by measuring the loss for a few batches of data, however, the loss may vary greatly between batches. There is no guarantee that it will find the optimal learning rate. Still, it proved to be much more helpful than the learning rates manually provided.

Finally, an unfortunate blunder was discovered late in the project's life cycle. Fastai reserves a part of the training set to be set aside and used for a validation set after training. This means that the text classifiers were only trained on 70% of the data available for training. Fortunately, this affected all the text classifiers equally and thus does not invalidate the conclusion.

After training and being evaluated using cross validation, the first text classifier seemed superior to the others. However, after being evaluated on the test set, the text classifiers performed very similarly, as seen in Table 6.5. Unfortunately, the fourth text classifier did not do well. It is very likely this is due to a bug and is not a fair assessment of the model's performance. Despite this trouble, there is still value in comparing it to the third text classifier, because they share the same random seed. Of course, they have to be compared on the grounds of the cross validation results, found in Tables 6.3 and 6.4.

As concluded in Section 6.4, the first and second model are very close in performance with respect to the positive label. However, since they are predicating on an imbalanced data set, the MCC can be more appropriate metric when considering what text classifier is the overall best one.

To acquire a decent impression of how proficient these text classifiers are with respect to the positive label, it helps to consider the performance of a model that produces "random" predictions with the same proportions of classes as in the data set. As mentioned in Section 6.3.1, the mortality rate class distribution is about 9.6% positive cases, which is about ten percent. A model with random predictions that is aware of the skewed data would randomly predict a positive case one tenth of the time. The measured performance of such a model is likely to have an accuracy of about 81% and an F_1 score about 0.1. From these estimations, it becomes quite clear that all the models are considerably better than random. If a model predicted every subject would live, it would have an accuracy of 90%, but the F_1 score would suffer terribly because the number of true positives would be zero. If a model was to predict every subject to pass on, the accuracy would be 10% and the F_1 score would be about 18%.

8.3 Stacking free-text features with heterogeneous data

The first and second text classifier, were chosen to extract text features from. These two were picked because they performed the best in the previous experiment, and maximizing any impact of text features would also emphasize its utility. It would be interesting to see if these differences in the text classifiers' performances manifest themselves in the classifiers' performances in experiments in Section 7.5.

The overarching goal of Chapter 7 was to answer the research question in Section 4.4; can the extracted information be used in combination with other data? As such, some decisions were made to keep a certain level of abstraction in order to pay attention to the intention of the experiment rather than the gritty details. This is why PyCaret was used to suggest the most fit classifiers, as well as tune them and finalize them. However, this raises a potential limitation, because the trained classifiers might have been more performant if more attention was dedicated to tuning them. The idea is that by maximizing their performances, their differences will be more significant. However, it was still a clear separation between the best models from each of the data sets.

All the experiments followed the same steps. First, construct a data set, which means to aggregate vitality features along with basic information about the subject's admission. For the experiments with text features, these were merged into the data set as well. Second, the top five models were found, and the top three were fine-tuned further. After the models were fine-tuned, they were inspected briefly. Our ability to recognize and reason the different features in the feature importance figures are very limited as we have no background in medicine. There was not much discussion about the features apart from noting which features appeared throughout the figures, and the number of text features in the top ten most important features. The best performing classifier for each experiment, all gradient boosting classifiers, were then evaluated on the test set.

In short, three classifiers were compared to one another. The first was trained on a data set containing text features extracted from the first text classifier, the second was trained on a data set containing text features extracted from the second text classifier, and the third was trained on a data set without any text features.

The first classifier, trained on a data set containing text features from the first text classifier, outperformed the two others. It was followed by the other classifier trained on a data set with text features. The classifier that had no text features available performed the worst.

9. Conclusion

In this thesis created a model in English trained on medical free-text and explored what predictions could be made using that model.

9.1 Can natural language processing with deep learning pick up on linguistic features in clinical notes?

To explore this we trained a language model using the fastai library and utilizing transfer learning to yield quicker and better results. The best scoring language model was trained through an intermediate set, in our case Medline, to try and focus more within the medical field. It got a result of 5.2 of perplexity and 64% accuracy (Table 5.5, which was better results than we were anticipating. When predicting on a small sample of medical notes we saw that the all the models had started to write like the structure of the free-text clinical notes (Tables 5.3 and 5.5). These results are a bit speculative as it is difficult to say exactly how good the text the models predict are, but we can see some definitive indications that the models have learnt some of the linguistic features. Therefore we conclude that a language model can pick up on some linguistic features in the free-text clinical notes.

9.2 Can a text classifier built from the language model extract useful information from clinical text

All of the trained classifiers could with a certainty much better than random predict whether a subject would die or not. Apart from the very worst text classifier that was likely a fluke, the worst text classifiers had a mean F_1-score of greater than 55.0%, which strongly speaks to the model's capability. To conclude, a text classifier built from the language model can indeed extract useful information from clinical text.

9.3 Can the extracted information be used in combination with other data?

The extracted free-text features were observed to attribute positively to the model's performance alongside other features. Upon inspecting the feature importance, the explored model's were proven to utilize both free-text features and other features from an already structured data set. The experiments suggests they complement each other. In conclusion, the extracted free-text features can be used in combination with other data.

9.4 Can the deep learning models be explored and be made sense of?

While we were able to accurately interpret the performance of the text classifiers and inspect the most important features, we were not successful in making sense of why they produced the different predictions. We are under the impression that there is still work to be done here in the field because there seems to be no easily implementable, flexible solutions. However, we are unable to reach any definitive conclusion at this point, because we have not provided evidence for nor against the claim.

10. Further work

For future work, it would be interesting to discover how well the model performs specialized on Norwegian medical data. Unfortunately, as mentioned, it is difficult to get access to such data in Norway, due to strict data regulations. It would also be useful to provide data extraction tools so that other models can also be trained on the same data.

NLP on clinical free-text notes and making predictions on clinical free-text. Our model was trained using ULMFiT and the fastai library, but recently transformer models has been performing better at NLP tasks. Therefore it would be interesting to see if we could further improve the predictions and models using transfer learning on one, or multiple, of the transformer models published on hugging face (Hugging Face, n. d.). Maybe these models could perform better, we do not know. This would likely affect the experiments in Section 7.5 as well.

Creating further experiments on different problems would also be interesting. Could a model predict how long a patient is going to be hospitalized based on the first entry for that stay? Can it predict the illness of a patient? There are many experiments that could be considered and we would have loved to do, but that will be for another time.

Stacking free-text features There are multiple interesting experiments that can be explored. Some of which can possibly further improve the classifiers, perhaps to utilize the text features to a greater extent.

One path to explore would be by combining text features from more than one language model at a time. Perhaps the text classifiers would pick up different patterns and provide different viewpoints that together complement each other. However, if two text classifiers were to produce similar text features, they could overemphasize the characteristic in the text that the feature represents.

Another potential improvement could be to employ common machine learning techniques. For example, by ensembling multiple models or by dedicating more resources to the fine-tuning process. Feature engineering can also improve the classifiers, however that would be ineffective without medical competence. In fact, a medical viewpoint should always be considered and a stronger anchoring to the domain would likely contribute to all parts of this projects.

Additionally, it would be interesting to explore other sources of unstructured data, such as images, and extract features from them. Although we chose to specialize in text, there is no reason that images should not work. The idea still remains to structure the data by extracting features and forwarding them into a traditional machine learning model. It would also be interesting to investigate what the features represent in the image, if one could somehow reverse engineer the process.

Model exploration. A field we did not get to explore as much as we wanted to is the field of explainable AI. We didn't put this high on our list of priorities,

and it was therefore not explored in much detail. There exist several tools for explainable AI in NLP contexts. An example is PureCode (Purecode webpage), which is able to visualize what words the model used to get to its conclusion. Another one is the Language Interpretability Tool (LIT) (Wexler and Tenney, 2020) from Google AI, which produces impressive results. However, LIT is developed for transformer models, and we do not know if our models would work out-of-the-box with this tool. It would still be very intersting for future work to explore our models further to uncover what helped them reach their conclusions.

Part VI

Literature and References

References

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

[2] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[3] K. D. Foote. A Brief History of Machine Learning, 2019. URL `https://www.dataversity.net/a-brief-history-of-machine-learning/`. Accessed: 2020-05-15.

[4] A. M. Turing. Computing Machinery and Intelligence. *the Journal of The Mind Association*, LIX(236):433–460, 1950. doi: https://doi.org/10.1093/mind/LIX.236.433.

[5] Swarthmore Edu. Hodgkin-huxley experiments, n. d. URL `https://www.swarthmore.edu/NatSci/echeeve1/Ref/HH/HHmain.htm`. Accessed: 2021-05-28.

[6] K. D. Foote. A Brief Histry of Natural Language Processing, 2019. URL `https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/#`. Accessed: 2020-05-12.

[7] K. Xue, Y. Zhou, Z. Ma, T. Ruan, H. Zhang, and P. He. Fine-tuning BERT for Joint Entity and Relation Extraction in Chinese Medical Text. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, volume abs/1908.07721, pages 892–897, 2019. URL `http://arxiv.org/abs/1908.07721`.

[8] Wendong G. Hosomura N. Malmasi, S. and A. Turchin. Comparing information extraction techniques for low-prevalence concepts: The case of insulin rejection by patients. *Journal of Biomedical Informatics*, 99(1):1, 2019. doi: https://doi.org/10.1016/j.jbi.2019.103306.

[9] Hassanien A. Shaalan, K. and F. Tolba. Intelligent Natural Language Processing: Trends and Applications". pages 3, 101, 435, Cairo, Egypt, 2018. Springer.

[10] Jeremy Howard and Sylvian Gugger. *Deep Learning for Coders with Fastai and PyTorch: AI Applications Without a PhD*. International series of monographs on physics. O'Reilley, 2020. ISBN 1492045527.

[11] S. Wang, F. Ren, and H. Lu. A review of the application of natural language processing in clinical medicine. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 2725–2730, 2018. doi: 10.1109/ICIEA.2018.8398172.

[12] Alistair EW. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[13] Ary L. Goldberger, Luis AN. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[14] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2 edition, 2018.

[15] Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Inc., 1st edition, 2017. ISBN 1491962291.

[16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[17] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495.

[18] Yordanov Ventsislav. Introduction to Natural Language Processing for Text, 2018. URL `https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63`. Accessed: 2021-05-28.

[19] Tableau. 8 common examples of natural language processing and their impact on communication, n. d. URL `https://www.tableau.com/learn/articles/natural-language-processing-examples`. Accessed: 2021-05-28.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

[22] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182, 2017. URL `http://arxiv.org/abs/1708.02182`.

[23] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhiming Ma, and Tie-Yan Liu. Asynchronous Stochastic Gradient Descent with Delay Compensation for Distributed Deep Learning. *CoRR*, abs/1609.08326, 2016. URL http://arxiv.org/abs/1609.08326.

[24] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/wan13.html.

[25] Chip Huyen. Evaluation metrics for language modeling. *The Gradient*, 2019. URL https://thegradient.pub/understanding-evaluation-metrics-for-language-models/. Accessed: 2021-05-13.

[26] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, 87:12–20, 2018. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2018.09.008. URL http://www.sciencedirect.com/science/article/pii/S1532046418301825.

[27] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

[28] Merity Stephen, Xiong Caiming, Bradbury James, and Richard Socher. Pointer sentinel mixture models. volume abs/1609.07843, 2016. URL http://arxiv.org/abs/1609.07843.

[29] Julie Young. Understanding metrics, 2020. URL https://www.investopedia.com/terms/m/metrics.asp. Accessed: 2021-02-05.

[30] Koo Ping Shung. Accuracy, Precision, Recall or F1?, 2018. URL https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Accessed: 2021-05-29.

[31] Saliya Ekanayake. Precision, reall, and accuracy in binary classification, 2019. URL https://medium.com/@esaliya/precision-reall-and-accuracy-in-binary-classification-160b1f35ccf2. Accessed: 2021-05-29.

[32] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS one*, 12(6):e0177678, 2017.

[33] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[34] The CITI Program. The Trusted Standard in Research, Ethics, and Compliance Training, 2017.

[35] BBC. 'Fake' Amazon workers defend company on Twitter, 2021. URL https://www.bbc.com/news/technology-56581266. Accessed: 2021-05-20.

[36] Jinmiao Huang, Cesar Osorio, and Luke Wicent Sy. An empirical evaluation of deep learning for ICD-9 code assignment using MIMIC-III clinical notes. *Computer Methods and Programs in Biomedicine*, 177:141–153, 2019. ISSN 0169-2607.

[37] Siddhartha Nuthakki, Sunil Neela, Judy W. Gichoya, and Saptarshi Purkayastha. Natural language processing of MIMIC-III clinical notes for identifying diagnosis and procedures with neural networks. *CoRR*, abs/1912.12397, 2019. URL http://arxiv.org/abs/1912.12397.

[38] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets. *CoRR*, abs/1906.05474, 2019. URL http://arxiv.org/abs/1906.05474.

[39] Ergin Soysal, Jingqi Wang, Min Jiang, Yonghui Wu, Serguei Pakhomov, Hongfang Liu, and Hua Xu. CLAMP – a toolkit for efficiently building customized clinical natural language processing pipelines. *Journal of the American Medical Informatics Association*, 25(3):331–336, 11 2017. ISSN 1527-974X. doi: 10.1093/jamia/ocx132. URL https://doi.org/10.1093/jamia/ocx132.

[40] Jonas Kemp, Alvin Rajkomar, and Andrew M. Dai. Improved hierarchical patient classification with language model pretraining over clinical notes. *CoRR*, abs/1909.03039, 2019. URL http://arxiv.org/abs/1909.03039.

[41] F Martin-Sanchez and K Verspoor. Big data in medicine is driving big changes. *Yearbook of medical informatics*, 9(1):14, 2014.

[42] Shirly Wang, Matthew B. A. McDermott, Geeticka Chauhan, Marzyeh Ghassemi, Michael C. Hughes, and Tristan Naumann. Mimic-extract. *Proceedings of the ACM Conference on Health, Inference, and Learning*, Apr 2020. doi: 10.1145/3368555.3384469. URL http://dx.doi.org/10.1145/3368555.3384469.

[43] Bulgarelli L. Pollard T. Horng S. Celi L. A. Johnson, A. and R. Mark. Mimic-iv (version 1.0)., 2021. URL https://doi.org/10.13026/s6n6-xd98.

[44] National Library of Medicine. Medline overview, n. d. URL https://www.nlm.nih.gov/medline/medline_overview.html. Accessed: 2021-02-25.

[45] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[46] Jianping Hua, Zixiang Xiong, James Lowey, Edward Suh, and Edward R. Dougherty. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, 21(8):1509–1515, 11 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti171. URL `https://doi.org/10.1093/bioinformatics/bti171`.

[47] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*, July 2020. URL `https://www.pycaret.org`. PyCaret version 2.3, Accessed: 2021-05-24.

[48] Hugging Face. Models, n. d. URL `https://huggingface.co/models`. Accessed: 2021-05-28.

[49] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[50] James Wexler and Ian Tenney. The Language Interpretability Tool (LIT): Interactive Exploration and Analysis of NLP Models, 2020. URL `https://ai.googleblog.com/2020/11/the-language-interpretability-tool-lit.html`. Accessed: 2021-05-28.