Detection of insurance fraud using NLP and ML

A study on three different NLP-techniques for text classification

Rasmus Bäcklund, Hampus Öhman

Master's thesis
2023:E64

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Abstract

Machine-Learning can sometimes see things we as humans can not. In this thesis we evaluated three different Natural Language Procces-techniques: BERT, word2vec and linguistic analysis (UDPipe), on their performance in detecting insurance fraud based on transcribed audio from phone calls (referred to as audio data) and written text (referred to as text-form data), related to insurance claims. We also included TF-IDF as a naive model. On all models we applied logistic regression on the extracted word embeddings. On word2vec and the linguistic analysis, we also applied a KNN-classifier on the word embeddings. For BERT, we instead opted to apply an LSTM-network on the merged CLS-token embeddings, due to the sequential nature of BERT's architecture.

For the audio data, all models achieved a Macro F1-score higher than 50% on a 95%-confidence interval, with at least one type of classifier. TF-IDF scored 58.2% $\pm$2.6%, BERT 56.0% $\pm$2.6%, word2vec 54.1% $\pm$3.8% and linguistic analysis 53.6% $\pm$3.0%.

For the text-form data, all models achieved a Macro F1-score higher than 50% on a 95%-confidence interval, with at least one type of classifier. TF-IDF scored 56.0% $\pm$2.3%, BERT 57.4% $\pm$0.9%, word2vec 56.0% $\pm$2.1% and linguistic analysis 51.4% $\pm$0.5%.

Each score reported is from using the best performing classifier for that model.

The above findings show that our models manage to learn something from the data, but due to rather small data sets and insurance cases from many different areas, it is quite difficult to draw any conclusions with high confidence. The results are not that much better than "guessing", and the small gain over 50% could be due to something else, such as bias in the data sets.

We feel that there is potential to use these techniques in a real setting, but the topic seems to need further exploration. We especially feel that there is potential in using transformer-based models, such as BERT, but currently it lacks the ability to analyse longer sequences due to computational limitations. With the current development pace of transformer models, it might be possible to use these in the future to get a better representation of what is being said, which hopefully would produce better results.

Popular Science Description

Försäkringsbolag behöver identifiera och hantera en stor mängd potentiellt bedrägliga försäkringsanspråk från deras kunder. Därför har försäkringsbolag en rad system för att försöka identifiera vilka anspråk som är bedrägliga och vilka som inte är det. Med den senaste utvecklingen inom maskininlärning och särskilt inom området för språkbehandling (NLP) kan dessa tekniker eventuellt användas för att underlätta identifiering av bedrägliga anspråk.

Syftet med denna avhandling är därför att undersöka möjligheten att använda NLP för det ovan nämnda syftet. Baserat på transkriberade samtal mellan kunder och företagsrepresentanter, samt på text skriven fritt av kunder eller som svar på olika frågor (beroende på anspråkstypen) kommer vi att träna flera modeller för att förutspå sannolikheten att ett anspråk är försäkringsbedrägeri. De tre olika modellerna som används i detta projekt är

- BERT
- word2vec
- lingvistisk analys

Modellerna kommer att skapa numeriska representationer av dessa texter, vilka sedan kommer att klassificeras med hjälp av olika tekniker inom maskininlärning. Slutligen kommer prestandan för varje modell att utvärderas för att avgöra om detta är ett användbart hjälpmedel i processen att undersöka potentiellt bedrägliga anspråk.

Acknowledgments

We would like to thank our supervisor at LTH, Maria Sandsten, for guiding us through the creation of this Master Thesis. We would also like to thank our supervisor, Fredrik Thuring, and the rest of the team at Trygg-Hansa, for supporting us through out this interesting project and providing us with the necessary tools and equipment that it required.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

The aim of this project was to assess if it is possible to predict fraudulent insurance claims using Natural Language Processing-techniques on data from both investigation calls and online data written by the customer. All modeling was done on text data. Furthermore, all customer claims are labeled as fraudulent or legitimate by claim handlers at Trygg-Hansa.

1.1 Problem background

An insurance company receives a significant amount of insurance claims every day. The vast majority of the claims are valid, however, some of these are fraudulent, meaning the insurance holder is lying about what reportedly happened to increase the possible claim amount. To deal with this, the company makes use of data processing to be able to automatically detect suspicious cases and send them for manual investigation. Both customers and claim handlers create text descriptions of the claims and it is believed that adding automatic analysis of these texts might enhance the performance of the data processing. Therefore, with the recent development in Machine Learning (ML) and Natural Language Processing (NLP), it might be possible to make use of these technologies to assist in the process of detecting fraud.

Using NLP-techniques to predict fraudulent texts is a rather unexplored subject, especially in the context of insurance claims. Some research has been done on predicting lies from conversations, predicting fake news, and financial scams.

1.2 Problem formulation

The prediction of fraudulent claims was to be done only based on text from the customer. We defined all text from the customer corresponding to a specific insurance claim as a stochastic process denoted by X. Furthermore, the correct label corresponding to X was denoted by Y, where:

$$Y = \begin{cases} 1 & \text{fraudulent claim} \\ 0 & \text{legit claim} \end{cases} \tag{1.1}$$

The goal was formulated as estimating p, where

$$p = E(Y|X) \tag{1.2}$$

using different types of NLP-models.

1.3 Research questions

- Is it possible to predict insurance fraud with NLP-techniques from conversation recordings between a customer and company representative or text data entered by the customer either in free text form or from Question & Answers?
- How do different NLP-techniques compare to each other when it comes to predicting fraud?

1.4 Data & Approach

1.4.1 Data

The data is divided into two parts. One containing 1334 unique phone calls corresponding to 1014 different insurance claims. This data set will be called "audio data" from here on. The other one contains short written descriptions of insurance claims as well as written responses to simple questions about the claims, both of which are written by the customer in an online form. This data set will be called "text-form data" from here on. The two different data sets are visualized in Figure 1.1.

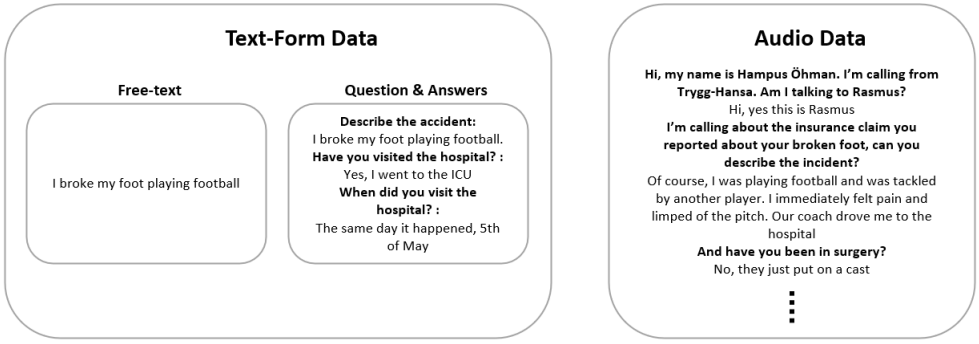

Figure 1.1: A visualization of the two data sets. The text-form data is located on the left-hand side. It consists of two subsets. The first one is free-text, which is when a customer describes the accident in free-text form. The second one is Questions & Answers, which is when a customer fills out a form with different questions. The audio data is on the right-hand side. For visualizing purposes, a short transcription of a conversation is presented. Bold text represents the company and regular text represents the customer. **Please note** that all text is made up and is not correlated to any actual insurance cases.

All data, both audio data and text-form data, are labeled as either fraudulent or not. This label is determined by one or several company representatives. It is important to note that all data points are given labels for the case as a whole, even if there are several calls or texts per case. It is also important to note that these labels are given by a company representative according to Trygg-Hansa's policies for determining insurance fraud. This means that we will be modeling whether or not a case is fraudulent

based on these policies, and the labels can therefore not really be called the "ground truth" for insurance detection outside of Trygg-Hansa.

1.4.2 Approach

Since all models are to be based on text data, the first step in the project is to transcribe all conversations in audio data using a speech-to-text model. In this step, the speakers are also separated from each other. We only want to predict the probability of fraud based on what the customer says and exclude the company representative.

Early in the project, it was decided to limit the number of models to three. Evaluating more models were deemed to be too time-consuming. It would also risk that the time spent on each model would decrease, which could affect the performance. A naive model was also included among the models. The use of a naive model is common in these types of projects. First off, it lets the authors get acquainted with the data. It also provides a benchmark to compare the results from the later models. The naive model was chosen to be a Term Frequency-Inverse Document Frequency (TF-IDF) together with logistic regression. This is a model simple to understand and simple to implement.

It is worth noting that for the more advanced models, none have been built and trained from scratch. With the resources invested by companies like Google, OpenAI, etc. it would be impossible for us to pre-train something that could compete with e.g. BERT. For the choice of the three models, we wanted models that were as different as possible from each other. We reasoned that this would yield the highest probability of finding a successful model. Even if the two data sets are somewhat different, we decided to use the same three techniques for both data sets. The three techniques chosen are:

1. BERT
2. word2vec
3. Linguistic analysis

During the training of all models, the data will be split into folds for k-fold validation. All models will consist of one NLP-part and one classification part. i.e. the data will be passed through an NLP-model which will output word embeddings. These will then be used to predict the probability of fraud with some kind of classification (e.g. logistic regression, k-nearest neighbor, or a Long Short-Term Memory (LSTM)). The same techniques (in general) will be used on both data sets. An overview of the workflow can be seen in Figure 1.2.

The test data in each fold will be used to compare the models. The label of each data point will be predicted, based on the estimated probability of fraud. These predicted labels will then be used to calculate recall, precision, and Macro F1-score, which will be our measures for comparison.

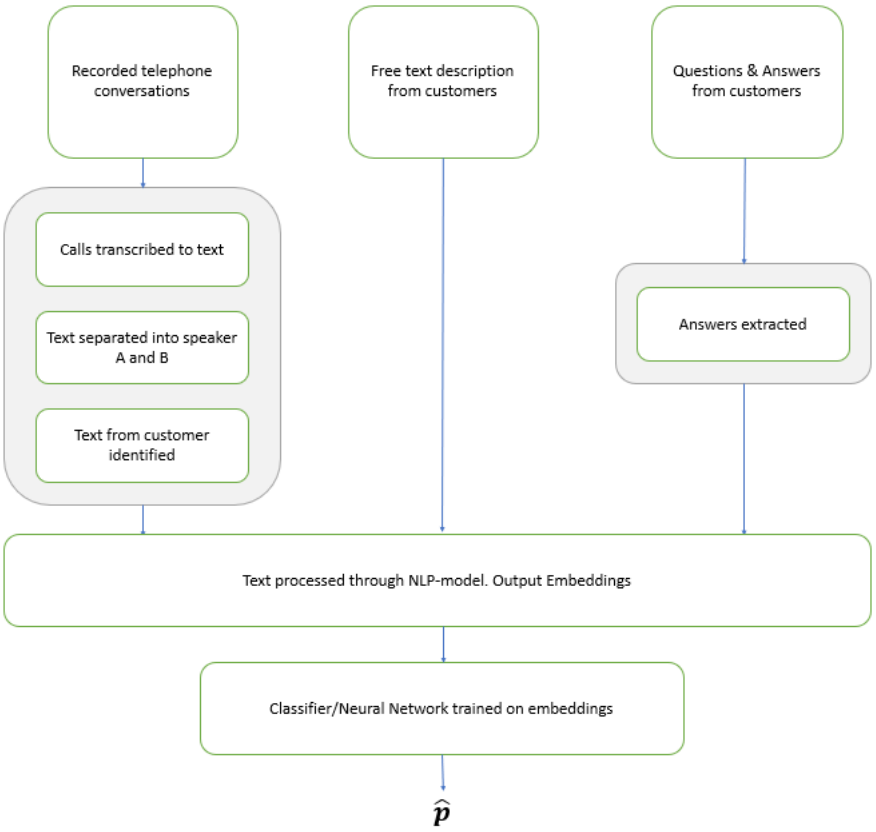

Figure 1.2: An overview of the used work-flow for this project.

Chapter 2
Background and Related Work

2.1 Background

This section will contain the background that is the foundation of this project. It can be seen as a short introduction to NLP in general and certain techniques in particular. Furthermore, the section also gives a short introduction to deception.

2.1.1 Deception

Throughout the history of society, humans have tried to detect lies and deception. Thus, lots of research have been done on the subject. A simple search of "lie detection" in Google Scholar results in over 4 000 000 results. Today there are many methods for detecting deception. Some of these are: Polygraphs, Voice Analysis and Brain observations [16]. However, these methods are often difficult to implement and hard to use on a larger scale. Furthermore, a method like Voice Analysis is also rather inaccurate. Another approach is to look at written text and its characteristics and structure.

2.1.2 Natural Language Processing (NLP)

NLP is a field within Artificial Intelligence and linguistics that revolves around building machines that are able to understand and generate human language. The technologies created within the field are sought to solve specific tasks such as translation, text generation, sentiment analysis, text classification, and question-answering among many other tasks. In recent years there has been large development within the field. NLP-models are used in applications like conversational agents (Alexa or Siri), chatbots like GPT-4, or in Google's search engine to complete sentences and improve results [5].

General architecture of an NLP-model consists of pre-processing, feature extraction, and some sort of modelling. The pre-processing often includes removing stop-words and tokenization of the input. For a machine to be able to use and understand the text it has to be tokenized, usually into a numerical representation. To create a model from the tokens, there has to be a feature extraction. The feature extraction can be of a simpler method, for example counting the occurrence of certain words. There are also more advanced methods where the machine is trained to understand the relationship between words and how they are used in a sentence. Each word is then given a word

embedding, which is the numerical representation of that word. For these feature extractions, the same word can have very different embeddings depending on the context of the word. Lastly, the features extracted are typically used together with a Neural Network or a Transformer to be able to perform the desired task [5].

2.1.3 Term Frequency-Inverse Document Frequency (TF-IDF) with logistic regression

TF-IDF is a way of determining the importance of words in a document or corpus [12]. It is a rather simple NLP-technique, but a widely used one. Simply put, the method counts the occurrence of a word in each text and takes into account how many of these texts the word appears in.

2.1.4 Transformers

Traditionally, NLP models and sequence transduction tasks have been built with either RNNs or CNNs. In 2017, Vaswani et al. proposed a new architecture using only attention mechanisms, in the paper [32]. One limitation with models like RNNs or CNNs is how computationally expensive it is to relate inputs and outputs at different positions in a sequence. The number of operations required usually grows linearly or logarithmically with the distance between the positions. Whereas for a transformer, the number of operations required is constant, thus making it easier to create relations between distant positions in a sequence.

Since invented, Transformers have been used widely for different NLP-tasks, especially for pre-trained language models. These pre-trained models are trained on large, often unlabeled, data sets, such as Wikipedia pages, news articles, etc. To perform a specific task, these models are then fine-tuned for the desired task. Some of the most prominent models are GPT, BERT, and XLNET. In this thesis, a model will be built based on BERT.

2.1.5 word2vec

word2vec is an NLP-technique that was presented by a team at Google in 2013 [19]. It was one of the breakthroughs in the NLP-world within "transfer learning", which means that the training of the model can be done by someone with lots of data and computational power and the parameter values can then be shared and used by someone else. The model works by creating numerical representations of words in the form of vector embeddings. These make up a vector space, where similar words or words that often appear in the same context are located close to each other. The two most common architectures of word2vec are Continuous Bag of Words (CBOW) and Skip-gram. CBOW is a lot faster and works better with frequently occurring words, and comparatively, Skip-gram is more computationally heavy but does a better job at representing rare words and phrases.

2.1.6 Speech-to-text with Whisper

Whisper is a collection of five speech recognition models able to perform transcription of speech in multiple languages as well as speech translation from multiple lan-

guages to English. The five different models are trained with a different amount of parameters, ranging from 39 million for the smallest model "tiny", to 1550 million parameters for the largest model "large. The performance varies widely depending on which language is being transcribed, where Spanish and English perform the best. For example, when tested on the FLEURS dataset, which contains speech and correct transcriptions in different languages, Swedish has a Word Error Rate (WER) of 8.5% compared to 3.0% for Spanish.

The model is largely built on the Transformer architecture that consists of multiple encoder and decoder blocks. This is the same type of architecture that BERT uses, so for more information on that part read Section 3.2.2. Together with the Transformer architecture Whisper also uses a multitask training format, which allows it to use several different components, such as language recognition, voice activity detection, and translation.

WhisperX

WhisperX is a fork of Whisper that provides more accurate timestamps than Whisper, by incorporating other audio-related Python packages [1]. The more accurate timestamps can be used together with NeMo to produce better diarization.

2.1.7 NeMo

NeMo is a toolkit provided by NVIDIA that contains models for different NLP-related tasks [22]. One of those is speaker diarization. NeMo is able to separate speakers in audio files, which it does by using Voice Activity Detection (VAD), Speaker Embedding Extraction (SEE), and a Neural Diarizer (ND). The VAD detects the presence or absence of speech and generates timestamps for these, the SEE extracts speaker embeddings, from the parts VAD flagged as active, which contain voice characteristics and the ND estimates the speaker labels from these features.

2.2 Related Work

There is some previous research that might be helpful for the purpose of this project. This research will be shortly described in this section. It involves both specific research about insurance fraud for Trygg-Hansa and research about deception and NLP in general.

2.2.1 Tonal Analysis

A similar project, also done for Trygg-Hansa, was carried out by Steneld in 2022 [10]. The aim of the project was the same as this project, i.e. to predict fraudulent insurance claims. Steneld tried to achieve this by performing Tonal Analysis. Thus, the data used was only voice recordings of investigations, called audio data in this project.

The models used in Steneld's project did not manage to generalize the validation data. That is, the models were not able to predict whether an insurance claim was fraudulent or not.

During the project, some pre-processing of the data was done. First off, the silence was removed. Secondly, a speaker diarization was performed. Since the aim was to determine whether a customer's claim was fraudulent or not, only the audio of the customer was interesting. The diarization was done in two parts. First, turning points was identified, i.e. points where there is a change of speaker. Then two audio streams were constructed, where each audio stream consists of all audio from one of the speakers. Secondly, the customer and the handler from Trygg-Hansa were identified. Simplified, this was done by matching audio streams from calls that were known to be with the same handler. We can see this process as a Binomial distribution, with p being the probability of a speaker being correctly labeled. The identification was manually evaluated on 20 calls. 18 of these 20 were deemed to be correctly labeled, resulting in a 95% confidence interval of $I(0.66 \;\; 0.99)$ for p.

2.2.2 NLP-techniques for fraud prediction

To our knowledge, the research on using NLP on insurance-based deception is very limited. However, some research has been carried out in the area of NLP-techniques for deception tasks in general.

Newman et.al [20] claim that their research found three categories associated with deception. These are fewer self-references, more negative emotion words, and fewer markers of cognitive complexity. The authors predict lies/truths by first classifying words and then they fit a logistic regression to the data. Words are classified into 72 different categories according to the Linguistic Inquiry and Word Count (LIWC) dictionary [24]. One word can be part of several categories. For example, "cried" is a part of sadness, negative emotion, overall affect, and past tense verb. The authors reduce the number of categories using three rules. First off, they remove categories that were thought to reflect the content of the document. The reasoning was that the goal of the model was to be able to predict deception independent of the subject. Secondly, categories with very low frequency were removed. Finally, they also removed categories that might be unique to spoken or written transcripts, for example, *uhm* or *eh*. They achieve a prediction accuracy of 61 % across their five different studies.

Duran et.al [8] did a similar project to Newman et.al in 2010. Like the authors of [20], their idea is to classify words into different word classes. Instead of using the LIWC library, they use the Coh-Metrix software. Coh-Metrix tracks word features on a deeper level than LIWC. It has over 700 linguistic indices, compared to the 72 of LIWC. The main difference between Coh-Metrix compared to LIWC is that Coh-Metrix asses a collection of words, rather than every word individually [8]. The study let two students conversate about four given subjects, where one of them was instructed to be deceptive on two of the subjects. The authors then search for linguistic indices that had a significant difference in usage across the deceptive conversations and the truth full conversations. For example, they found that the speakers used more third-person pronouns when being deceptive.

Chang et.al created a chatbot to help detect financial fraud in Taiwan, first published in [3]. The purpose of the chatbot is to let people consult the chatbot to help them determine whether an incoming claim is legal or not. The user describes the situation to the chatbot. The chatbot is built in two parts. The first part extracts the semantics

of the incoming text and the second part predicts whether it is illegal or not. There are three possible outputs of the second part of the bot, which are legal, illegal, or pending confirmation. The chatbot is trained on the seven most common financial frauds in Taiwan, which are related to: ATMs, shopping websites, mobile payments, and counterfeiting. The authors try four different NLP-techniques to get word embeddings from the text. These are: Word2vec, ELMO, BERT and DistilBERT. These are used together with six different classification methods, namely: random forest, naïve Bayes, SVM, Adaboost, K-nearest-neighbour, and an ANN (Artificial Neural Network). For fraud detection, they achieve an accuracy of 98,7 % using DistilBERT with a Random Forest classifier. It is worth noting that the nature of the fraud and the fact that they use three classes for classification enables great accuracy.

Chapter 3
Theory

This chapter seeks to further explain the theory used in this project. For instance, the NLP-techniques that have been used will be covered more deeply. Furthermore, all methods used for classifying will be described.

3.1 Related Machine Learning (ML)

This section will cover some basic Machine Learning techniques relevant to the project. The purpose of the section is to give readers that are inexperienced in the field of Machine Learning some basic knowledge about ML to be able to follow along in the rest of the thesis.

3.1.1 Machine Learning terms

This section will cover some relevant Machine Learning terms, especially related to training settings [6].

- Epoch - number of full training passes. For each epoch, all data in the training set is used to train the model.
- Batch size - number of data points the model processes per iteration.
- Loss Function - the function used in training to calculate how far off the model is from the actual label.
- Optimizer - numerical method for minimizing the loss.
  - ADAM - stands for Adaptive Momentum. It is an optimizing algorithm for gradient descent.
- Learning Rate - a number that tells the optimizer how strongly weights and biases should be adjusted between iterations.
- Label - the correct "answer" of a data point. In this project Fraud or Legit.
- Iteration - a single update of model parameters.
- Hidden layers - All layers between the input layer (with features) and the output layer (with predictions).

- Fine-tuning - a second optimization to parameters on an already trained model, usually for a new problem that is different from the one in the original training.
- Dropout - sometimes used in Neural Networks for regularization. It randomly removes a number of units in a network layer for a gradient step. i.e. selected weights are not updated in that iteration.

3.1.2 Neural Networks (NN)

A neural network is a connection of nodes, interlinked with each other through connections. In a simple dense feed-forward neural network, all nodes are connected and the data is passed forwards through the model. There is an input layer, where the data is entered, an output layer, which usually produces some kind of prediction or probability, and all layers of nodes in between are called hidden layers. For each connection between one layer and another, there is a weight. These weights are multiplied by the incoming data and summed up for each node. Depending on the result, the node will "fire" data to the next layer of nodes, or stay inactive. The weights related to these connections are initialized randomly and updated throughout the training of the neural network [9].

3.1.3 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are a type of Neural Networks. It is commonly used when dealing with sequential data, for example, speech or time-series data. The main difference between an RNN and an NN is how information is passed through the network. In an NN, the inputs of a sequence are assumed to be independent of each other, meaning no information is passed between two inputs. RNNs on the other hand, cycle information back to itself. This allows an RNN to not only consider the current input X_t, but also previous inputs $X_{0:t-1}$ [28]. A simple overview of the architecture of an RNN compared to a Feed-Forward Neural Network can be seen in Figure 3.1.

3.1.4 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of RNN. It is specially designed to handle longer sequences of data. One weakness of RNNs is their incapability to learn long-term dependencies, making them less efficient when handling sequential data where the relevant connection of inputs is further apart. LSTMs handle this by using a memory cell. The memory cell can handle and store data for a longer period of time. The memory cell consists of three gates, an input gate, a forget gate, and an output gate. The gates have rather self-explanatory tasks. The forget gate controls what information is to be kept and what information is to be discarded from the memory cell. The input gate controls what information is fed into the cell and lastly, the output gate controls what information is passed on from the memory cell [11].

3.1.5 Logistic regression

Logistic regression is a statistical technique for classifying and analysing data sets. Consider a set of data $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of data points and d is the

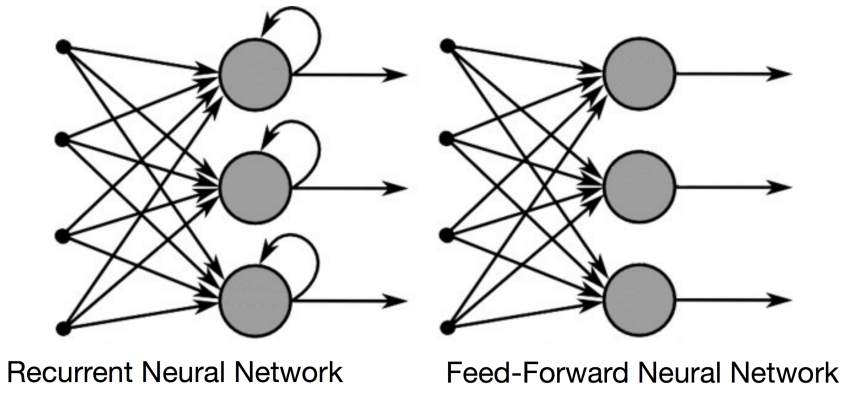

Figure 3.1: Overview of the architecture of a Recurrent Neural Network (on the left hand side) and a Feed-Forward Neural Network (on the right hand side). Figure source: "How I Classified Images With Recurrent Neural Networks" [13].

number of features in each data point. Further, let $\mathbf{y}$ be a vector of binary outputs such that $y_i = 1$ or $y_i = 0$. A linear regression could then be described as

$$\mathbf{y} = \mathbf{X}\beta + \epsilon\,, \tag{3.1}$$

where β is a vector of unknown parameters and ϵ is an error vector. y_i is assumed to be a Bernoulli random variable with probability p_i. Since the desired estimation, $\hat{y}_i$, of y_i is a probability between 0 and 1, the regression in equation (3.1) is poorly suited for the task. Instead, the logistic function is introduced as

$$\mathbb{E}[y_i = 1|x_i, \beta] = p_i = \frac{e^{x_i\beta}}{1 + e^{x_i\beta}} = \frac{1}{1 + e^{-x_i\beta}}\,. \tag{3.2}$$

Assuming that all observations are independent, the likelihood function is

$$\mathbb{L}(\beta) = \prod_{i=1}^{n} (p_i)^{y_i}(1 - p_i)^{1-y_i} = \prod_{i=i}^{n} \left(\frac{e^{x_i\beta}}{1 + e^{x_i\beta}}\right)^{y_i} \left(\frac{1}{1 + e^{x_i\beta}}\right)^{1-y_i} \tag{3.3}$$

and the log-likelihood function

$$\ell(\beta) = \ln(\mathbb{L}(\beta)) = \sum_{i=1}^{n} y_i \ln\left(\frac{e^{x_i\beta}}{1 + e^{x_i\beta}}\right) + (1 - y_i)\ln\left(\frac{1}{1 + e^{x_i\beta}}\right)\,. \tag{3.4}$$

To fit the parameters β, the Maximum log-Likelihood (ML) is used. That is, β is set so that the value of (3.4) is maximized. The ML does not have a closed-form solution. Thus, the Maximum Likelihood Estimations are achieved through the use of numerical optimization algorithms. Once the parameters have been fitted, probabilities of new data point i can be estimated using the features in x_i. If dealing with unbalanced data sets, one can use weighted logistic regression. In short, this means that the loss in the optimizer is calculated differently for different classes. The optimizer is "punished" more if it predicts the less common class [17].

3.1.6 K-nearest neighbor

K-nearest neighbor is a classifying technique for Machine Learning. It classifies classes in a simple way. The *k*-nearest neighbors are identified, and the class is determined by those neighbors. There are a few ways to determine which points are the closest neighbors. The most common one is to use the euclidean norm of all input features. The *k*-nearest neighbors then cast a vote on which class the data point should belong to. The class with the most votes is then the prediction of the algorithm. One could also use a weighted K-nearest neighbor, weighting the weight depending on the distance [4]. We introduce a simple example in Figure 3.2. If $k = 3$, the predicted class would be red, since two out of the three nearest neighbors are red. If we instead choose $k = 5$, the predicted class would be blue.

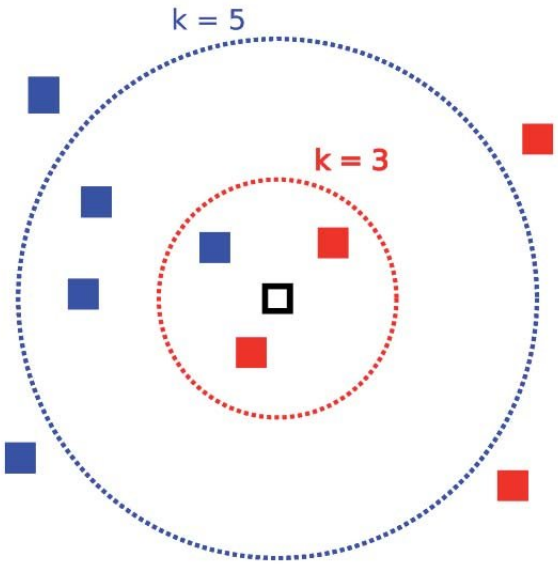

Figure 3.2: Simple overview of the K-nearest neighbor algorithm. The white square represents the point desired to classify. The red and blue squares represents training data with two different classes. Figure source: "Visualization of Uncertainty in LANDSAT Classification Process" [30].

3.2 NLP-techniques

3.2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

To understand how the TF-IDF works, we introduce a definition for document and corpus. In our case, a document is a string representing all free-text inputs for a specific case. In our case, the corpus is all the documents in the data set. The Term-Frequency is based on the number of occurrences for each word (term) in a document. Since the length of a document varies from just a few terms to many sentences, the frequency is divided by the total number of terms in the document. Formally, we get

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}. \tag{3.5}$$

Where $f_{t,d}$ is the total number of occurrences of term t in document d and the denominator is the total number of terms in document d.

Just looking at the occurrences of words will give common words such as: "the", "a" and "is" high importance. Thus, the Inverse Document Frequency is introduced. The idea is to increase the importance of words that occur rarely in the corpus and diminish the importance of frequently occurring words. To compute this, the fraction of documents with the specific word is calculated. That is the number of documents, d, containing the term t is divided by the total number of documents in the corpus, N. Then, we take the natural logarithm of the inverse of this fraction. More formally we get

$$idf(t, D) = \log\left(\frac{N}{|d \in D : f \in d|}\right). \tag{3.6}$$

The denominator is the number of documents d containing the term t.

To compute the final weight, these two terms are multiplied, resulting in the $T \times N$ matrix $tf - idf(t, d)$. T is the total number of unique words in all documents and N is the total number of documents in the corpus. A simplified example of the matrix is shown in Table 3.1.

	Document 1	Document 2	...	Document N
Term 1	tfidf(t_1, d_1)	tfidf(t_1, d_2)		tfidf(t_1, d_N)
Term 2	tfidf(t_2, d_1)	tfidf(t_2, d_2)		tfidf(t_2, d_N)

Table 3.1: Simple overview of how the TF-IDF matrix will look. tfidf(t_i,d_j) represents the weight of term i in document j.

3.2.2 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers, or BERT, is a language representation model that uses Transformers. It was proposed in 2019 by Devlin et.al. [7].

BERT pre-training

One of the main advantages of BERT, according to the authors, is that it is bidirectionally pre-trained. This is in opposition to other pre-trained models which traditionally are pre-trained either in a left-to-right context or a right-to-left context. The pre-training for BERT is done with two different tasks. The first one is Masked Language Modelling. In the input sequence, 15 % of the tokens are masked at random. The task is then for the model to predict the masked tokens. With this approach, the model uses the context both on the left and the right side of the masked token.

The other pre-training task is Next Sentence Prediction. Two sentences, A and B are chosen from the corpus. 50 % of the time, B is the next sentence from A, and the other half B is just a random sentence from the corpus. The task is then for the model to predict whether B is actually the next sentence or just a random sentence from the corpus.

The architecture of the pre-training allows the model to be easy to fine-tune for specific tasks, with state-of-the-art performances. The fine-tuning is simply done by adding a task-specific input and output, and then all the parameters are fine-tuned.

Sequence limitations of BERT

One limitation of Transformers is the complexity required for long sequences. The self-attention mechanism has a computational complexity of $\mathcal{O}(n^2)$. Thus, the sequence length is limited to 512 tokens. For the purpose of this project, 512 is not long enough. Some of the recorded calls are over one hour long and will therefore contain much more than 512 words. To deal with this, there are mainly two proposed options in the literature [14].

The first option is to use a "hierarchical" or "concatenating" approach. The idea is to split the document into fractions of length 512 [23]. Then, every fraction can be run through a BERT-model. To compose this into some kind of result, the output of each fraction is used. There are several options for this stage. One could use a Transformer once again, some RNN or CNN, or a more simple method like logistic regression.

The other option is to build a Transformer-based model that is not limited to 512 words per document. To accomplish this, the attention-mechanism is slightly modified. Two of the most prominent models for long sequences are Longformer and Big Bird [2] [33]. The attention in Longformer is based on windowed attention. i.e. every token attends to $w/2$ tokens to its left and $w/2$ tokens to its right, where w is the window size. Furthermore, some tokens are chosen, at random, to have global attention. Tokens with global attention will attend to all other tokens. The attention in Big Bird is similar but with an addition. Big Bird also introduces random attention. r tokens will attend another token, chosen at random. These approaches reduce the computational complexity of the attention-mechanism, which allows a sequence size of 4096.

Hierarchical BERT

The application of a hierarchical BERT has been tested in the paper [23]. The authors split the input sequence into fractions with overlap. BERT has two useful outputs for this purpose. First of all, it outputs a pooled representation of the last transformer block and also a posterior probability. The authors use both outputs, but separately. Once each fraction has been run through BERT, the outputs form a sequence. This sequence is then processed either through an RNN or a Transformer.

3.2.3 word2vec

Word2vec creates numerical word representations, called embeddings, through training done with shallow neural networks. There are two main architectures of word2vec, Continuous Bag Of Words (CBOW) and Skip-gram. These are trained a bit differently but are then used the same in a classification task, where the created embeddings are used as inputs to a classifier. Due to the need for large amounts of data to get proper embeddings, it is possible to use someone else's pre-trained model on your own data. The training is explained below.

CBOW

The CBOW architecture is a feedforward neural network, that consists of one input layer, one projection layer, and one output layer[19], see the left part of Figure 3.3. The input layer is of size N, where N is how far forward and backward the "context window" is looking. Each element will consist of a 1-hot encoded vector representation of that word, which will be the length of the vocabulary, V. A 1-hot encoded vector is a vector with all zeros except for a single 1, which represents the element in question. These vectors, or embeddings as they are called, are then averaged out in the projection layer and sent to a dense softmax layer which outputs the most likely target word. Softmax is an activation function that takes all inputs and turns them into a probability distribution between 0 and 1, where the higher the value the more likely it is to be the correct prediction, according to the model.

Conceptually this can be thought of as follows. For each target word, an input/expected output relationship will look like ([context], target word). The context size is determined by the window size N. If the sentence is "I was playing football and was tackled by another player" and the window size is 2, three examples of the input/expected output relations are ([was, football], playing), ([playing, and], football) and ([tackled, another], by). The model tries to predict the target word from the context, computes the loss, and then adjusts the parameter weights through backpropagation. This is done for all such relations and then repeated for several epochs.

Skip-gram

The Skip-gram architecture can be seen as the opposite of the CBOW architecture. Instead of using context to predict a word, it uses a word to predict the context, see the right part of Figure 3.3. The training is done by grouping (context, target word) together, like in CBOW, while also grouping a randomly chosen word together with the target word, like (random word, target word). The true pairs are given a positive label (1) and the "fake" pairs are given a negative label (0), in hopes of teaching the model which words are contextually relevant together and which are not. The word embeddings are created just as in CBOW. Then the dot product of the vector embeddings of the pairs mentioned above is calculated and put through a dense layer with a sigmoid activation function, which outputs either a 1 or a 0. Then the weights are updated with the same procedure as in CBOW.

3.2.4 Linguistic Analysis with UDPipe

Research suggests that the use of certain word classes can be an indicator of deception. The authors of [8] use the LIWC library. The LIWC library is not available in Swedish, meaning it can't be used for this project. Instead, we went with a similar method called UDPipe. UDPipe is a multilingual pipeline that provides language-agnostic tokenization, tagging, lemmatization, and dependency parsing of raw text. It allows the user to map words into different word classes, such as verbs, conjunctions, adjectives, etc.

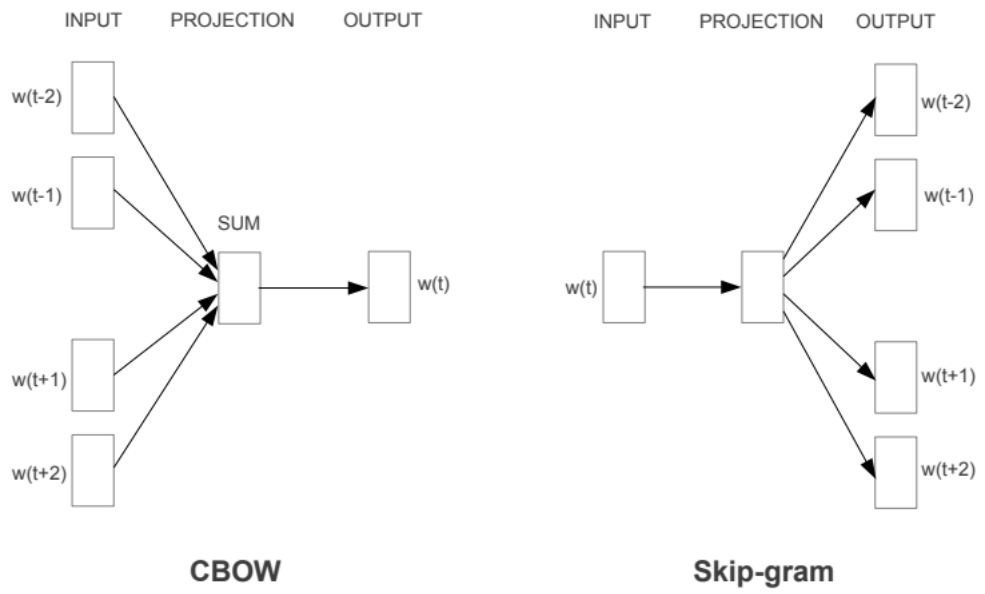

Figure 3.3: Model architectures for word2vec. CBOW predicts the current word based on the context, while Skip-gram predicts the surrounding words given the current word. Figure source: "Efficient Estimation of Word Representations in Vector Space" [19].

3.3 Evaluation methods

3.3.1 F1-score

F1-score is a metric that can be used to evaluate classification tasks. The F1-score is based on two terms, Precision and Recall. To understand these terms, we introduce the confusion matrix, see Figure 3.4. Precision is then calculated as

$$P = \frac{TP}{TP + FP}, \tag{3.7}$$

where TP is True Positive and FP is False Positive. Recall is calculated as

$$R = \frac{TP}{TP + FN}, \tag{3.8}$$

where FN is False Negative. Lastly, the F1-score is the harmonic mean of Precision and Recall. That is

$$F1\text{-}score = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2}{\frac{P+R}{PR}} = \frac{2PR}{P+R}, \tag{3.9}$$

where P is precision and R is recall [27]. The F1-score is calculated for both classes. Macro F1 is calculated as the arithmetic mean of the F1-score for each class

$$Macro\ F1\text{-}score = \frac{\sum_{i=1}^{n} F1\text{-}score_i}{n}, \tag{3.10}$$

where $F1\text{-}score_i$ is the F1-score for class i.

The F1-score is especially useful, compared to a more simple metric like accuracy, if the data set is unbalanced. An unbalanced data set is one where one label is more common than others [15]. This can be illustrated with a simple example. Consider a data set where 90 % of the points are labeled 1 and 10 % are labeled 0. Further, let's say the model only predicts label 1. Looking at the accuracy (which will be 90 %) the model seems to be doing great when it in fact has learned nothing.

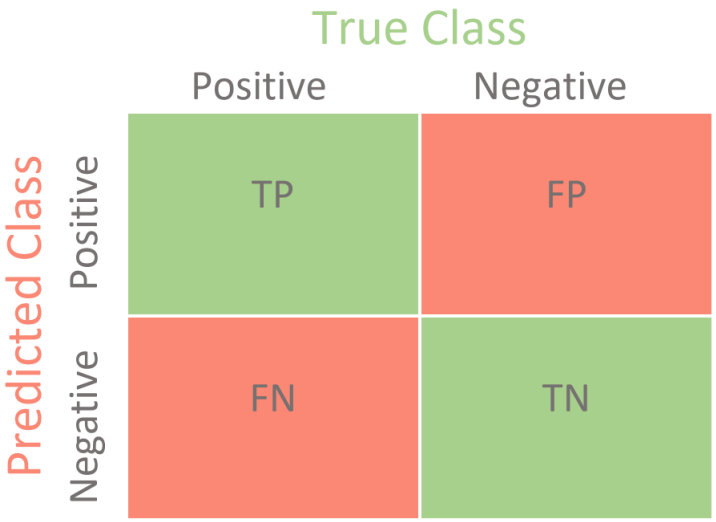

Figure 3.4: Outline of a confusion matrix in a binary classification task. Figure source "Weighting Confusion Matrices by Outcomes and Observations" [29].

3.3.2 K-fold cross validation

When training machine learning models generalization is desired. In other words, you want the model to perform well on data that it has not been trained on, i.e. data that the model has not seen before. This is one of the reasons that data is often split into training data and test data. To evaluate a model's performance one usually looks at how well it models the test data.

K-fold cross-validation is a method for splitting data into a training set and a test set. The data is divided into k number of randomly drawn subsets, called folds. Then, the model is trained on $k-1$ folds and evaluated on 1 fold. This is repeated k times so that every fold is the test set once. At the end of each iteration, the desired evaluation scores are retrieved and averaged to get a mean [25]. Furthermore, this allows for a standard deviation to be estimated. Thus, a confidence interval can also be constructed. An overview of k-fold cross-validation can be seen in figure 3.5.

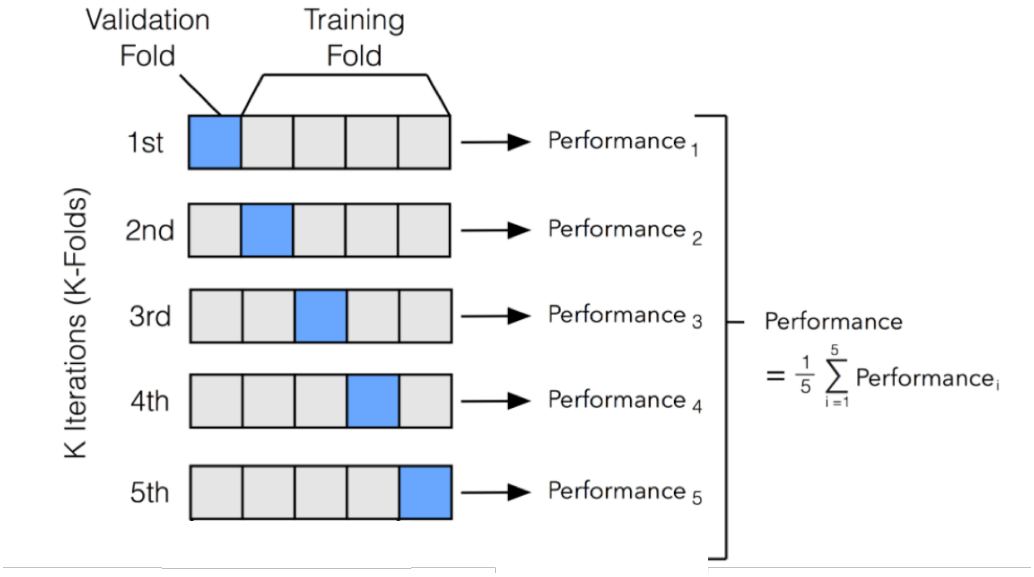

Figure 3.5: Overview of k-fold cross validation with $k = 5$. Each iteration represents training and validation of a model. Image source "K Fold Cross Validation" [26].

Chapter 4
Method Results

As mentioned before we decided to test three models as well as a naive one. We opted for three models that would be as different as possible. This was thought to be a good approach to maximize the chances of finding a model that performs well on the data. We chose TF-IDF together with a logistic regression as our naive model and three following NLP-techniques as our main models.

1. BERT
2. word2vec
3. Linguistic analysis (with UDpipe)

The purpose of this chapter is to describe the overall procedure of the project. i.e. how we went from raw data to building models, to classifying and finally evaluating the results. It will consist of three main parts:

- Pre-processing of data
- Choice of NLP-techniques to model the data
  - and their respective results
- Summary of results

These parts will be presented separately for each data set, as the procedure differed quite a lot between the two.

All methods are evaluated using the confidence interval of Precision, Recall, and Macro F1-score, described in section 3.3.1. The reason for using the Macro F1-score is that the data sets are imbalanced. The Macro F1-score is the average of the F1-score for each class, providing us with a fair metric for the model's performance despite an unbalanced data set. To ensure that the model does not stem from a beneficial split of training and test data, we used k-fold cross-validation. All tables will be presented with the mean of the measure as well as a 95%-confidence interval. i.e. $\pm\hat{\sigma} * 1.96/\sqrt{n-1}$, where s is the estimated standard deviation, 1.96 is the z-value for a 95% confidence interval and n is the number of folds.

4.1 Audio data

4.1.1 Pre-processing of the data

First off, all calls have to be transcribed since our models can only handle text data. In this step, the speakers were also separated from each other. This was done using a combination of Whisper, WhisperX, and NeMo. Whisper performs the initial transcriptions, WhisperX gives the transcriptions time stamps with improved precision and NeMo performs the diarization. We discussed using the diarization already done by Steneld in his project [10]. We tried running Whisper on a few calls using this diarization, however, we found that the transcriptions were more accurate when we used the whole conversation as input to Whisper.

There were about 300 calls that NeMo was unable to diarize, probably due to poor audio quality. Since our data set is quite small, we wanted to minimize data loss. Therefore we decided to use Steneld's diarization for these in complement to the procedure above.

Once the audio data was transcribed, we had to identify which one of the speakers that were the customer. We considered basing this identification on some simple "rule" that could be easily implemented but found that none of the rules we came up with were consistent through all calls. Instead of spending a lot of time on building something that we did not even know would be able to handle this issue well, we decided to do it manually. This was done by simply reading through the transcribed text and determining which of the speakers was more likely to be the customer. Albeit time-consuming, this ensured the quality of the identification. It also gave us an insight into the quality of the transcriptions.

Some of the insurance cases had several investigation calls. All calls regarding the same insurance case are considered to be the same data point. This is because there is only one label given per claim and there is no way for us to tell beforehand which call would "contain the fraud". Thus, the diarized transcriptions of calls belonging to the same insurance case were merged together. This was done based on the case-id belonging to each insurance case.

4.1.2 NLP-techniques and results

The audio data originally consisted of 1334 calls. 1049 of these were diarized by NeMo, 185 by Steneld's method, and the rest were discarded due to poor diarization and/or transcription. These calls correspond to 940 unique insurance cases, and thus 940 data points. The distribution of data points can be seen in Table 4.1

Class	Number of cases	Fraction
Legitimate	286	30.4%
Fraudulent	654	69.6%

Table 4.1: The table shows the distribution of Legitimate and Fraudulent cases in the audio data. It shows both the number of cases and its fraction of the whole data set.

TF-IDF with logistic regression

Before running TF-IDF, numbers and stop words [21], which are very common words and would dilute the TF-IDF scores if included, were removed from all documents. It seemed unreasonable to include words that occur very rarely, hence, we chose to only include words present in at least 1% of the calls. This resulted in 3218 unique words. The resulting TF-IDF matrix was used together with logistic regression, which resulted in poor results. Because our data set is quite unbalanced, we decided to apply weights to the loss function, to punish miss-labeling the smaller class. This worked well and we continued to do so for all other models on the audio data.

A usual procedure when using logistic regression models is to remove the least significant parameters. All parameters with an absolute value of less than 0.9 for the β-estimate were removed. This resulted in a model with 8 parameters instead.

The results of both unweighted and weighted logistics regression can be seen in Table 4.2. Furthermore, estimates of resulting β-parameters can be seen in Figure 4.1, and the eight remaining β-parameters after reduction can be seen in Figure 4.2.

	coefficient		coefficient
fall	1.104802	eh	-0.665058
försäkring	1.076259	kollega	-0.671173
säga	1.049018	super	-0.693811
måste	0.916142	j	-0.770546
minns	0.911314	kolla	-0.781246
stämmer	0.857942	s	-0.790885
inga	0.853255	foten	-0.896303
helt	0.813044	mobilen	-1.000752
rätt	0.780200	polisen	-1.106709
vill	0.762294	kvitto	-1.169266

Figure 4.1: Features with the ten highest and ten lowest values of β-parameter estimates (3128 features total), from weighted logistic Regression on TF-IDF matrix for audio data.

	coefficient	Lower_CI	Upper_CI
försäkring	1.582539	-1.611898	4.776977
säga	1.539968	-2.014615	5.094551
fall	1.384426	-3.348390	6.117241
minns	1.304675	-1.995348	4.604698
måste	1.267893	-2.943821	5.479606
mobilen	-1.197655	-5.314789	2.919479
polisen	-1.203878	-3.882325	1.474570
kvitto	-1.234667	-4.352019	1.882685

Figure 4.2: Estimates and confidence interval for the eight most important features for weighted logistic regression on TF-IDF matrix for audio data.

Model: TF-IDF	Class	Precision		Recall		F1-Score	
Logistic regression							
3218 features	Legit	0.0	±0.0	0.0	±0.0	0.0	±0.0
	Fraud	0.695	±0.032	0.993	±0.005	0.815	±0.023
	Macro avg	0.347	±0.018	0.495	±0.003	0.408	±0.011
Weighted logistic regression							
3218 features	Legit	0.367	±0.044	0.433	±0.044	0.393	±0.036
	Fraud	0.731	±0.039	0.672	±0.024	0.7	±0.025
	Macro avg	0.549	±0.025	0.553	±0.027	0.546	±0.026
8 features	Legit	0.403	±0.043	0.653	±0.043	0.497	±0.041
	Fraud	0.791	±0.035	0.579	±0.031	0.667	±0.029
	Macro avg	0.597	±0.024	0.616	±0.026	0.582	±0.026

Table 4.2: Classification report for TF-IDF model with logistic regression, on audio data.

Hierarchical BERT

Transformers have proven to perform well on a wide range of NLP-tasks. Considering the length of our sequences (number of words per data point) we could not use a standard Transformer, like BERT straight away, since it is capped at a sequence length of 512. There are some Transformers adapted to handle longer sequences, but these models were excluded since none of them are pre-trained in Swedish to our knowledge. It is possible to re-train a "normal" transformer into the architecture of a "longformer". Sagen discusses this process in [18], where Sagen re-trains a multilingual short model into a long one and then lets the model process long sequences in other languages than English. This approach was excluded mainly for two reasons. First off, the conclusion of Sagen's thesis was that this approach did not give a satisfying result. Secondly, this would require a lot of resources to re-train a model. Instead, we chose to use a hierarchical approach, which splits the input sequence into chunks that are shorter than 512 and can therefore fit into BERT. This does not require us to

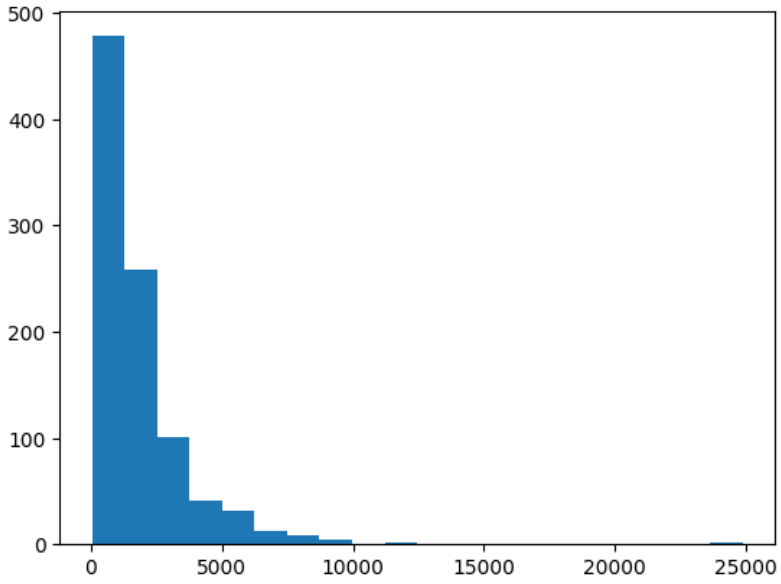

Figure 4.3: Plot of a histogram for the tokenized sequence length of the audio data. The number of bins is 20.

put in any extra effort in training a new model from scratch, and it has been proven to perform well on longer sequences [23].

There are some choices to be done for the hierarchical approach. One could either choose to fine-tune the parameters of BERT for the specific task on each chunk or just use the pre-trained parameters. The authors of [23] achieve better results when they fine-tune their BERT-model. However, fine-tuning BERT on every chunk might lead to feeding the model with incorrect labels, since our audio data have one label for the whole conversation (see Figure 4.4 for a graphical overview of this issue). This means that we either consider the whole conversation to be fraudulent or the whole conversation to be legit. In a fraudulent case, it is likely that the customer is only being deceptive in certain parts of the conversation. If we were to fine-tune BERT on every chunk we would likely have many sequences labeled as fraud where the customer is actually being truthful.

Instead, we decided to use one of the outputs of BERT, which is a token representation of the input sequence called the CLS-embedding (Classification-embedding). In short, it can be explained as an embedding containing the information for the entire input sequence. Each chunk is fed into BERT and their respective CLS-embedding is then concatenated and can be used as an input to some other model, see 4.5. Despite using this approach, we still excluded the data points with the longest tokenized sequence length for performance purposes. A histogram of tokenized sequence lengths can be seen in Figure 4.3. The cutoff was decided to be at 8160, which excluded 14 data points.

For the classification, we decided to train an LSTM-network and use logistic regression. The LSTM architecture is known for working well with sequential data and logistic regression is being used for comparison to the other models. The LSTM-

network had a single LSTM-layer with 16 units, optimizer Adam, learning rate 0.001, Categorical Cross-Entropy (CCE) as loss function, dropout of 0.4, batch size 8, and ran for 25 epochs. The results can be seen in Table 4.3.

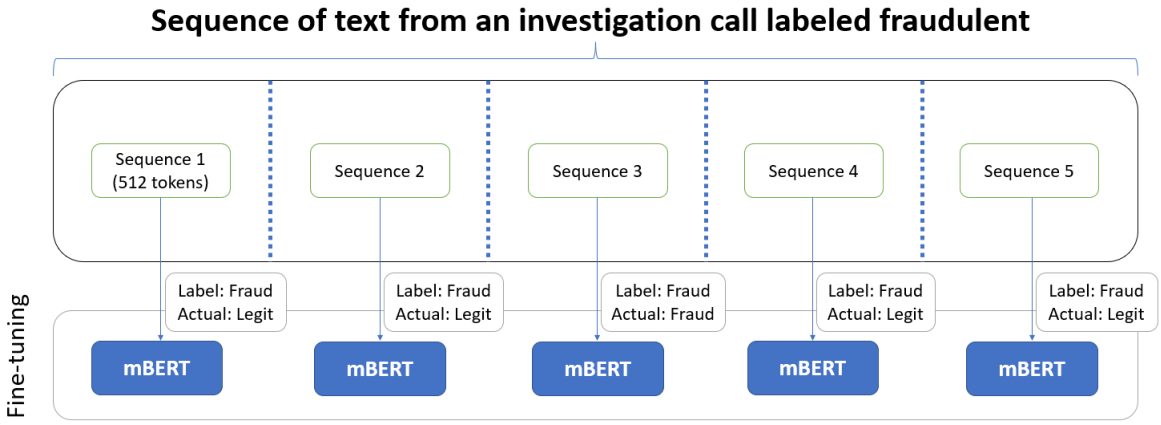

Figure 4.4: An overview of the possible fine-tuning of a data point with a longer sequence length than 512. The call is labeled as fraudulent, but the deception is (possibly) only present in one of the sequences. Since we do not know where the deception is taking place, the whole call is labeled as fraudulent. If this is the case, BERT is fine-tuned with misleading labels.

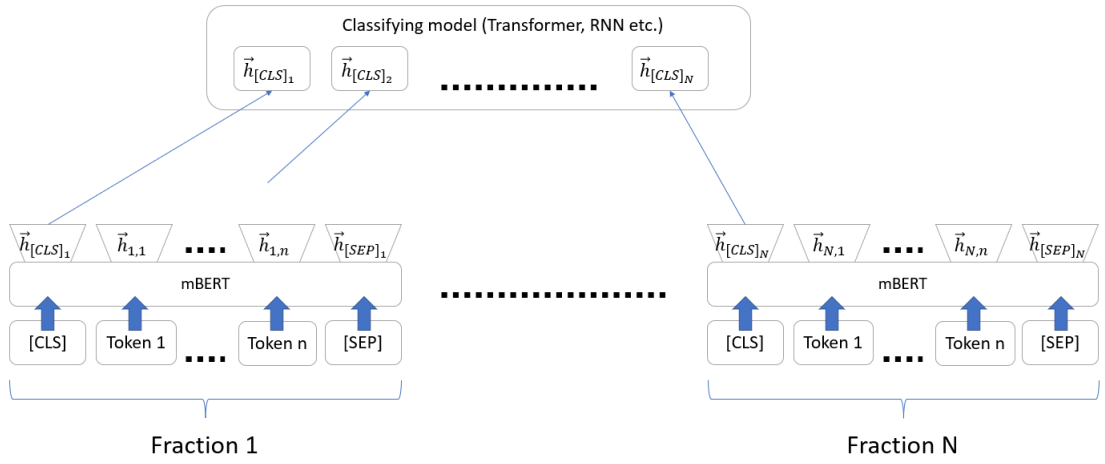

Figure 4.5: An overview of the architecture of a hierarchical approach using BERT. Fraction n represents the first fraction of a text in a data point.

Model: BERT	Class	Precision		Recall		F1-Score	
Logistic regression	Legit	0.385	±0.054	0.354	±0.06	0.365	±0.048
	Fraud	0.727	±0.033	0.757	±0.03	0.741	±0.02
	Macro avg	0.556	±0.026	0.556	±0.024	0.553	±0.025
LSTM-network	Legit	0.386	±0.043	0.520	±0.078	0.435	±0.040
	Fraud	0.751	±0.038	0.635	±0.049	0.683	±0.028
	Macro avg	0.568	±0.026	0.577	±0.032	0.560	±0.026

Table 4.3: Classification report for logistic regression and an LSTM-network trained on the extracted CLS-tokens from BERT, on audio data.

While having discarded the approach of fine-tuning every chunk, due to the risk of miss-labeling, we still wanted to see if fine-tuning on our own data could result in an improvement. We decided to test if fraud could be more present in the beginning, middle, and end of a call, by fine-tuning and predicting those parts in isolation. The training was done with optimizer Adam, learning rate 0.001, CCE as loss function, batch size of 16, and ran for four epochs. The results can be seen in Table 4.4.

Model: BERT	Class	Precision		Recall		F1-Score	
First chunk	Legit	0.214	±0.122	0.163	±0.131	0.173	±0.122
	Fraud	0.705	±0.020	0.856	±0.090	0.767	±0.040
	Macro avg	0.460	±0.059	0.510	±0.030	0.471	±0.045
Mid chunk	Legit	0.266	±0.102	0.287	±0.130	0.269	±0.106
	Fraud	0.714	±0.033	0.756	±0.96	0.724	±0.038
	Macro avg	0.489	±0.057	0.521	±0.022	0.498	±0.037
Last chunk	Legit	0.183	±0.131	0.162	±0.114	0.167	±0.116
	Fraud	0.702	±0.036	0.857	±0.090	0.764	±0.033
	Macro avg	0.443	±0.079	0.51	±0.033	0.466	±0.055

Table 4.4: Classification report for BERT fine-tuned on different chunks of audio data with a final linear layer as classifier.

word2vec

For the second model, we wanted an NLP-model with a different way of creating word embeddings, compared to BERT. We decided to go with a model that creates a vector space where words that are related to each other are located close to each other. These models are pre-trained completely different to BERT, and do not use the Transformer architecture. There are a few of these models available, but we chose word2vec, for the simple reason that it was the only model that we found pre-trained on the Swedish language. We used these pre-trained weights and combined them with our own vocabulary to update the vector space to fit our task. The model was built with vector size=100, window=10, min_count=1, workers=8, and sg = 1 and trained for five epochs. This model could then be used to create word embeddings for our texts which we used in a k-nearest neighbor and a logistic regression. The k-nearest neighbor uses $k = 3$ neighbors and weighted distance. Weighted distance means that closer neighbors have a higher impact on the predicted result. The results can be seen in Table 4.5.

Model: word2vec	Class	Precision		Recall		F1-Score	
Logistic regression	Legit	0.361	±0.036	0.505	±0.074	0.417	±0.045
	Fraud	0.735	±0.054	0.610	±0.040	0.667	±0.041
	Macro avg	0.548	±0.037	0.556	±0.047	0.541	±0.038
K-nearest neighbour	Legit	0.338	±0.062	0.217	±0.051	0.261	±0.053
	Fraud	0.703	±0.042	0.814	±0.037	0.753	±0.034
	Macro avg	0.52	±0.044	0.516	±0.032	0.507	±0.039

Table 4.5: Classification report for word2vec with logistic regression and k-nearest neighbours as classifiers, on audio data.

Linguistic analysis

We decided to go for a somewhat simpler and more understandable model as the third one. Research like [20] and [8] suggests that deception can be linked to the usage of certain word classes. There are two arguments for this choice. First off, it is different from the two previously chosen models. Secondly, it is easier to understand this approach compared to the previous models. With the use of UDPipe, we extracted different linguistic characteristics for each word in a call, such as nouns, verbs, and adjectives. One extraction tagged the words with the universal Part-of-Speech tags[31] (16 classes) and the other also included the syntactic dependency relations (39 classes), which is the relation between words in a sentence. The total occurrences were counted for each class for each call and divided by the total number of words in that call. This was then used as input to a k-nearest neighbor classifier and a logistic regression. The k-nearest neighbor uses $k = 3$ neighbors and weighted distance. The results can be seen in Table 4.6.

Model: Linguistic	Class	Precision		Recall		F1-Score	
Logistic regression							
16 classes	Legit	0.337	±0.049	0.483	±0.095	0.395	±0.064
	Fraud	0.728	±0.042	0.590	±0.048	0.649	±0.039
	Macro avg	0.533	±0.042	0.538	±0.048	0.522	±0.042
55 classes	Legit	0.356	±0.026	0.505	±0.051	0.414	±0.029
	Fraud	0.733	±0.037	0.599	±0.036	0.658	±0.033
	Macro avg	0.543	±0.028	0.552	±0.034	0.536	±0.030
K-nearest neighbour							
16 classes	Legit	0.334	±0.083	0.188	±0.054	0.237	±0.057
	Fraud	0.703	±0.037	0.836	±0.029	0.763	±0.023
	Macro avg	0.52	±0.046	0.512	±0.033	0.499	±0.036
55 classes	Legit	0.256	±0.061	0.147	±0.022	0.183	±0.03
	Fraud	0.683	±0.031	0.805	±0.037	0.737	±0.02
	Macro avg	0.469	±0.027	0.477	±0.021	0.459	±0.02

Table 4.6: Classification report for linguistic analysis with logistic regression and k-nearest neighbours, on audio data.

4.2 Text-form data

4.2.1 Pre-processing of the data

For the text-form data, there was not as much pre-processing needed. As mentioned before, we want to build our models on only text stemming from the customer. Thus, all questions were removed from the Questions & Answers data. Furthermore, the free text and Questions & Answers were merged together. We also checked for insurance claims with an entry in both sets, but there were none.

4.2.2 NLP-techniques and results

The text-form data contains 5385 data points in total. The distribution of legitimate and fraudulent cases can be seen in Table 4.7

Class	Number of cases	Fraction
Legitimate	3986	74.0%
Fraudulent	1399	26.0%

Table 4.7: The table shows the distribution of legitimate and fraudulent cases in the text-form data.

In principle, the NLP-techniques used are the same for the text-form data as for the audio data. However, some things are adjusted for the text-form data.

For all models built on the text-form data, we chose to have $k = 5$ folds for the k-fold cross-validation, meaning that each fold contains 1077 data points. The reason we chose this number to be lower than for the audio data was because this data set has more data points. Thus, we believed the training data in each validation fold to be big enough with only 5 folds.

Just like for the audio data, all results will be presented as a classification report with mean and confidence intervals for all measures.

TF-IDF

First off, the text-form data was evaluated on the naive model. Stop words and numbers were excluded from the Term-Frequency. This resulted in 19009 unique words. This is a lot of features, especially since the data set contains 5385 data points. We chose to only include words present in at least 1% of the documents in the corpus. This gave 328 unique words. Just as for the audio data, the text-form data is also quite unbalanced, and applying weights to the loss function for all models turned out to improve performance in this case as well. The results for both logistic regression and weighted logistic regression can be seen in Table 4.8.

All parameters with an absolute value of less than $1,2$ for the β-estimate were removed. This resulted in a model with 20 parameters instead. The resulting classification report can also be seen in Table 4.8. Furthermore, the estimated β-values can be seen in Figure 4.6.

Model: TF-IDF	Class	Precision		Recall		F1-Score	
Logistic regression							
328 features	Legit	0.748	±0.018	0.970	±0.100	0.844	±0.009
	Fraud	0.442	±0.048	0.070	±0.020	0.118	±0.032
	Macro avg	0.594	±0.018	0.520	±0.007	0.480	±0.012
Weighted logistic regression							
328 features	Legit	0.812	±0.011	0.590	±0.029	0.682	±0.019
	Fraud	0.342	±0.025	0.610	±0.090	0.438	±0.027
	Macro avg	0.576	±0.015	0.598	±0.022	0.560	±0.023
20 features	Legit	0.884	±0.018	0.330	±0.014	0.482	±0.011
	Fraud	0.314	±0.020	0.878	±0.018	0.464	±0.024
	Macro avg	0.600	±0.010	0.606	±0.011	0.472	±0.018

Table 4.8: Classification report for TF-IDF with unweighted and weighted logistic regression, on text-form data.

	coefficient	Lower_CI	Upper_CI		coefficient	Lower_CI	Upper_CI
guld	2.045960	0.575875	3.516044	platsen	-1.445556	-2.884003	-0.007109
ta	1.699090	0.629349	2.768832	brand	-1.487404	-2.585520	-0.389288
källaren	1.623177	0.203679	3.042675	parkeringen	-1.493377	-2.687436	-0.299318
ligger	1.441436	-0.288400	3.171272	vägen	-1.546520	-2.402035	-0.691005
hej	1.436955	0.144329	2.729582	bak	-1.562773	-2.788451	-0.337095
barn	1.372638	0.206876	2.538399	diket	-1.570041	-2.766109	-0.373974
vattnet	1.360642	0.056210	2.665074	tre	-1.694261	-3.342516	-0.046007
dator	1.192767	0.210520	2.175013	bilar	-1.789942	-3.498776	-0.081108
handen	1.191379	0.021915	2.360842	bilen	-2.401224	-2.848025	-1.954424
lägenhet	1.149966	0.024324	2.275608	kört	-2.445161	-3.804428	-1.085893

Figure 4.6: Estimates and confidence intervals for the ten largest and ten smallest β-values from TF-IDF with weighted logistic regression, on text-form data.

BERT

The procedure for BERT was a bit different for the text-form data than the audio data. The length of the sequences in audio data had a big impact on what type of models we could use. For the text-form data, only 15 data points had a tokenized sequence length longer than 512 (which is the limitation for BERT). A histogram of the tokenized sequence lengths can be seen in Figure 4.7. Because of this, we saw no reason to use a Hierarchical BERT for the text-form data. Instead, we fine-tuned BERT with our labels. The training was done with optimizer AdamW, CCE as loss function, batch size of 16, and ran for four epochs. The resulting classification report can be seen in Table 4.9

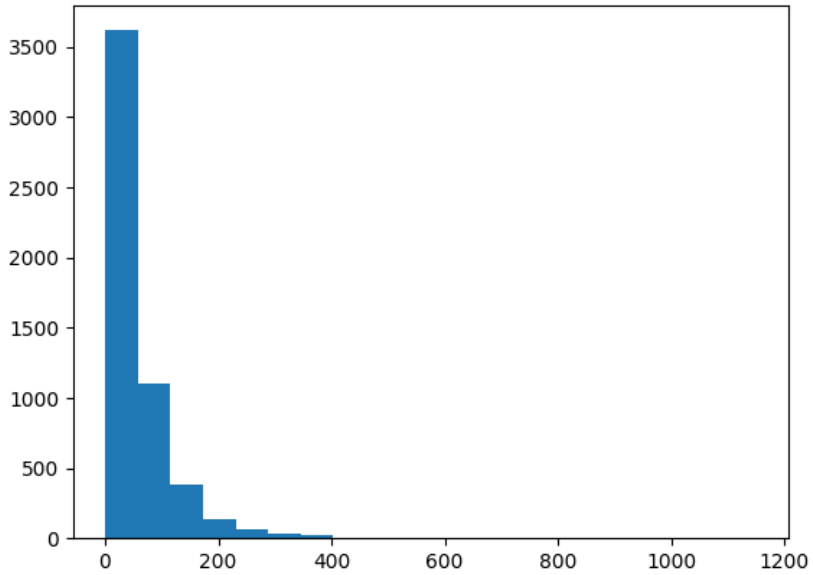

Figure 4.7: Plot of a histogram showing the distribution of tokenized sequence length of the text-form data. The number of bins is 20

Class	Precision		Recall		F1-Score	
Legit	0.824	±0.016	0.598	±0.019	0.692	±0.013
Fraud	0.358	±0.011	0.632	±0.037	0.454	±0.013
Macro avg	0.590	±0.01	0.616	±0.015	0.574	±0.009

Table 4.9: Classification report for a fine-tuned BERT with a final linear layer as classifier.

word2vec

word2vec was carried out in the same way for the text-form data as for the audio data. For word2vec, stop words were also excluded. The model was built with vector size=100, window=10, min_count=1, workers=8, and sg = 1. It was then trained for five epochs. The resulting embeddings were used as input data for a k-nearest neighbors classifier and logistic regression. The K-nearest neighbor uses $k = 3$ neighbors and weighted distance. All results can be seen in Table 4.10

Model: word2vec	Class	Precision		Recall		F1-Score	
Weighted logistic regression	Legit	0.824	±0.015	0.564	±0.016	0.672	±0.013
	Fraud	0.346	±0.034	0.652	±0.054	0.452	±0.040
	Macro avg	0.584	±0.019	0.608	±0.025	0.560	±0.021
K-nearest neighbour	Legit	0.754	±0.019	0.888	±0.004	0.816	±0.011
	Fraud	0.344	±0.019	0.168	±0.03	0.224	±0.026
	Macro avg	0.548	±0.018	0.528	±0.011	0.522	±0.019

Table 4.10: Classification report for text-form data with word2vec and random forest as classifiers.

Linguistic Analysis

The third technique, Linguistic Analysis, was carried out in the same manner as for audio data. It was used together with K-nearest neighbor and logistic regression. The K-nearest neighbor uses $k = 3$ neighbors and weighted distance. The results can be seen in Table 4.11.

Model: Linguistic	Class	Precision		Recall		F1-Score	
Logistic regression							
16 classes	Legit	0.764	±0.018	0.556	±0.015	0.646	±0.015
	Fraud	0.290	±0.026	0.516	±0.037	0.370	±0.032
	Macro avg	0.526	±0.016	0.536	±0.025	0.508	±0.018
55 classes	Legit	0.766	±0.009	0.584	±0.019	0.662	±0.016
	Fraud	0.292	±0.022	0.486	±0.026	0.366	±0.020
	Macro avg	0.528	±0.011	0.536	±0.009	0.512	±0.013
K-nearest neighbour							
16 classes	Legit	0.748	±0.016	0.838	±0.013	0.79	±0.012
	Fraud	0.294	±0.036	0.196	±0.028	0.236	±0.035
	Macro avg	0.522	±0.023	0.518	±0.019	0.512	±0.019
55 classes	Legit	0.748	±0.015	0.83	±0.007	0.786	±0.005
	Fraud	0.298	±0.016	0.206	±0.011	0.24	±0.01
	Macro avg	0.522	±0.004	0.518	±0.004	0.514	±0.005

Table 4.11: Classification report for linguistic analysis with logistic regression and k-nearest neighbours as classifiers.

4.3 Summary of Results

To get an overview of all results, the best F1-score for all NLP-architectures for both data sets will be presented in Table 4.12. For example, for linguistic analysis for audio data, the best F1-score out of logistic regression with 16 and 55 classes and K-nearest neighbor with 16 and 55 classes will be presented in the summary table.

NLP-Technique	Version/Classifyer	Macro F1
Audio data		
TF-IDF (8 features)	logistic regression	0.582 ±0.026
BERT	CLS with LSTM	0.560 ±0.026
word2vec	logistic regression	0.541 ±0.038
Linguistic (55 classes)	logistic regression	0.536 ±0.030
Text-form data		
TF-IDF (328 features)	logistic regression	0.560 ±0.023
BERT	BERT fine-tune	0.574 ±0.009
word2vec	logistic regression	0.560 ±0.021
Linguistic (55 features)	k-nearest neighbor	0.514 ±0.005

Table 4.12: Summary of all results. The table shows the best result for all NLP-techniques and for both data sets.

Chapter 5
Discussion

5.1 Results

Some of the methods show a small ability to generalize the data, with confidence intervals for Macro F1-score being strictly above 50 %. An F1-score of above 50% indicates that the model is learning something and that it is better than just guessing.

TF-IDF performed quite well for both the Audio data and the Text-form data. This seemed a bit surprising for a model that basically only counts words. However, there is a rather big risk that these positive results stem from a bias in the claim type. If we look at the most significant features, most of them are words that seem to be connected to one type of insurance case. For example "mobilen" (mobile phone), "bilen" (the car), and "guld" (gold) are probably only occurring in calls relating to certain claim types. Furthermore, none of the parameters for the Audio data, even in the reduced TF-IDF model, are significantly different from zero. For the Text-form data, some are, but when the logistic regression was done on only those parameters the F1-score decreased. For these two reasons, TF-IDF might not be that reliable despite having a Macro F1-score significantly over 50 %.

For the Audio data, a hierarchical BERT architecture was used. This was in turn combined with two different techniques for classifying, logistic regression and an LSTM network. The LSTM network performed best. One reason for the good performance of the LSTM network is probably its ability to capture relations between inputs far away from each other, which makes it useful for sequential data.

For the BERT model on audio data, it was fine-tuned for different parts of the audio data (beginning, middle, and end). This was done for two reasons. To see if it could be beneficial to fine-tune BERT and to see if there could be any conclusion drawn on where in a conversation a lie is being told. Furthermore, fine-tuning BERT has been done previously with good results. The mean of all Macro F1-scores is below 50%, suggesting that fine-tuning BERT for the audio data did not create a better model. Further, it indicates that for the Audio data, the BERT model couldn't predict lies better in a specific part of the data. This result was not surprising. Calls in audio data look very different and it is not unlikely that fraud takes place in the beginning of some calls and the end of some calls. Furthermore, this approach doesn't utilize all the data, which might be a reason that it can't model the data.

word2vec gave a Macro F1-score with a confidence slightly over 50% for both the Audio data and the Text-form data, when used together with a weighted logistic regression. This suggests that fraudulent sequences of text are placed somewhat closer to each other in their vector representations compared to legitimate ones.

The Linguistic analysis did not perform that well. The authors of [20] construct a somewhat similar model with good results. There are a few reasons that the Linguistic analysis performed in this thesis does not perform as well. One reason could that the information in the different word classes is not informative enough for the data at hand. The word classes used from UDpipe differ a bit from LIWC. Another reason could be the fact that the data is different in [20]. In that article, the subjects are asked to produce a lie in the form of an opinion that is not their own. How lies are told might be different between arguing for your opinion and telling an actual series of events. Furthermore, the subjects in [20] are told by a supervisor to lie, as opposed to initializing it themselves as in this project.

5.2 Imbalanced data

One difficulty throughout the project was the imbalanced data sets. This was handled by using weighted loss functions, nudging the models to predict the less common class more often. Another possible way of dealing with this is to downsample the larger of the two classes until the data set is balanced. This was not done mainly for one reason. The data sets are both rather small, and downsampling would mean sacrificing a lot of the available data.

5.2.1 Effects of Weighted Loss Function

Apart from increasing the Macro F1-score for all models it was used on, it also balances the other measures. The Recall for the most common class becomes lower whereas the Recall for the uncommon class is increased. Furthermore, the Precision for the common class is increased whereas the precision for the uncommon class is decreased. This is an expected result. With a weighted loss function, the loss function penalizes false predictions higher if it is on the less common class. Thus, the model will learn to predict the less common class more often, resulting in a higher recall for that class.

A more balanced distribution between Precision and Recall for the two classes is not necessarily desired. Failing to label one class could be more expensive than mislabeling the other class. In the context of this thesis, one could compare the cost of a false positive (legitimate claim classified as fraud) with a false negative (fraudulent claim classified as legitimate). The effect of these wrong labels would probably be: 1. the company pays insurance money they would not have needed to pay and 2. the company needs to re-investigate the legitimate claim and pay the (rightfully) insurance. It might be more expensive to reimburse a fraudulent claim than re-investigating a legitimate claim or vice versa. Because of this, the highest Macro F1-score might not always be the best model for the purpose even if it is the model that best generalizes to the data. Furthermore, the cost of paying a fraudulent claim vs re-investigating a legitimate one might differ depending on what type of case it is. For example, pay-

ing for a car reparation is more expensive than changing the screen on a telephone. With this in mind, it could be useful to create separate models for different insurance types, with different weights for the loss function.

5.3 Quality of transcription and diarization

One factor that could have affected the result is the quality of transcriptions and diarizaiton. Even if all transcriptions were quickly reviewed there were still transcriptions and diarizaitons that were not perfect. Furthermore, bad transcriptions and/or diarizations forced data points to be removed. This leads to two issues for the modeling. First off, it means fewer data points and in turn less data to train on. Secondly, the data contains errors such as lousy transcribed parts and remaining text from the company representative. With a higher quality of transcription and diarization the result could possibly have been more reliable. However, perfect data is rarely (never) the case when building these types of models.

5.4 Fraud vs Lie

One important thing to note about this project is that all modeling is based on labels from the insurance company. This means that the ground truth, per se, is not whether someone is lying or not, but in fact, if the insurance company assesses this claim to be fraudulent or not. Furthermore, fraudulent insurance claims can look different. A fraudulent claim could vary from a customer already being reimbursed by another insurance company to a customer lying about jewelry being stolen.

The distinction between a fraudulent claim and a lie might seem small, but it is still a difference to note. The models in this project should not be seen as lie detectors, but rather as fraud detectors.

5.5 Limitation of Swedish models

The pool of available NLP-models is a lot smaller for Swedish compared to English models. For example, there is no Longformer pre-trained in Swedish. This excluded some possible techniques for the modeling. Furthermore, most models are trained on much more data in English compared to Swedish. This gives English models a better condition to perform well. On the other hand, most of the models are pre-trained on text from Wikipedia or news. It is not certain that pre-training models on more of these types of texts will improve their performance on fraud classification.

5.6 Further research

Machine Learning and NLP is a fast-growing field. New and better models are trained and produced at a fast pace. Redoing a similar project like this in a few years could give more promising results.

Due to the sensitive nature of the data, online tools have been excluded. A large and strong model like GPT-4 could possibly perform better than the ones tested in this project.

Another interesting take could be to combine the results and models from this project with the ones from Steneld's [10]. In other words, combining the tonality of speech with what is actually said. It might be challenging to combine these techniques, but it should be possible.

Chapter 6
Conclusion

There were two questions that we wanted to answer with this master thesis.

- Is it possible to predict insurance fraud with NLP-techniques from conversation recordings between a customer and company representative or text data entered by the customer either in free text form or from Question & Answers?
- How do different NLP-techniques compare to each other when it comes to predicting fraud?

We analyzed three different NLP-techniques: BERT, word2vec and linguistic analysis, and also included TF-IDF as our naive model. To answer the second question we look at our results. For the audio data, TF-IDF together with logistic regression scored the highest Macro F1-score of 58.2% $\pm$2.6%. For the text data, our fine-tuned BERT model scored the highest Macro F1-score of 57.4% $\pm$0.9%. Worst performed the BERT model that was fine-tuned on the first, middle, and last part of each call in the audio data, which could be expected as it loses a large part of the context.

To answer the first question is much harder. Being able to predict insurance fraud based on what is being said or written is a really complex problem. No one has really created a consistently working lie detector before, and there might be a reason for that. While we aren't exactly trying to predict truths and lies, as a lie detector would, the task of detecting insurance fraud is still very hard. Our models gave predictions that were better than tossing a coin, so they were actually able to learn something from the data, but what that something is, is hard to tell. The reason that TF-IDF and word2vec perform better than guessing could be due to bias in the data set as discussed in the previous chapter. Certain words might appear more often in types of insurance cases where fraud is more common, which would "fool" the model. Due to the "black-box" nature of deep learning, we can't really look at what models like BERT and LSTM learned, and therefore it is harder to determine if it actually learned to predict fraud or if it learned something else. Theoretically, they should be able to capture more complex linguistic traits in a text than the other models, but this can only be assured through more testing.

We believe that there could be future potential in using these kinds of techniques to detect insurance fraud. Complex Large Language Models (LLM), such as BERT and Chat-GPT, have become insanely good in the last months. This is mainly because

of the enormous amount of data it has been trained on. With the right amount of computing power and a large enough data set, it is possible that a model could be developed to be able to detect insurance fraud with much better precision than we were able to in this project.

Bibliography

[1] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. "WhisperX: Time-Accurate Speech Transcription of Long-Form Audio". In: *INTERSPEECH 2023* (2023).

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].

[3] Jia-Wei Chang, Neil Yen, and Jason Hung. "Design of a NLP-empowered finance fraud awareness model: the anti-fraud chatbot for fraud detection and fraud classification as an instance". In: *Journal of Ambient Intelligence and Humanized Computing* 13 (Mar. 2022).

[4] Padraig Cunningham and Sarah Jane Delany. "k-Nearest Neighbour Classifiers - A Tutorial". In: *ACM Computing Surveys* 54.6 (July 2021), pp. 1–25.

[5] DeepLearning.AI. *A complete guide to Natural Language Processing*. URL: https://www.deeplearning.ai/resources/natural-language-processing/. (accessed: 8-May-2023).

[6] Google for developers. *Machine Learning Glossary*. URL: https://developers.google.com/machine-learning/glossary/. (accessed: 29-May-2023).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[8] Nicholas D. Duran, Charles Hall, Philip M. McCarthy, and Danielle S. McNamara. "The linguistic correlates of conversational deception: Comparing natural language processing technologies". In: *Applied Psycholinguistics* 31.3 (2010), pp. 439–462.

[9] Larry Hardesty. *Explained: Neural networks*. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. (accessed: 11-May-2023).

[10] Henrik Steneld. *Estimating the risk of insurance fraud based on tonal analysis*. eng. Student Paper. Lund University, 2022.

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80.

[12] tf-idf. *tf-idf — Wikipedia, The Free Encyclopedia*. 2023. URL: https://en.wikipedia.org/wiki/Tf-idf. (accessed: 15-March-2023).

[13] Nathalie Jeans. *How I Classified Images With Recurrent Neural Networks*. URL: https://medium.com/@nathaliejeans/how-i-classified-images-with-recurrent-neural-networks-28eb4b57fc79. (accessed: 30-may-2023).

[14] Hoa Le. *Paper Dissected and Recap #4 : which BERT for long text ?* URL: https://lethienhoablog.wordpress.com/2020/11/19/paper-dissected-and-recap-4-which-bert-for-long-text/. (accessed: 15-March-2023).

[15] Kenneth Leung. *Micro, Macro Weighted Averages of F1 Score, Clearly Explained*. URL: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f. (accessed: 29-May-2023).

[16] Lie Detection. *Lie Detection — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-March-2023]. 2023. URL: https://en.wikipedia.org/wiki/Lie_detection.

[17] Maher Maalouf. "Logistic regression in data analysis: An overview". In: *International Journal of Data Analysis Techniques and Strategies* 3 (July 2011), pp. 281–299.

[18] Markus Sagen. *Large-Context Question Answering with Cross-Lingual Transfer*. eng. Student Paper. Uppsala University, 2021.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[20] Matthew L. Newman, James W. Pennebaker, Diane S. Berry, and Jane M. Richards. "Lying Words: Predicting Deception from Linguistic Styles". In: *Personality and Social Psychology Bulletin* 29.5 (2003), pp. 665–675.

[21] NLTK. *Documentation*. URL: https://www.nltk.org/api/nltk.text.html?highlight=stopwords. (accessed: 18-April-2023).

[22] NVIDIA. *User-guide for NeMo Speaker Diarization*. 2023. URL: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/speaker_diarization/intro.html#. (accessed: 09-June-2023).

[23] Raghavendra Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. *Hierarchical Transformers for Long Document Classification*. 2019. arXiv: 1910.10781 [cs.CL].

[24] James Pennebaker, Martha Francis, and Roger Booth. "Linguistic inquiry and word count (LIWC)". In: (Jan. 1999).

[25] Rukshan Pramoditha. *k-fold cross-validation explained in plain English*. URL: https://towardsdatascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0bc0. (accessed: 29-may-2023).

[26] Hirad Rezaei, Alireza Amjadian, Mohammad Sebt, Reza Askari, and Abolfazl Gharaei. "An ensemble method of the machine learning to prognosticate the gastric cancer". In: *Annals of Operations Research* (Sept. 2022).

[27] Yutaka Sasaki. "The truth of the F-measure". In: *Teach Tutor Mater* (Jan. 2007).

[28] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. arXiv: 1912.05911 [cs.LG].

[29] Bryan Shalloway. *Weighting Confusion Matrices by Outcomes and Observations*. URL: https://www.bryanshalloway.com/2020/12/08/weighting-classification-outcomes/. (accessed: 29-May-2023).

[30] Jiri Stastny, Vladislav Skorpil, and Jiří Fejfar. "Visualization of Uncertainty in LANDSAT Classification Process". In: Jan. 2014.

[31] Universal Dependencies. *Universal POS tags*. 2022. URL: https://universaldependencies.org/u/pos/. (accessed: 10-June-2023).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[33] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. *Big Bird: Transformers for Longer Sequences*. 2021. arXiv: 2007.14062 [cs.LG].

Master's Theses in Mathematical Sciences 2023:E64
ISSN 1404-6342

LUTFMS-3489-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/