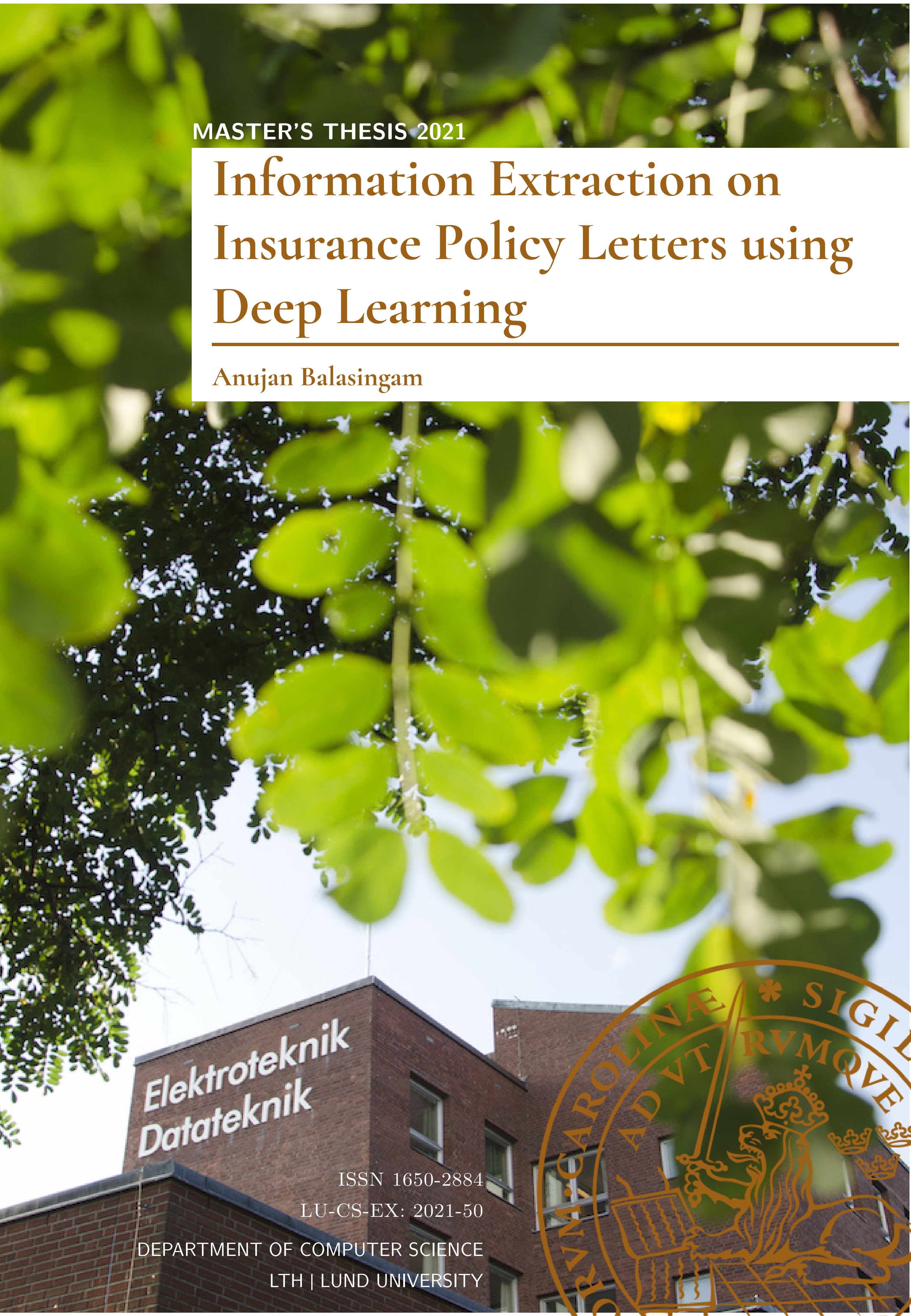
MASTER'S THESIS 2021

Information Extraction on Insurance Policy Letters using Deep Learning

Anujan Balasingam



ISSN 1650-2884
LU-CS-EX: 2021-50

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-50

Information Extraction on Insurance Policy Letters using Deep Learning

Anujan Balasingam

Information Extraction on Insurance Policy Letters using Deep Learning

Anujan Balasingam

dat14aba@student.lu.se

December 3, 2021

Master's thesis work carried out at Söderberg & Partners AB.

Supervisors: Marcus Klang, marcus.klang@cs.lth.se
Robert Karlsson, robert.karlsson@soderbergpartners.com

Examiner: Elin Anna-Topp, Elin_Anna.Topp@cs.lth.se

Abstract

Managing business documents can be a daunting task, especially when this is done manually and the volume is high. These documents, insurance documents in our case, hold critical business information that needs to be extracted, in this thesis we are going to research how it can be done automatically with deep learning techniques. We are using a multinomial logistic regression classifier as our baseline model and experimenting with different BiLSTM architectures for our neural models. We train the different classifiers on data we gathered from the databases of Söderberg & Partners AB. The highest score in Accuracy of the files, which is our strictest evaluation method, on previously seen templates is 0.728 ± 0.007. On unseen templates, the score was 0.0. The results show that simple models can be effective at extracting information from different templates of documents that they have been trained on but that it needs more fine-tuning to improve generalization.

Keywords: Information Extraction, Deep Learning, Natural Language Processing, Rule- and pattern based systems, Neural Networks

Acknowledgements

First of I want to thank my supervisor Marcus Klang for all the help and ideas I received during the whole process of writing my thesis. This thesis would also not be possible without the resources from Söderberg & Partners AB in the form of hardware and data, special thanks to Robert Karlsson for providing access to virtual machines and computers along with help on many other smaller issues.

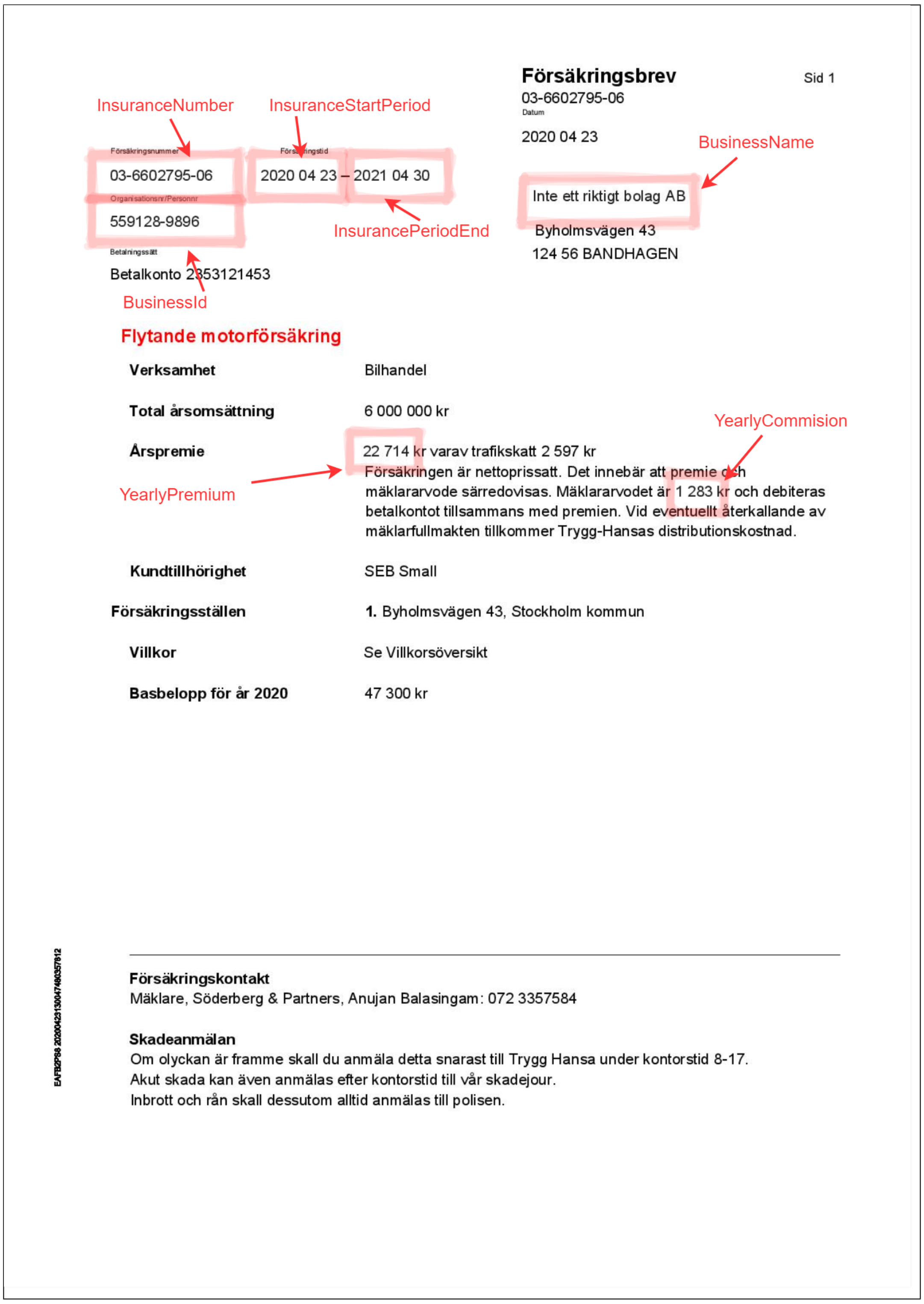
Contents

Chapter 1

Introduction

Business documents are important artifacts that enable trade between two or several parties. These business documents are for instance invoices, insurance documents, etc. that contain information about payment amount, dates, signatures, and so on. The documents are usually in paper form or in a digital format such as Portable Document Format (PDF). Companies extract information from these documents and store them within IT systems to keep track of their business agreements. The extraction is performed by the employees manually and entered into the IT system. As the volume of documents increases over time the traditional method of manual processing becomes outdated, it is a highly mundane and labour-intensive task that is also prone to human error. By letting machines take care of this process the employees can focus more on other tasks that could help the company grow. However, it is still a challenge to achieve full automation of information extraction from documents and is still an open problem.

The automation process can be described in the following way: Convert the document in Figure 1.1, which can be in an image format by scanning the physical copy or as a PDF, to a structured output seen in Figure 1.2.


InsuranceNumber
InsuranceStartPeriod
BusinessName
InsurancePeriodEnd
BusinessId
YearlyCommision
YearlyPremium

Försäkringsbrev Sid 1

03-6602795-06

Datum

2020 04 23

Försäkringsnummer

03-6602795-06

Försäkringstid

2020 04 23 – 2021 04 30

Organisationsnr/Personnr

559128-9896

Betalningssätt

Betalkonto 2353121453

Inte ett riktigt bolag AB

Byholmsvägen 43

124 56 BANDHAGEN

Flytande motorförsäkring

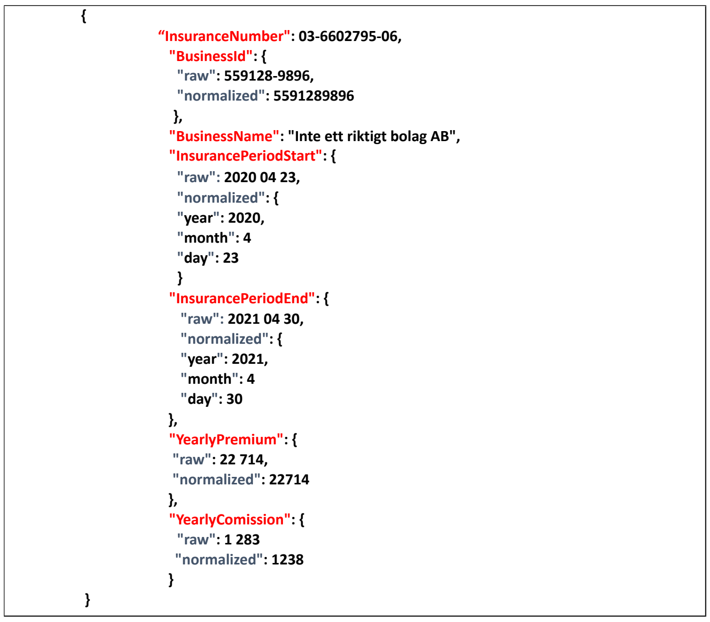
Verksamhet	Bilhandel
Total årsomsättning	6 000 000 kr
Årspremie	22 714 kr varav trafikskatt 2 597 kr Försäkringen är nettoprissatt. Det innebär att premie och mäklararvode särredovisas. Mäklararvodet är 1 283 kr och debiteras betalkontot tillsammans med premien. Vid eventuellt återkallande av mäklarfullmakten tillkommer Trygg-Hansas distributionskostnad.
Kundtillhörighet	SEB Small
Försäkringsställen	**1.** Byholmsvägen 43, Stockholm kommun
Villkor	Se Villkorsöversikt
Basbelopp för år 2020	47 300 kr

Försäkringskontakt
Mäklare, Söderberg & Partners, Anujan Balasingam: 072 3357584

Skadeanmälan
Om olyckan är framme skall du anmäla detta snarast till Trygg Hansa under kontorstid 8-17.
Akut skada kan även anmälas efter kontorstid till vår skadejour.
Inbrott och rån skall dessutom alltid anmälas till polisen.

EAFB2P88 2020042315004748035781 2

Figure 1.1: Insurance policy letter in a PDF



```
{
        "InsuranceNumber": 03-6602795-06,
         "BusinessId": {
          "raw": 559128-9896,
          "normalized": 5591289896
         },
         "BusinessName": "Inte ett riktigt bolag AB",
         "InsurancePeriodStart": {
          "raw": 2020 04 23,
          "normalized": {
          "year": 2020,
          "month": 4
          "day": 23
          }
         "InsurancePeriodEnd": {
           "raw": 2021 04 30,
           "normalized": {
           "year": 2021,
           "month": 4
           "day": 30
         },
         "YearlyPremium": {
          "raw": 22 714,
          "normalized": 22714
         },
         "YearlyComission": {
          "raw": 1 283
          "normalized": 1238
         }
 }
```

Figure 1.2: The PDF documents structured counterpart

The reason why it has to be in such a format is that the IT systems require a specific structure, it is now machine-readable by arranging the fields to a certain standard and structuring them in key-value format. There are arguments for PDF documents being digital already, sometimes referred to as semi-structured documents, this is not necessarily wrong but to a computer, it is still unstructured text and images. The structured counterpart seen in Figure 1.2 can now be further processed within IT systems without requiring any manual input from any human operator.

What we mean by semi-structured data in this context is that it does not conform with formal structures of a data model that is associated with relational databases for instance, but that it contains tags or markers that separate semantic elements and have fields of data [Sukanya and Biruntha, 2012]. The term unstructured data is a reference to data that doesn't have a predefined data model or fit relational database tables, it usually is text-heavy but may contain data such as dates and numbers that we are interested in. [Sint et al., 2009]. In contrast to these two data forms is the structured data counterpart, these have predefined data models and fit in database tables.

1.1 Motivation

Söderberg & Partners AB (S&P) has recently bought the software service "ABBYY FlexiCapture" which is a platform with the ability to classify and extract information from documents with a rule and pattern-based approach combined with Machine Learning. There are several services like this on the market that provide a solution for automatic extraction of information from documents, but most of these come with some form of limitation. These services work solely on previously observed templates of documents [Palm et al., 2017], a template is a distinct layout for certain documents that often are unique for each sender. The techniques that are used in these solutions consider keywords and the layouts of the templates, meaning the system relies on first being able to classify and divide the documents into different templates.

A human operator for this process is required to create complex rules that stem from the logical structures for each distinct template that arrives, this does require some specific domain knowledge. Ideally, the skills needed to create the rules for such templates should be kept low, especially when the occurrence of new documents may be a frequent event or the sample of documents of a certain type may be small. When extracting information from documents you focus on the fields that are important for the specific task. Let us take an invoice as an example, the most important information in such a document may be the premium, the due date, and the reference number to name a few. The rest of the text in the document does not add any value and therefore does not require any extraction as we can see in the example above.

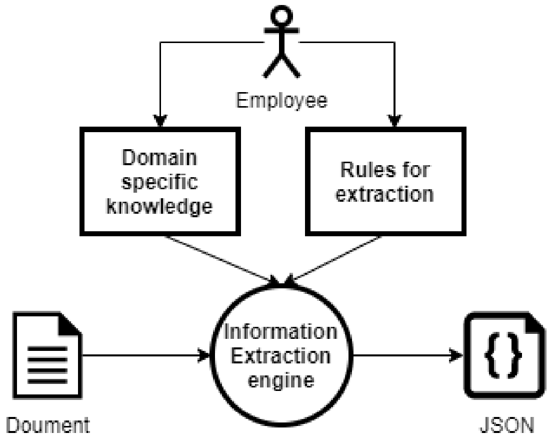


Figure 1.3: Rule and pattern-based system

These solutions that require such manual configuration of complex rules are built on conventional Machine Learning techniques that are not able to simply process the data in its raw form. In recent years another area in Machine Learning has been making major advances in fields like image recognition, speech recognition and has also shown extremely promising results in solving Natural Language Processing (NLP) problems [LeCun et al., 2015], namely Deep Learning. One of the advantages of Deep Learning techniques is that the input data is in its raw form and the representations needed for detection and classification can be found automatically, with this in mind I will now present the goal of this thesis.

1.2 Goal

The goal of this study is to build a system that can learn the patterns and extract information from both previously seen and unseen templates of documents given minimal data. This case is also more practical since the business documents that the companies manage may change over time due to new customers or current ones changing their layouts.

1.2.1 Scope

The data is collected from S&P's databases of insurance documents. The data is stored with its corresponding document and has been manually annotated by the employees over the years. We will focus on insurance policy letters that S&P receives from different senders to study how well the extraction of data works on these types of documents, this same approach can be applied to other form of documents as well, such as invoices, contracts and so on.

In regards to scrambled data, there is a major area of study that we will disregard from this research, namely Optical Character Recognition (OCR), which is the task of translating handwritten or printed text characters to machine-encoded text [Chaudhuri et al., 2017]. In some cases, even well functioning engines will fail to translate the text properly due to the poor quality of images. We are are going to assume that an OCR engine is used to prepare the documents that are to be extracted, well functioning OCR engines that are open-sourced like Google's Tesseract is one example[1].

1.2.2 Contribution

The rule and pattern-based systems that were previously addressed allow users to define any number of fields that they wish to extract from a given template, in our system we assume instead that there are a fixed number of specific fields that should be extracted from any document if they are present. With this approach, the system should be less dependent on specific layouts of documents but will be limited to a specific task.

1.3 Research questions

1. Which deep learning techniques have performed well in the area of document information extraction?
2. How does the quantity of our annotated data affect the performance of the system?
3. How well does our deep learning system generalize to unseen templates?

1.4 Outline

In this report, we will start with presenting the necessary background theory that the reader needs to have a grasp of to follow the methodology of this study. The data collection and choice of models will be addressed in the following chapters. Lastly, the results will be presented and discussed followed by a conclusion where we reflect on what we can take away from this research.

[1]https://tesseract-ocr.github.io/

Chapter 2

Background

2.1 Information extraction

Information Extraction (IE) is the process of extracting structured data from unstructured or semi-structured documents used for communication between humans [Sarawagi, 2008], e.g. transforming a PDF document (figure 1.1) to mapped fields (figure 1.2).

Portable document format

In this section we will give the reader a brief overview of the portable document format and address the parts that are relevant for this thesis. PDFs are based on PostScript page description language [Adobe Systems Inc, 1999]. PDFs often seem to contain structured text and data, this is however not the case. There are no paragraphs, sentences or even words defined, only the characters and their placements. In other words, unlike other document formats a PDF does not contain a stream of text.

A PDF document consists of a collection of objects that together describe the appearance of one or more pages. A PDF file contains the objects making up a PDF document along with associated structural information, all represented as a single self-contained sequence of bytes [van der Knijff, 2009].

Since there are no structures such as sentences or words in PDFs, these need to be constructed before extracting the information. The construction of such structures are made based on heuristics and the positioning of the characters. One approach taken in PDFMiner, a tool for extracting information from PDFs, to construct the text is broken down in three steps: 1) group characters into words and lines 2) group the lines into boxes and 3) group the text boxes in a hierarchical order.

Grouping characters into words and lines

Each character has an x-coordinate and a y-coordinate for its bottom-left corner and upper-right corner that together form a bounding box. These bounding boxes determine the characters that belong together, the characters that are both horizontally and vertically close are grouped onto one line. The distance between how far away characters can be from each other are determined by heuristics e.g the distance between two characters "s" and "t" has to be smaller than a set margin M, see figure 2.1, to be grouped together. Characters that are further away from each other than the distance M but in a proximity smaller than W belong on the same line. There are other heuristics that are considered when calculating what characters belong on the same line as well, such as vertical alignment of the bounding boxes and insertion of spaces between characters since PDFs are not aware of the space character. We will not dive into specifics about the calculations, but the reader is recommended to read the documentation for PDFMiner if it is of interest[1]. The result of this step produces a list of lines, where each line consists a list of characters.

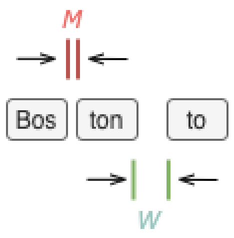


Figure 2.1: Example of grouping characters to words.

Grouping lines into boxes

Each line has a bounding box that is determined by the bounding boxes of the characters that it contains. The bounding boxes are then used to group the lines, similarly to the previous step. If lines are both horizontally overlapping and vertically close they will be grouped together. Lines that are grouped together have a distance, calculated from the tops and bottoms (L_1 and L_2 in figure 2.2), that is smaller than a defined margin. The result of this stage is a list of text boxes, where each box consists of a list of lines.

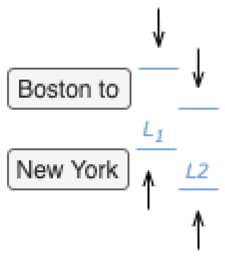


Figure 2.2: Example of grouping lines.

[1]https://github.com/pdfminer/pdfminer.six

Example of grouping text boxes

In this step, two text boxes that are closest to each other are merged continuously. The proximity is calculated as the area that is between the two text boxes, see the gray area in figure 2.3. It is the area of the bounding box that surrounds both lines, minus the area of the bounding boxes of the individual lines.

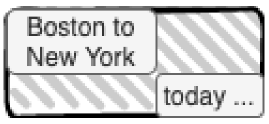


Figure 2.3: Example of grouping text boxes.

E-mail, text messages and more are examples of other forms of communication. We will refer to all such artifacts as unstructured since all of them contain data described in the human language that do not conform with formal data structures that are associated with relational databases.

The motivations for extracting such data into a structured machine-readable format:

1. The communication is often done through artifacts such as documents and emails written in the human language.
2. Managements and storage of such artifacts are maintained with computers.
3. The computers do not understand the communication.
4. Computers

Imagine a scenario where you are exchanging a lot of e-mails with job recruiters and trying to arrange a meeting for an interview. You agree on a date, time, receive a link to the video chat, etc. and so you create an event in your calendar, even though all the information is conveyed in the e-mail the application can not create the event for you automatically without human verification.

Understanding unstructured communication would require strong Artificial Intelligence (AI) that is completely on the same level as human intelligence in all aspects [Yang et al., 2017]. If such a system would exist today there would be no necessity of having IE systems or many other Machine Learning systems for that matter, as we could simply instruct the AI to do any number of tasks such as booking a conference room for a meeting at a given time simply by reading an e-mail conversation between colleagues. There are systems that have been making strides in this direction today but for this thesis we will focus on task-specific systems.

Two examples of broad approaches taken for these systems fall into the categories:

1. Rule- and pattern based systems.
2. Token classification.

2.1.1 Rules and patterns

Rule and pattern-based IE is a well-researched area that is continuously expanding to this day. A survey by [Chiticariu et al., 2013] found that these systems still dominate the commercial world even though it is considered obsolete by the academia. The common theme for this solution is that you look for patterns in the unstructured data and make use of these to extract the data you are interested in. Some examples of patterns in data that we would like to extract may be a company's organization number or the due date in an invoice. If we are assuming that there is a certain standard that an organization number must follow, for instance, $"0123-45678"$ where it has four digits at the beginning followed by a dash sign and 6 digits at the end, then it would be possible to capture this with a regular expression (regex) [Thompson, 1968]. Dates might be more complex since there can be various formats such as "2020-02-23" or "23rd Feb 2020". In this case, we would have to expand a simple pattern like the one above and create more to try to match the various ways of writing dates. The problem with this approach is that regex might match to other fields in the document also, the regex will also match to other strings with the same format like $2546-32-46$ which may be a variant on an insurance number for a given template.

To account for this problem rules are created to decide which string should be chosen, for example, choosing the candidate that has the closest relative position to the keyword Organization number. These rules and patterns will vary between templates and needs to be evaluated per template, in 1.1 specifying that it should be at the top of the page would be an additional rule for choosing the organization number given that template will remain the same [Chiticariu et al., 2013]. This is a simple example of how one would approach extracting information from documents, but it is also one of the main advantages of rule and pattern-based IE systems. They can be simple to understand and modify, the error can be traced back to the rules where the rules can be modified by the users. However, therein lies one of the drawbacks of this method, which is the need for operators to manually create and adjust the rules and patterns [Chiticariu et al., 2013].

2.1.2 Token classification

There is another major approach in Information Extraction, namely token classification. The main idea is to classify a symbol, which can either be on word-level or character level, to a label determining if it should be extracted or not. A labeling scheme can be of such as Beginning-Inside-Out (BIO) [Ramshaw and Marcus, 1995] scheme used in The Air Travel Information Services (ATIS) data set. One of the advantages of such a scheme is that it enables the extraction of fields that stretch over multiple words, meaning fields can contain delimiter characters. See table 2.1 for an example of the ATIS data set, in the table we see that labels most of the time are split into two parts, it contains a prefix label followed by the label for a named entity e.g city. The prefix label "**B-**" indicates the beginning of a field and subsequently, the "**I**" label indicates the continuation of that label. The last label in the data set "**O**" indicates that the word is not a key token to be extracted, hence not a part of the labels.

Another variation of BIO labeling scheme is Beginning-Inside-Out-End-Single (BIOES). The

Word	Label
Show	O
flights	O
from	O
Boston	B-city
to	O
New	B-city
York	I-city
today	O
.	O

Table 2.1: ATIS data set example.

three first labels in BIOES are the same as the former, the extended labels "E" and "S" refer to the end of a field and a single-word entity respectively [Liu et al., 2018]. One of the primary advantages of token classification addresses the problem we mentioned with the rules and pattern-based approach, supervised learning enables the discovery of complex patterns in the data that the human operator might not be able to find which may prove to be optimal. Since these patterns are learned from the labeled training data, they have shown to be more resistant to common noise in the training data such as misspellings, formatting differences like in the case with dates and so on. This move of giving the learning algorithm the responsibility of creating the complex patterns does not come without some costs, which are the drawbacks of this method. The labeling of the training data is usually done by humans [Chollet et al., 2018] and requires quite extensive work. The system is also often more difficult to understand compared to a rule and pattern-based approach which makes it more difficult to improve [Chiticariu et al., 2013]. These are the reasons why rule and pattern-based approaches are still dominating the industry today, the effort required to label the training data makes this approach not as feasible for many practical IE tasks.

2.2 Machine Learning

This section addresses the theory and methods applied in this thesis, everything falling under the Machine learning paradigm. Some brief knowledge about simpler machine learning methods will first be addressed and then proceed to dive deeper into deep learning. Mostly focusing on Artificial Neural Networks (ANN's) that are relevant for this thesis, related work is also presented for readers that need to freshen up their memory or are new to the area.

2.2.1 General concepts

The common approach to work with a data set with the given structure described in section 2.1.2 is to use supervised learning, which maps input data to the annotations. There are many supervised classifiers to choose from such as logistic regression, neural networks, decision trees and so on. Let us discuss some general ML concepts before we dive deeper into different kinds of classifiers and our choice of models in this thesis.

Training, validation and test

An important concept we would like to address is the importance of training-, validation- and test set. The training set contains a sample of data that we use to build the model along with its parameters. The trained model is then challenged with another sample of data, the validation set, this set is used to assess the model's accuracy. Based on the errors from the validation set, the optimal parameters for the model are selected by using the lowest validation error. This procedure is called model selection [Hastie and Tibshirani, 2009]. The test set is a sample of data that is not used during the model selection process, it acts as a blind test in the end to observe how well the model performs on data it has not seen before.

Overfitting

A problem that occurs in every machine learning problem during training is overfitting [Chollet et al., 2018], this is related to the tension of optimization and generalization. Optimization is the process of adjustments made during training to achieve the best results possible on the training data, generalization is how well the model performs on a sample of data that it has never seen before. These two are very much correlated at the early stages of training, as the model performs better on the training data the performance on the test data will improve as well. The generalization will stop to improve after a certain amount of steps however, validation metrics will stagnate and degrade. The model starts to learn patterns that are specific for the training data, this leads to worse performance on the test data. The best way of preventing this, which is called regularization, is to get more training data when possible. There is another effective method of achieving regularization but more on that in the section 2.2.2 where we address Artificial Neural Networks.

Logistic regression

In this thesis, we are going to use a variant of the logistic regression classifier as our baseline model. Logistic regression can be defined with the following equation:

$$P(Y=1|X) = \frac{1}{1+exp(\omega_0 + \sum_{i=1}^{n} \omega_i X_i)},$$

where Y is a discrete value and $X = x_1, ..., x_n$ is a vector containing discrete or continuous values, ω is chosen by maximizing the conditional data likelihood, which is the probability of the observed Y values in the training data [Arabnia and Tran, 2016]. Advantages of using models like Logistic Regression are that the computations can be quickly calculated by computers and the results are easy to interpret, this model in particular only works on binary data. An extension of this simple model, Multinomial Logistic Regression (MLM), can classify data into several categories that have no natural ordering. It is a maximum likelihood estimator that has shown great results when solving classification problems with numerous categorical values [Chanyeong and Alan, 2002]. For more complex models we will now start diving into deep learning.

2.2.2 Deep Learning

Deep learning is defined by oxford languages as: "a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data". The concept of deep learning was already well understood back in the 1990s and some areas have not changed much since then, the reason for deep learning making strides in recent years lies in three factors. 1) Hardware improvements 2) Datasets and benchmarks created and 3) algorithmic advances since then. Deep Learning is part of the machine learning family and takes on a new way of learning representations of data by using successive layers. In other words, deep learning is not meant to describe any deeper understanding of the data but how many levels of layers contribute to the model. This is in contrast to shallow machine learning methods, which only use one or two layers for the representation and rely on feature engineering of the data from humans to be effective [Goodfellow et al., 2016]. These layers in Deep Learning are learned through artificial neural networks, which have the structure of layers stacked on top of each other [Chollet et al., 2018].

Artificial Neural Networks

The term *neural networks* is a reference to the study of the nervous system - Neurobiology. Central concepts of deep learning were inspired from our understanding of the human brain but it should not be interpreted as ANN's are models of the brain. However, similarly to the human brain ANN's makes decisions based on collective input from all of its' neurons. The basic structure of ANN's is built of three types of layers: input, hidden and output layers, see figure 2.4 for a graphical representation. The layers are responsible for the data transformations of the input which are determined by its' weight parameters. Fine-tuning the weight parameters is essentially the iterative process of *learning*, to find the correct set of values for the weights of each layer such that the inputs transform to their corresponding output during training.

Previously when we addressed overfitting we mentioned that there is another effective way to achieve regularization, this is by incorporating a method called dropout. Dropout is a reference to dropping out both hidden and visible neurons along with their paths in the network temporarily [Srivastava et al., 2014], see figure 2.4. These paths get randomly disabled by being multiplied with 0 with a certain probability. When the sets of paths and neurons are disabled, the process now becomes equivalent to training different neural networks. The different networks will overfit in various ways and so averaging the effects of the different networks will lead to reduced overfitting. Dropout helps to deal with issues such as collecting

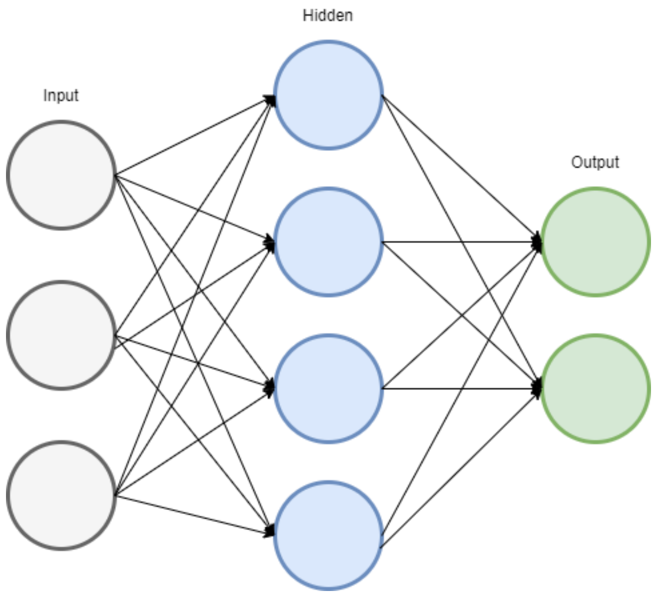


Figure 2.4: Basic architecture of an ANN.

more training data and finding optimal hyper-parameters for large complex networks which can be a daunting task.

Embeddings

At this point, we need to address how words and characters are represented mathematically. The most usual approach of doing it is through dense high-dimensional vectors called embeddings. These vectors can either be trained manually or one can make use of ones that have been pre-trained on large unannotated corpora. Similar words are made to have similar values in these embedding spaces, one of the most popular approaches for this is called Skip-gram methods [Mikolov et al., 2013].

The challenge with embeddings is that there is no uniform way that words relate to each other, a lot of words have different meanings depending on the language and culture. This entails that the choice of embedding space depends on what kind of task one wants to solve. In this thesis, we are only going to focus on character embeddings, which are trained at the same time as our classification task.

Character embeddings have shown to be one of the fundamental inputs for neural networks when researching NLP tasks [Chen et al., 2015]. The characters are usually extracted from the training set unless otherwise specified and composed to a character dictionary C with the size $|C|$. Unknown characters are mapped to a special symbol that is not used elsewhere. Each of the characters are represented as a real-valued vector (character embedding) $\mathbf{v}_c \in \mathbb{R}^d$ where d is the dimension of the vector space. The character embeddings are at last stacked into an embedding matrix $\mathbf{M} \in \mathbb{R}^{d \times |C|}$.

This is done by incorporating an embedding layer [Chollet et al., 2018] into our model, the weights of this layer are initially random and adjusted during training with backpropagation as with other layers.

Backpropagation

The central algorithm in deep learning that enables optimization is called backpropagation. Backpropagation is a highly computational operation that is more often than not improved by components called *optimizers*, common ones are for instance Adam and Stochastic gradient descent (SGD) [Kingma and Ba, 2014]. We will refrain from explaining the theory [Hecht-Nielsen, 1992] behind the algorithm in depth since it is quite complex and implementation is not required for working with deep learning tasks. The *optimizer* determines how the partial results from backpropagation should be applied, the goal is to produce better models with either fewer steps or in general. In ANN's the training is achieved through a feedback loop, see figure 2.5.

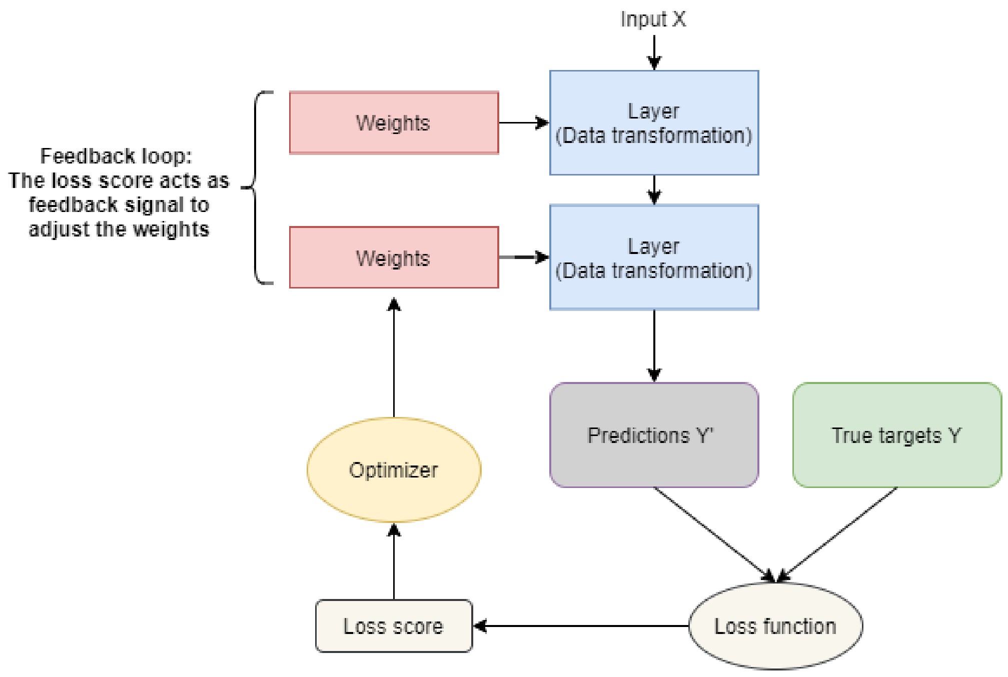

Input X
Weights
Layer (Data transformation)
Feedback loop: The loss score acts as feedback signal to adjust the weights
Weights
Layer (Data transformation)
Predictions Y'
True targets Y
Optimizer
Loss score
Loss function

Figure 2.5: Feedback loop in Recurrent neural networks.

During training, each time the data is processed in the network the error is calculated by comparing the predicted output to the expected output. The error found in the comparison, the loss score, is calculated in the loss function which is then used by the backpropagation algorithm to calculate how each layer from top to bottom contributed to the loss score, thus calculating the weight modification needed in each layer. Once the calculation is done it recursively modifies the weights of each layer until it reaches the input layer.

Some other concepts that we would like to cover in this thesis to give the reader deeper insight to understand the methodology are feedback loops implemented in Recurrent Neural Networks (RNN's) to achieve optimization, as well as embeddings- and CRF layers.

Reccurent Neural Networks

RNN's are part of the neural network family that works with sequential data, the input is a sequence of vectors $(x_1, x_2, ..., x_n)$ and returns another sequence $(h_1, h_2, ..., h_n)$ that in a way represents some information about the sequence at every step in the input. What makes sequential data distinguishable from single data points is that the order of the data points matters. These networks have shown improvements in the state-of-the-art of many NLP areas, for instance in speech recognition [Graves and Schmidhuber, 2005].

Recurrent neural networks do in theory have the capability to learn and remember long dependencies but it is shown that they more often than not tend to be biased towards their most recent inputs [Bengio et al., 1994]. In the approach of alleviating this problem, a new form of RNN's was developed back in 1997, called LSTMs, that would be able to capture long term dependencies with the help of an integrated memory-cell [Hochreiter and Schmidhuber, 1997].

Long short-term memory

As previously mentioned LSTMs are a type of RNN, thus it also works with sequential data and is also able to remember long-term dependencies. There are few variations of LSTMs and few iterations on the LSTM neural network will result in what we today know as bidirectional LSTM.

A forget gate was proposed in 1999 and added to the network which gave it the ability to forget earlier activation at certain steps and reset the hidden state of the node [Gers et al., 2000], this was important since the networks could still not accurately learn from certain very long or continual time series that did not have an explicitly defined start and end. The issue lied in the internal values of the cells growing without bound because of the continual output stream, even though it should have been reset at times.

The steps which allow the model to remember the long-term dependencies whilst also enable short-term predictions can be represented mathematically as:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{ht-1} + b_{hi}) \tag{2.1}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.2}$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{2.3}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.4}$$

$$c_t = f_t \times c_{t-1} + i_t \times g_t \tag{2.5}$$

$$h_t = o_t \times \tanh c_t \tag{2.6}$$

where i_t, f_t, t_t, o_t are the input, forget, memory cell, and output gate, h_t is the hidden state and the output in each step, and c_t is the cell state. σ and $tanh$ are the sigmoid function and hyperbolic tangent functions respectively.

The things we have mentioned about LSTMs so far are mostly about their ability to remember both short and long-term dependencies, it is also assumed that the relation of the sequential data is only in the forward direction. Graves and Schmidhuber (2005) proved however that the same procedure can be applied with the sequence fed in the backward direction also, resulting in the Bidirectional LSTM (Bi-LSTM) network.

In this thesis, we are going to use a BiLSTM-CRF architecture which is a BiLSTM model stacked together with a Conditional Random Fields layer (CRF) [Lafferty et al., 2001]. CRFs assign a well-defined probability distribution over possible labels, trained by maximum likelihood. The loss function in CRFs guarantees convergence to the global optimum. Furthermore, the reason for choosing to experiment with this additional layer in this thesis is because they have been shown to produce higher performance when combined with LSTM networks. For a graphical representation of the BiLSTM-CRF model see figure 2.6.

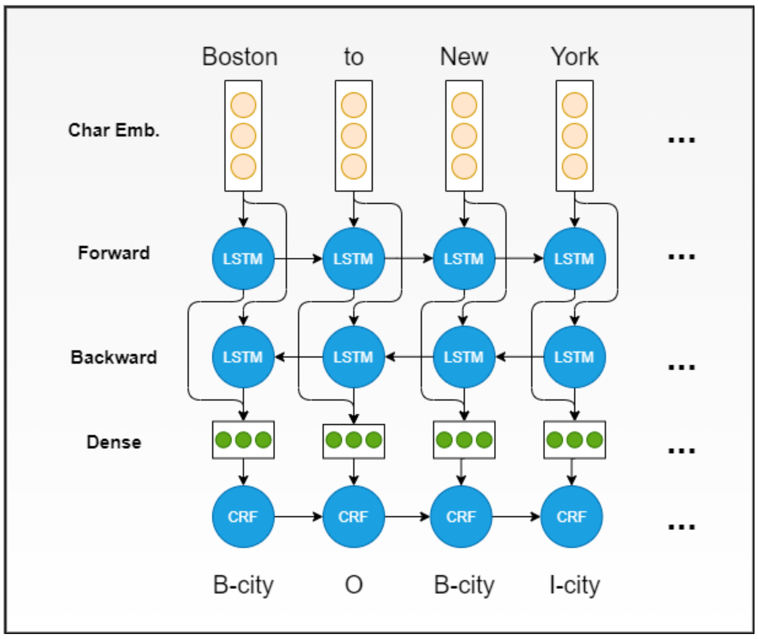


Figure 2.6: BiLSTM-CRF

2.3 Evaluation metrics

One of the most popular metrics with researchers today is Accuracy. The definition of Accuracy is the ratio between the number of correctly classified samples and the overall number of samples [Chicco and Jurman, 2020]. This metric works well with multi-class cases such as this problem in the study but it is however important to deal with imbalanced data sets when using this metric to evaluate the performance. This is because it will produce an overoptimistic estimation of the majority class when classifying, which is the "O" label described in section 2.1.2.

There are other methods that deal with overcoming imbalanced data like Matthews correlation coefficient (MCC) but has not gained popularity within classification problems yet. It may be because there are certain situations where MCC cannot be defined or it displays large fluctuations, there are although workarounds that are available for those cases. The F_1-measure is going to be one the evaluation metrics used to evaluate the token classification in this study. It consist of the metrics *precision* and *recall* that are combined into one score. F1 is one of the most popular evaluation metrics in machine learning research, not only in binary classification scenarios but also in multi-class cases such as this one. To read more about these metrics, the reader is highly recommended to read the report written by [Chicco and Jurman, 2020]. The results of the extracted information from the documents can be divided into four classes. For a visual presentation of the classes, see Figure 2.7

True positive (tp) Actual positives that are correctly classified by the system.

False positive (fp) Actual negatives that are incorrectly classified by the system.

True negative (tn) Actual negatives that are that are correctly classified by the system.

False negative (fn) Actual negatives that are incorrectly classified by the system.

Recall is the fraction of correctly extracted samples that are relevant:

$$R = \frac{tp}{tp + fn} \tag{2.7}$$

Precision is the fraction of extracted samples that are relevant:

$$R = \frac{tp}{tp + fp} \tag{2.8}$$

F_1**-measure** is the combination of R and P using the harmonic mean:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{2.9}$$

Optimizing the performance of a system on *Precision* and *Recall* tends to be a difficult task. This is because they tend to work against each other. For instance, let us imagine that there is a threshold for determining if a document is correctly classified. If the system is to be optimized for recall the threshold for the classification could be decreased, the consequence would be that the number of false positives would increase leading to worse performance on *precision*. Vice versa, if the system is to be optimized for precision, we would reduce the number of false positives by increasing the threshold for classification, the consequence of this is the increased number of false negatives resulting in worse performance on *recall*. The harmonic mean in the F_1-measure is used to reduce the score when recall and precision are not in balance.

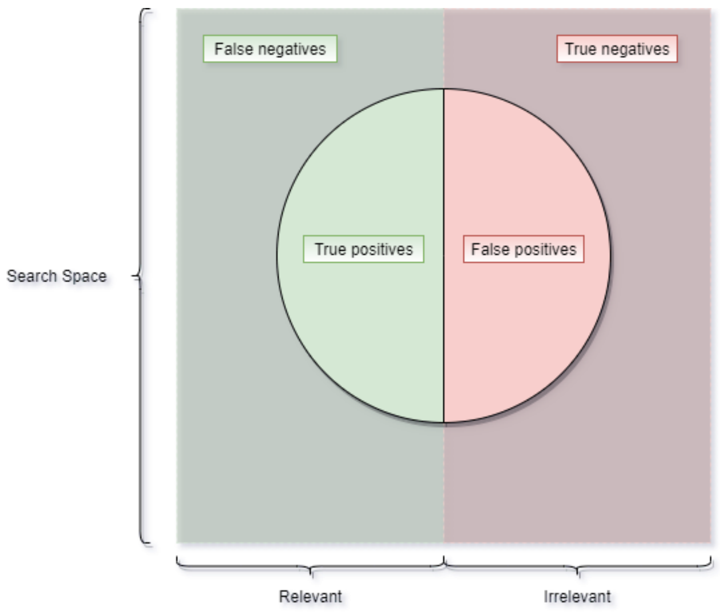


Figure 2.7: The search space for documents

2.4 Related Work

A fundamental aspect that is needed for our system is annotated data for the set of specific fields we are going to extract from the insurance policy letters. Some systems require such preparation made by ITESOFT and DocuWare, databases of templates are created beforehand where fields, keywords and positions for each field are stored [Schuster et al., 2013, Rusinol et al., 2013]. ABBY FlexiCapture works in a similar fashion, when the template has been classified the keywords and positions for each field are considered when suggesting field candidates. Each candidate has a score using heuristics such as proximity and uniqueness of the keywords, so the candidates with the best scores are chosen. Alternative ways of approaching this problem can be found in a study made by [Medvet et al., 2011], where they use a probabilistic model for finding the most similar pattern in a template and extract the fields with an associated parser. Unfortunately, the aforementioned studies and systems cannot be directly compared with our approach since the data sets and evaluation process are not publicly available. Although what we do have in common is that annotated data from each template is required to accurately extract the information from the documents correctly.

The task we are trying to solve in this thesis is closely related to Named Entity Recognition (NER), thus it is relevant to address this area as well. NER plays a key role in Natural Language Processing tasks. It is a form of information extraction, NER is mainly used to locate the named entities in unstructured text such as name, date, location, organization, etc. [Fu et al., 2021]. For years, ML problems were constrained to solving problems such as classification and pattern recognition with hand-crafted feature engineering, it was not possible to learn from the data in its raw form. This limitation often required attentive engineering and domain expertise to transform the raw data to a suitable format such as feature vectors from which the learning system could recognize patterns and output some form of classification. This was the case for traditional named entity recognition methods, for example the study [Yang et al., 2017] where they presented a few shallow machine learning studies that produced F1 scores at 70% for biomedical named entity recognition.

Neural architectures for NER were experimented with here the approach was to only use supervised training data and unlabeled corpora by [Lample et al., 2016]. With this approach, they previously obtained state-of-the-art performances in 3 out of 4 languages and were very near with the fourth one. This study, where word and character level embeddings RNN's with CRFs was used, inspired the neural architectures we are going use due to their simpler nature. Similar studies like ours, where LSTM models have been used to achieve information extraction [Palm et al., 2017] have been researched. The research made by (Palm et al. 2017) experiments with LSTM models on both previously seen and unseen templates of invoices, the models performed well on both types and showed signs of generalization.

There are also other neural architectures that have achieved impressive results in recent years. This is an article about a great library for NLP, FLAIR, which has been used to achieve state-of-the-art performance today [Akbik et al., 2019]. BERT is a classifier that was introduced in 2018 [Devlin et al., 2018], the goal is to create a general model that can be fine-tuned for many NLP tasks. The results showed that rich unsupervised pre-training is an essential step in many NLP systems.

Chapter 3

Method

The first three sections in this chapter are going to be focused around the data set, the challenge of collecting a sufficient amount of data for the study. The following sections will address the pipeline and models that we have considered during the thesis. Some technical information about the models will be presented to give the reader an overview of how they work.

3.1 Data collection

Initially, our plan was to collect the insurance documents and corresponding fields of interest from the database. The idea was to gather lots of data that has been stored in the database over the years which would result in at least some 10.000 documents, the documents have also been annotated and maintained by the employees which would hopefully mean that there were relatively small inconsistencies with the data. Unfortunately, some obstacles were met during this approach that required some big adjustments, customers updated their insurance policies and newer versions of the documents were revised regularly. The consequences of this meant that sometimes the newer documents no longer matched with the older annotated data. This certainly was the case for older insurances dating back 8-10 years which made them not useful at all, since it would not be worthwhile to check if all fields were correct in respective documents. A large number of documents were also stored as images with different scan quality that created garbled output from the OCR-engine used at S&P, meaning that a portion of documents that may have had correct annotated fields will still need to be excluded from the data set due to poor text quality. Since OCR-related problems are a whole research area on their own this made us more discouraged to further pursue this way of collecting data.

One of the ways to work around this problem is to only use digital documents that were sent from specific insurance companies, the focus would now shift to manually annotating the documents. This would obviously mean that the data set will be smaller than what we had hoped and countless hours of the research going to collecting the data set. The strategy is to produce a data set consisting of 2100 insurance policy documents from different insurance companies resulting in 8 different templates. The models will be trained on 5 distinct templates and be evaluated on how well they perform on these templates initially and then evaluated on how well they perform on 200 documents consisting of 3 other templates they have never seen before.

Due to confidentiality the different senders of the templates cannot be disclosed, we will give each template a unique id for reference in table 3.1 instead. Table 3.1 displays the distribution of the seen templates we train our model on, the amount of each template in each set. The data set was split in a 60/20/20% ratio for training, validation and test respectively for the previously seen templates. Table 3.2 displays the distribution of documents from the unseen templates we have in our data set.

Template id	Train	Validation	Test	Total amount
1	324	108	108	540
2	306	102	102	510
3	252	84v	84	420
4	138	46	46	230
5	120	40	40	200

Table 3.1: Distribution of previously seen templates in our data set.

Template id	Total amount
6	84
7	62
8	54

Table 3.2: Distribution of unseen templates in our data set.

Since manually annotating the documents was a mundane and labour-intensive task, the focus lied on annotating 7 fields per document that were considered to be the most important judging by what was stored in the databases. The fields are the ones that were referenced early in the study, see image 1.2. The naming convention of these fields is in CamelCase, the same format S&P uses for fields in their databases.

InsuranceNumber is the unique id for a client's insurance case, the same client can have several insurances within the same company and thus the id will be different depending on the insurance.

BusinessId is in reference to the client's organisation number, some clients may not have an insurance via their company and this field may not be present in all documents.

BusinessName is the client's company name.

InsurancePeriodStart and **InsurancePeriodEnd** are the dates for which the insurance is active under.

YearlyPremium and **YearlyComission** are premiums the client is paying for the service.

Our hypothesis about these specific fields is that there will be a lot of uniqueness i.e. information entropy among each field on a word level. Since insurance numbers are supposed to be unique, there will not be many fields that are identical except with documents that are about the same insurance. This is more or less the case for the rest of the fields also. In the next section, we will address our assumptions and motivate our choices for the embeddings.

3.2 Data Analysis

For our neural models, we are going to focus on character embeddings due to this task being very much a pattern matching problem. The fields we want to extract do have a lot of disparity among each class, the fields do not have a lot of similarities except for their pattern. For instance, an insurance number in figure 1.1 has the format $xx - xxxx - xx$ where x is a digit and does not necessarily have an explicit list of values it can contain since the number of combination is high.

The character vocabulary of the data set will be around 100 tokens in total when using single character embeddings. We would also like to experiment with bi-character embeddings to create a larger embeddings space to see if the models can make use of these to learn and find more optimal patterns for classification. Bi-character embeddings are essentially produced in the same manner we have described character embeddings back in section 2.2.2. The difference between single character embeddings (Char) and bi-character embeddings (Bi-char) is that the latter has an embedding space composed of combination of characters i.e. tuples of characters $\{(a, b), (b, a)\}$. The embedding space with bi-characters would be a bit large if we did not filter out combination of characters that are not prevalent as much in the data set, so we filter out combinations that occur less than three times in total in the training set which result in an embedding space of 998 tokens.

The large embedding space is one of the advantages of using large pre-trained word embeddings in NLP tasks, but the problem is that those instances they're trained on won't be applicable to this specific task. Not to mention that there are not a lot of pre-trained embeddings to choose from in the Swedish language.

This is not to say to word embeddings have no value in this context, more about this will be reflected upon in chapter 5 section.

3.3 Annotation Guidelines

The annotation starts with extracting the fields manually from the documents into a structured machine-readable format, in our case the JSON format was chosen [JSO,]. Each document that was annotated was assigned a unique filename in the JSON file, this was important when working with this scale since filenames could easily overlap and we needed a way of distinguishing them for evaluation. The filename would then be set as a key for each corresponding JSON object.

To annotate the documents we opted for an automated solution, the algorithm for the automated solution is shown in the figure below. The content of the PDF files would be extracted and segmented into bounding boxes, *GetTextSegments* in the figure, with PDFMiner. Each segmentation in the PDF will be considered and checked if it matches an annotated value in the JSON object counterpart, if it does then that segment will be annotated with the field. In such cases where a segment contains multiple fields, the corresponding segment will be labeled with all matching fields. In the case where the segmentation did contain any of the fields, which was most of the time, it was labeled as "None".

Algorithm 1: Automatic training data extraction

```
$result \leftarrow \{\}$
foreach $document \in documents$ do
    $textSegments \leftarrow GetTextSegments(document)$
    foreach $textSegment \in textSegments$ do
        foreach $field \in JSON(document, field)$ do
            if $textSegment = field$ then
                $labels \leftarrow GetLabels(textSegment)$
            Add($result, textSegments, labels$)
```

This would produce a little bit of noisy data since segments could be mislabeled at times, depending on text quality and other factors, which would require manual correction but it would allow for easily expanding the available data. As long as there are signs of the models learning and performing well overtime on the data sets it is a reasonable trade-off in our opinion.

3.4 Pipeline and overview

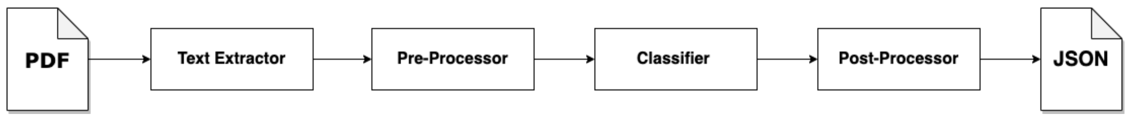


Figure 3.1: Pipeline

A general pipeline for our neural models can be seen in Figure 3.1.

Text extractor

We start with extracting the text from the PDFs according to their bounding boxes in hierarchical ordering. The text is in other words extracted and segmented according to their enclosed coordinates in the document. The argument for doing so is because the PDFs have semantic separations of the data fields we are interested in, like we discussed back in section 2. Usually, text enclosed within the same bounding box in unstructured documents such as insurance policy letters, invoices, etc. provides good information for the models to find patterns in. For example, the keyword "Date" may be in the same bounding box as the fields *InsurancePeriodStart* and *InsurancePeriodEnd*. Hence, the models will learn that the text following respective keyword may be one of the candidates for the corresponding field.

Pre-Processor

In our baseline model we experimented with pre-processing steps such like stemming the text and removing stop words. Stemming is a normalization method, reducing words to their word stem, and has shown to improve performance in information extraction of documents [Korenius et al., 2004]. The stop words are removed since it is most commonly used words in human language and do not add additional information of text. We also hypothesized that creating more features by replacing the numbered fields like the dates and insurance number with tokens would improve the performance with the model. However, this did not seem to improve the performance by much and in some cases, the results were less impressive so we decided to discard the last step. Before feeding the input to the classifier a CountVectorizer from Keras was applied to the text segments to transform each token to a corresponding index.

There were no particular pre-processing steps taken in our neural models, the data was transformed using the embeddings and fed into the classifier afterwards.

Classifier

We chose to work with the Multinomial Logistic Regression (MLM) we addressed back in chapter 2 for our baseline model. Three advantages of using this model is that it is widely available, the computations can be quickly calculated by computers even with a vast amount of categories, and the results are simple to interpret. This makes it desirable choice for a baseline model [Chanyeong and Alan, 2002].

We chose to work on four variations on LSTM models for our neural networks, we will dive deeper into the architecture of the neural models in the following sections.

Post-Procesing

Once the classification was finished in our neural models, the characters got assigned with BIO-label which had the highest probability. Before chunking, the characters and their corresponding labels were post-processed. Characters with labels that started with the prefix "I" but were preceded with a character that had an "O" label was changed to the prefix "B". A character that was labeled with the prefix "I" but followed with a character that had the label "O" was changed to the prefix "E". This step was crucial to improving the performance of the system. See figure 3.3 for an example of a classified date field in our neural models.

Character	Label
2	B-InsurancePeriodStart
0	I-InsurancePeriodStart
1	I-InsurancePeriodStart
9	I-InsurancePeriodStart
	I-InsurancePeriodStart
1	I-InsurancePeriodStart
1	I-InsurancePeriodStart
	I-InsurancePeriodStart
0	I-InsurancePeriodStart
1	E-InsurancePeriodStart

Table 3.3: BIOES-labeling scheme example on a field

The last step in the post-processing was normalizing the classified fields to a preferred format. The fields were lastly written in to a JSON object before the evaluation.

3.5 LSTM models

LSTM's are a form of Recurrent Neural Network, we addressed the theory of LSTM's back in section 2.2.2. RNN's have the ability to model the entire insurance document in a principled manner and our hypothesis is that this attribute will yield good performance in our task. The argument for experimenting with neural networks is also that it does not require feature engineering for the models to be effective. The architecture of the neural models and the hyper-parameters were fixed to all four variations of our LSTM models. These hyper-parameters were chosen based on the overall performance of the models during testing. We experimented with random intializers for the embedding-layer, however, this produced varying results so we opted for using a fixed initializer moving forward. Different fractions in the drop-out layers where also considered, between the span of 20% and 50% in both layers. See figure 3.2 for graphical representation of our neural architecture for the LSTM models.

The first layer in the model is an embedding layer, initialized with Glorot uniform. The reason for using Glorot uniform as our initialization method is due to better performance in our experiment and in a study where they compared several activation functions against each other [Pedamonti, 2018]. The following layer is a drop-out layer with the drop-out rate of 30 % [Gal and Ghahramani, 2015]. The drop-out layers and their parameters were added to the model to avoid overfitting.The following layers are the LSTM layers, the forward and backward layer that we discussed back in section 2.2.2. The last layer in the architecture is one last drop-out layer with a drop-out rate of 20 % and the dense layer. The dense layer is replaced with the CRF layer on the BiLSTM-CRF models, see Figure 3.2 for an overview of the model architecture.

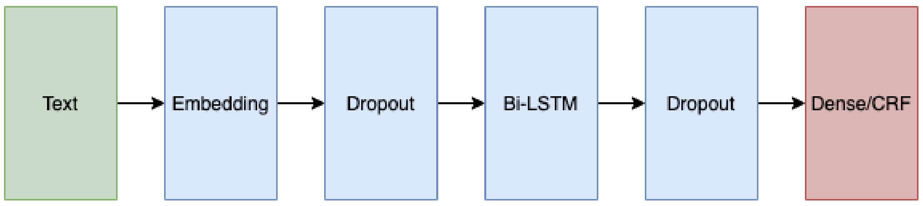


Figure 3.2: The LSTM model

Each model was trained and retrained and evaluated 10 times to get an overall view of how well it performs on this task. We chose to take this approach since the models' parameters are random at the start, in other words, the models have different initial conditions. These initial conditions may lead to better performance in some models than others.

3.6 Evaluation

The evaluation of the models is measured on three performance metrics:

1. The overall accuracy of JSON objects.
2. The accuracy of the extracted fields per JSON object.
3. Their F1 scores on the seven fields.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write, it is also easy for machines to parse and generate. The fields in our JSON objects is referenced earlier in the thesis, see figure 1.2 for a visual representation.

The overall accuracy of JSON objects
The purpose of this evaluation is to measure the total accuracy of all extracted files. A file is classified as correct if all fields in the file are correct.

The accuracy of the extracted fields per JSON object
This evaluation focuses on each field per document. We use this to measure the performance of extraction on each individual field per document.

Their F1 scores on the seven fields
We addressed this metric back in section 2.3, we use it to evaluate how well the models classify each field in the whole data set.

Chapter 4

Results

In this chapter, we will present the results of our research. The first section presents the overview of the results, the model which had the overall best result for the experiments is briefly addressed. Tables relating to the neural models' training and improvement with access to more data are shown. The following sections will dive deeper into each experiment and present each score for respective models.

4.1 Overview

The neural model that had the best performance in our experiment was the BiLSTM-CRF model with single character embeddings. To see how well the accuracy of the model is on extracting correctly per JSON object based on the amount of training data see figure 4.1. Each step is incremental of 10% of the total training data available. We can see that the accuracy plateaus moderately at 80%, it does however increase slightly afterward which is an indication that more training data still would increase the accuracy.

To ensure that overfitting is not occurring we plotted the training losses and validation losses in 4.2 where the orange and blue curves represent the training loss and validation losses respectively. This plot represents the training for the BiLSTM-CRF model and as we can observe the curves are not showing signs of diverging, all models had a similar plot.

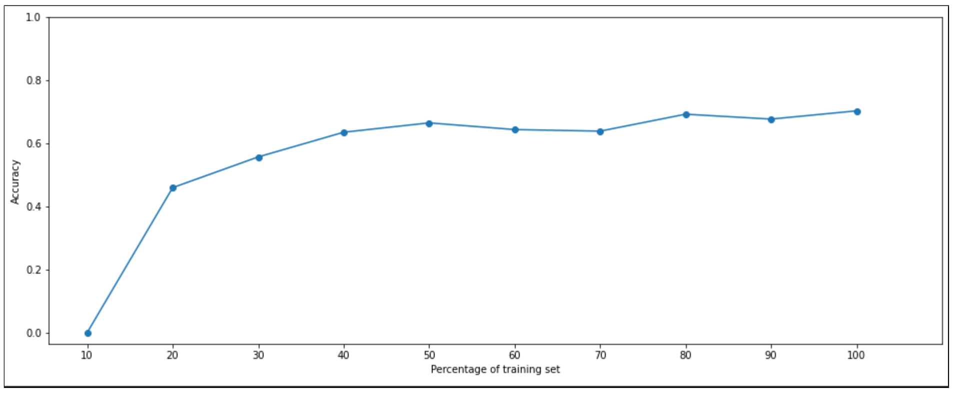


Figure 4.1: Increasing accuracy of BiLSTM-CRF with more training data.

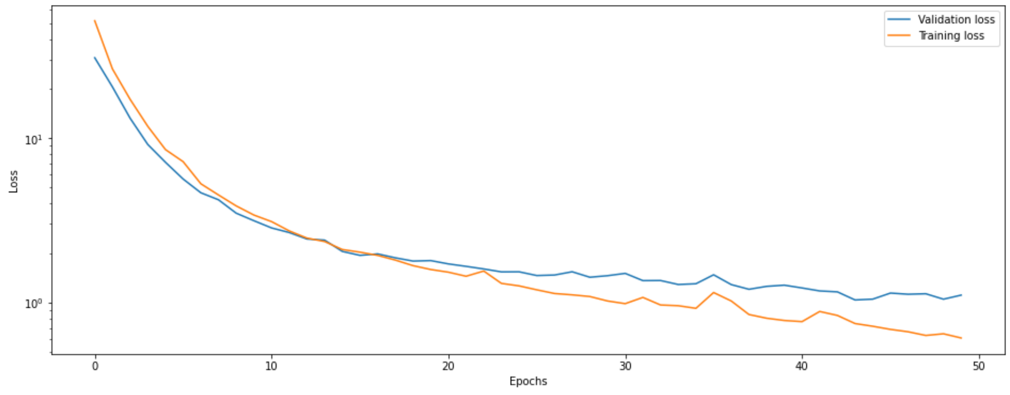


Figure 4.2: Plot displaying training loss and validation loss of our models.

4.2 Seen Templates

The first three tables presented in this section refer to the results achieved on the previously seen templates. The best score for each field is written in bold. The model which has the highest overall performance will be the one we choose to test on the unseen templates. Each model has been trained and tested 10 times so that we could get an estimate on how well this kind of model would perform in practice, creating a confidence interval at 95%. In table 4.3 the F1 scores achieved for the fields are presented.

Accuracy of the JSON objects

In Table 4.1 we see that the BiLSTM-CRF model with the character embeddings performed slightly better than the model without the CRF layer, it also a less variant in its accuracy. This is because this evaluation is the strictest in our approach, the whole file is regarded as wrong if one field is not correctly predicted. We will dive deeper into the performance on the specific fields now.

Model	Accuracy
Baseline (MLM)	0.285
BiLSTM (Char)	0.716 ± 0.011
BiLSTM (Bi-char)	0.517 ± 0.015
BiLSTM-CRF (Char)	**0.728 ± 0.007**
BiLSTM-CRF (Bi-char)	0.548 ± 0.002

Table 4.1: Accuracy of the JSON objects.

Accuracy of the field per JSON object

In Table 4.2 the accuracy of each field per file is presented. One field that was expected to have a worse performance is *YearlyComission*. It performs well on this data set at first glance, however, since it is the least frequent field in the data set the results may be misleading. If the field is not present in the file then that field should be left empty per our implementation. It is also possible that the model may still have found some patterns in this field as we can see that it does with *YearlyPremium*.

Field	Baseline (MLM)	Bi-LSTM (Char)	BiLSTM (Bi-char)	BiLSTM-CRF (Char)	BiLSTM-CRF (Bi-char)
InsuranceNumber	0.724	0.983 ± 0.005	0.985 ± 0.011	**0.993 ± 0.002**	0.987 ± 0.001
BusinessId	0.413	0.979 ± 0.006	0.974 ± 0.003	**0.989 ± 0.009**	0.968 ± 0.001
BusinessName	0.809	**0.880 ± 0.013**	0.692 ± 0.015	0.873 ± 0.004	0.696 ± 0.001
InsurancePeriodStart	0.786	0.936 ± 0.011	0.931 ± 0.005	**0.937 ± 0.004**	0.926 ± 0.002
InsurancePeriodEnd	0.775	0.921 ± 0.008	0.984 ± 0.001	**0.985 ± 0.001**	0.983 ± 0.001
YearlyPremium	0.933	0.908 ± 0.010	0.897 ± 0.008	**0.916 ± 0.007**	0.911 ± 0.002
YearlyComission	0.902	**0.977 ± 0.004**	0.915 ± 0.007	0.911 ± 0.008	0.932 ± 0.001

Table 4.2: Accuracy of the fields per JSON object.

F_1-measure

The baseline model with minimal feature engineering gives us an estimate of what is possible to achieve with simpler models, the model does perform well on some fields but not as much with others. We see that *BusinessName* is the hardest field to classify in all cases, which may not be so surprising due to its high variability. It does have a higher score on the LSTM models, indicating that it learned to find patterns on the field better. Company names in Sweden do usually contain the prefix or suffix 'AB' which may have helped the models with the predictions. Overall, it seems that the Bi-LSTM models trained on single character embeddings performed better, the additional CRF layer seems to have less variation.

In this task, we see that character embeddings worked better than bi-character embeddings. The additional CRF layer to the models improved consistency which led to fewer variations in the results. Because of this, we chose to test the BiLSTM-CRF model with single character

Field	Baseline (MLM)	Bi-LSTM (Char)	BiLSTM (Bi-char)	BiLSTM-CRF (Char)	BiLSTM-CRF (Bi-char)
InsuranceNumber	0.732	0.988 ± 0.002	0.987 ± 0.003	**0.989 ± 0.003**	0.988 ± 0.002
BusinessId	0.458	0.982 ± 0.007	0.989 ± 0.009	**0.992 ± 0.002**	0.983 ± 0.004
BusinessName	0.779	**0.934 ± 0.002**	0.809 ± 0.011	0.876 ± 0.003	0.796 ± 0.017
InsurancePeriodStart	0.864	0.914 ± 0.010	0.915 ± 0.007	**0.917 ± 0.005**	0.913 ± 0.010
InsurancePeriodEnd	0.887	**0.989 ± 0.008**	0.978 ± 0.002	0.978 ± 0.002	0.977 ± 0.003
YearlyPremium	0.948	**0.948 ± 0.005**	0.936 ± 0.006	0.938 ± 0.004	0.937 ± 0.003
YearlyComission	0.794	0.911 ± 0.005	0.913 ± 0.566	**0.914 ± 0.004**	0.908 ± 0.010

Table 4.3: F_1 on the fields from previously seen templates

embeddings on the previously unseen templates that the model has not been trained on.

4.3 Unseen templates

The following tables show the performance of the baseline and BiLSTM-CRF (char) models, the rest of the models were not evaluated on this set due to time constraints. We chose to work with the BiLSTM-CRF model with single character embeddings since it was the highest performing model in the previous experiment. The F_1-score score on the fields on these documents are shown in Table 4.6. All of the fields scores were quite lower than in the previous experiment which is quite expected since the patterns vary between the templates. However, it was not expected for *YearlyPremium* to have a score of 0.0 since the keywords corresponding to the values would help with the classification of the field. *YearlyCommision* is not included in this experiment since this field was not present in the set, therefore it has the score of 1.0 in tables 4.5 and 4.6.

The accuracy of the fields per JSON object had quite the drop in performance also where *YearlyPremium*, like with the F_1-score, had the worst performance. This is shown in 4.5. Since the accuracy of the fields per file is fairly low it results in an accuracy of 0.0 in total files that are classified as correct.

Model	Accuracy
Baseline (MLM)	0.0
BiLSTM-CRF (Char)	0.0

Table 4.4: Accuracy of the JSON objects from previously unseen templates

Field	Baseline (MLM)	BiLSTM-CRF (Char)
InsuranceNumber	0.108	0.486 ± 0.009
BusinessId	0.090	0.552 ± 0.310
BusinessName	0.114	0.651 ± 0.036
InsurancePeriodStart	0.000	0.128 ± 0.028
InsurancePeriodEnd	0.000	0.206 ± 0.011
YearlyPremium	0.000	0.000
YearlyComission	1.000	1.000

Table 4.5: Accuracy of the fields per JSON object from previously unseen templates

Field	Baseline (MLM)	BiLSTM-CRF (Char)
InsuranceNumber	0.110	0.568 ± 0.046
BusinessId	0.000	0.552 ± 0.340
BusinessName	0.116	0.501 ± 0.052
InsurancePeriodStart	0.000	0.122 ± 0.008
InsurancePeriodEnd	0.000	0.208 ± 0.009
YearlyPremium	0.000	0.000
YearlyComission	1.000	1.000

Table 4.6: F_1 on the fields from previously unseen templates

Chapter 5

Discussion

Evaluation

The models with the simple character embeddings showed better results overall and the CRF-layer did indeed improve consistency and reduced variability. To make sure we got the performance of how the models would be in practice we trained and evaluated the models 10 times on the previously seen templates. One inconvenient part of our evaluation is that it is quite the strict evaluation because it penalizes the field completely if one character in the field is incorrectly classified. The models might be better at finding the patterns of the fields than what the results show, just that it might need more fine-tuning with regex substitution of characters that are incorrectly classified i.e. the field "10 0O0". This problem is most likely due to our approach of automatic data extraction discussed in 3.3.

An additional evaluation method could have been added to gain more of a balanced evaluation into how the models perform, one such method would be to look at the classified field and calculate the edit distance to the correct field based on a heuristic. The accuracy of the whole document is the strictest evaluation, if one field is incorrectly classified in the document then that document is in turn classified as incorrect. The fields with the worst performance will significantly lower this part of the evaluation. This is a valuable aspect to evaluate still, as one would need to check the field in each JSON object with the corresponding machine-readable file that has been incorrectly classified. If the amount of files that have been incorrectly classified is relatively large in relation to the set then this task of correcting each file could become tedious and time-consuming as well. Fortunately, information extraction tasks do not usually have tasks that require the extraction of far too many fields as we can see with the examples addressed in chapter 2.

One reason for the bi-char embeddings not performing well compared to it's simpler counterpart might be because it adds more specific detail to fields which may inhibit the models from generalizing better, the risk for overfitting in the models increases. Since embedding space is larger it also puts more strain on the hyper-parameters, the models' parameters have to be tuned differently compared to the single character embeddings counterpart.

Generalization

The results show that the system performs well on the templates that it had previously been trained on. We see in our current implementation that some fields still have patterns that are recognized in the unseen documents, but as we see with *YearlyPremium* it is not. We believe the reason for the date fields not performing so well in the general case is due to different formatting in the templates. Even if the system does not generalize I still think this solution is feasible for this task due to its simplicity and good performance on previously seen templates. In section 3.1 we showed the tables that displayed the distribution of templates in our data set, we think one of the reason for our models not generalizing is due to lack of data. The data set is relatively small and not as diverse as one would have preferred. Annotating each document took quite some time and producing a larger amount of data manually would have taken a lot of time away from developing our models. It is not simple to compare the generalization of our models with other studies either as data sets that go into researches like these are not made publicly available.

Approach

In figure 4.1 we see that the BiLSTM-CRF model still shows signs of improvement when more training data was added. We mentioned that our belief behind the lack of generalization in our models was due to sparse data. One might question the experiment with deep learning in this research because of this. However, we still find some value in this research. Rule and pattern-based approaches only work on previously seen templates and might achieve good performance despite having a small amount of data available. In our research, we have studied the performance of simple neural network architectures with relatively sparse data and found that it is effective on previously seen templates as well. Deep learning alleviates the problem of requiring human operators for creating complex rules, this was one of the main reasons for choosing to experiment with deep learning. Perhaps a combination of deep learning and a rule and pattern-based approach would have yielded a greater performance in general.

Chapter 6

Conclusion

We presented our goals for this project in the form of research questions. Each question was addressed in the project and gave us some insight into the area of Information Extraction with Deep Learning. We chose to work with variants of Recurrent Neural Networks and found that the BiLSTM-CRF model gave us an Accuracy of 0.728 ± 0.007 on previously trained templates. We also saw that we still could have needed more data to make the model even more effective but that our approach did not work as well on previously unseen templates, meaning that it did not generalize. I believe that better results could have been achieved given more time to collect quality data and fine-tune the system. For simpler tasks like this, I would argue that neural networks like we experimented with in this thesis are a viable option. A previous study with a similar approach achieved greater results performance-wise [Palm et al., 2017], one of the reasons for this might be because they had access to a larger amount of data.

There are more advanced networks that have shown promising results in recent years but are less available and harder to configure due to higher costs. However, as different methods become more accessible and cheaper it is likely that our approach may become obsolete.

6.1 Future work

This system is only focused on seven fields but it could be extended to any number of fields given that they are annotated. That is the reason why I chose to see how well it performed on a field like *BusinessName*, where the field is mostly characters and not numbers. S&P does not necessarily use fields like this in their Customer Relationship Management platform but it was an interesting experiment that showed good results.

One immediate approach to improve the models is to experiment further with the hyper-parameters and the overall architecture, which could improve the results in both previously seen and unseen documents. Some suggestions would be to experiment with the batch sizes, learning rate and the number of epochs. This is a very time-consuming and naive approach to improve the performance and may only improve the results by a bit.

To improve the generalization of the models one could create features, add lists of values for the different fields such that it could find the fields better. An example is adding a list of different organization names in Sweden or adding names of the months such that the model can find the dates which have a format such as 21-April-2020. To increase the performance in extracting the fields one could incorporate rules for which candidate should be chosen like in the traditional approach, fields can be assigned different probability scores and the one with the highest score is then chosen. For example, if there are several *YearlyPremium* candidates then there could be a rule that chooses the one candidate that adds to a total sum given that several premium fields are present like with *YearlyCommision*.

With companies like S&P, it is not unusual for companies to e-mail insurance documents and their respective machine-readable format, if such e-mail is accessible and usable then it could alleviate a lot of the annotation time for the data sets with increased quality. This was not possible in our case but might be good to consider for future cases.

As mentioned in the introduction, Information Extraction is an open problem. We did not achieve as great results in regards to our third research question since the performance on the unseen templates was low. However, we believe this task can be solved efficiently with our chosen architecture and even more advanced ones that are state-of-the-art today. One hope is especially that more data becomes publicly available such that more direct comparison in Information Extraction with Deep Learning can be made.

References

[JSO,] Introducing json. `https://www.json.org/json-en.html`. Accessed: 2010-09-30.

[Adobe Systems Inc, 1999] Adobe Systems Inc, C. (1999). *PostScript language reference*. Addison-Wesley Longman Publishing Co., Inc.

[Akbik et al., 2019] Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019). Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

[Arabnia and Tran, 2016] Arabnia, H. R. and Tran, Q. N. (2016). *Emerging trends in applications and infrastructures for computational biology, bioinformatics, and systems biology: systems and applications*. Morgan Kaufmann.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, Neural Networks, IEEE Transactions on, IEEE Trans. Neural Netw*, 5(2):157 – 166.

[Chanyeong and Alan, 2002] Chanyeong, K. and Alan, C.-M. (2002). Multinomial logistic regression. *Nursing Research*, 51(6):404.

[Chaudhuri et al., 2017] Chaudhuri, A., K Ghosh, S., SpringerLink (Online, s., Mandaviya, K., and Badelia, P. (2017). *Optical Character Recognition Systems for Different Languages with Soft Computing*. Studies in Fuzziness and Soft Computing: 352. Springer International Publishing.

[Chen et al., 2015] Chen, X., Qiu, X., Zhu, C., Liu, P., and Huang, X.-J. (2015). Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

[Chicco and Jurman, 2020] Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6.

[Chiticariu et al., 2013] Chiticariu, L., Li, Y., and Reiss, F. (2013). Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832.

[Chollet et al., 2018] Chollet, F. et al. (2018). *Deep learning with Python*, volume 361. Manning New York.

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

[Fu et al., 2021] Fu, J., Liu, J., and Shi, W. (2021). Exploiting named entity recognition via pre-trained language model and adversarial training. *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), 2021 IEEE International Conference on*, pages 665 – 669.

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). A theoretically grounded application of dropout in recurrent neural networks.

[Gers et al., 2000] Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451 – 2471.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. MIT Press.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610.

[Hastie and Tibshirani, 2009] Hastie, T. and Tibshirani, R. (2009). & friedman, j.(2008). the elements of statistical learning; data mining, inference and prediction.

[Hecht-Nielsen, 1992] Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735 – 1780.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Korenius et al., 2004] Korenius, T., Laurikkala, J., Järvelin, K., and Juhola, M. (2004). Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM

'04, page 625–633, New York, NY, USA. Association for Computing Machinery.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[Lample et al., 2016] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Liu et al., 2018] Liu, L., Shang, J., Ren, X., Xu, F., Gui, H., Peng, J., and Han, J. (2018). Empower sequence labeling with task-aware neural language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Medvet et al., 2011] Medvet, E., Bartoli, A., and Davanzo, G. (2011). A probabilistic approach to printed document understanding. *International Journal on Document Analysis & Recognition*, 14(4):335 – 347.

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

[Palm et al., 2017] Palm, R. B., Winther, O., and Laws, F. (2017). Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 406–413. IEEE.

[Pedamonti, 2018] Pedamonti, D. (2018). Comparison of non-linear activation functions for deep neural networks on mnist classification task. *arXiv preprint arXiv:1804.02763*.

[Ramshaw and Marcus, 1995] Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning.

[Rusinol et al., 2013] Rusinol, M., Benkhelfallah, T., and d'Andecy, V. (2013). Field extraction from administrative documents by incremental structural templates. *2013 12th International Conference on Document Analysis and Recognition, Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, Document Analysis and Recognition, International Conference on*, pages 1100 – 1104.

[Schuster et al., 2013] Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C., Aliyev, K., and Hofmeier, A. (2013). Intellix – end-user trained information extraction for document archiving. *2013 12th International Conference on Document Analysis and Recognition, Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, Document Analysis and Recognition, International Conference on*, pages 101 – 105.

[Sint et al., 2009] Sint, R., Schaffert, S., Stroka, S., and Ferstl, R. (2009). Combining unstructured, fully structured and semi-structured information in semantic wikis. In *CEUR Workshop Proceedings*, volume 464, pages 73–87.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Sukanya and Biruntha, 2012] Sukanya, M. and Biruntha, S. (2012). Techniques on text mining. In *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pages 269–271. IEEE.

[Thompson, 1968] Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422.

[van der Knijff, 2009] van der Knijff, J. (2009). Adobe portable document format. *Inventory of long-term preservation risks, v0*, 2:20–56.

[Yang et al., 2017] Yang, X., Yumer, E., Asente, P., Kraley, M., Kifer, D., and Lee Giles, C. (2017). Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-11-10

EXAMENSARBETE Information Extraction on Insurance Policy Letters using deep learning
STUDENT Anujan Balasingam
HANDLEDARE Marcus Klang (LTH)
EXAMINATOR Elin-Anna Topp (LTH)

Informationsutvinning på försäkringsbrev med hjälp av djupinlärning

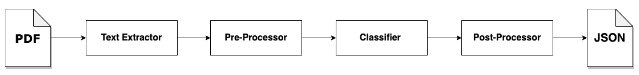
POPULÄRVETENSKAPLIG SAMMANFATTNING **Anujan Balasingam**

Hantering av affärsdokument så som försäkringsbrev kan vara en avskräckande uppgift, särskilt när detta görs manuellt och volymen är hög. Detta arbete utforskar hur processen kan skötas automatiskt med hjälp av djupinlärnings-tekniker.

Företag extraherar information (data) från affärsdokument och lagrar dem i IT -system för att hålla reda på sina affärsavtal. Extraheringen av data utförs, i traditionnellt avseende, manuellt av de anställda. I takt med att dokumentmängden ökar blir manuell extrahering till slut en ohållbar process. Det är en ineffektiv och arbetskrävande uppgift med risk för mänskliga misstag. Datat som lagras i dessa IT-system är i strukturerad form, detta betyder att formatet på datat överenstämmer med format som används i relationsdatabaser, datat som finns i affärsdokumenten behöver omvandlas då det inte ärav denna typ. Genom att låta maskiner sköta denna process av informationsutvinning och omvandling kan de anställda fokusera mer på andra uppgifter som kan hjälpa företaget att växa.

I mitt examensarbete har jag utforskat hur djupinlärnings-modeller kan användas för informationsutvinning från försäkringsbrev som är i PDF-format. Experimentet är uppdelat i två delar. Det första experimentet går ut på att utvinna information från försärkringsbrev med samma utseende som mina modeller har tränats på, s.k. sedda mallar, och det andra går ut att utvinna information från försäkringsbrev med annorlunda utseende, s.k. osedda mallar, gentemot vad den har tränats på. Jag har experimenterat med en enkel maskininlärnings-modell, Logistic Regression, för att få en uppfattning om utmanningarna i studien samt utforskat varianter av det neurala nätverket LSTM.

En allmän pipeline för båda modeller kan ses i figuren nedan.



Modellernas prestation utvärderas på tre olika sätt, en av utvärderingsmetoderna som ger oss en inblick på en högre nivå över hur modellerna presterar är de extraherade filernas noggranhet. Den högsta poängen i filernas noggrannhet, som är vår striktaste utvärderingsmetod, på sedda mallar är 0,728. På osedda mallar var poängen 0,0. Resultaten visar att enkla metoder kan vara effektiva för att extrahera information från olika mallar för dokument som den har tränats på men att den behöver mer finjustering för att förbättra generaliseringen.