Mittuniversitetet
MID SWEDEN UNIVERSITY

Artificial intelligence application for feature extraction in annual reports

AI-pipeline for feature extraction in Swedish balance sheets from scanned annual reports

Jesper Nilsson

Bachelor's thesis – Final project
Main field of study: Computer engineering
Credits: 15
Semester/year: Spring, 2024
Supervisor: Qing He
Examiner: Stefan Forsström
Course code: DT099

At Mid Sweden University, it is possible to publish the thesis in full text in DiVA (see appendix for publishing conditions). The publication is open access, which means that the work will be freely available to read and download online. This increases the dissemination and visibility of the degree project.

Open access is becoming the norm for disseminating scientific information online. Mid Sweden University recommends both researchers and students to publish their work open access.

I/we allow publishing in full text (free available online, open access):

☒ Yes, I/we agree to the terms of publication.

☐ No, I/we do not accept that my independent work is published in the public interface in DiVA (only archiving in DiVA).

Sundsvall 2024-06-06

.....

Location and date

Degree of Bachelor of Science with a major in Computer Engineering

.....

Programme/Course

Jesper Nilsson

.....

Name (all authors names)

1997-02-11

.....

Year of birth (all authors year of birth)

Abstract

The persistence of unstructured and physical document management in fields such as financial reporting presents notable inefficiencies. This thesis addresses the challenge of extracting valuable data from unstructured financial documents, specifically balance sheets in Swedish annual reports, using an AI-driven pipeline. The objective is to develop a method to automate data extraction, enabling enhanced data analysis capabilities. The project focused on automating the extraction of financial posts from balance sheets using a combination of Optical Character Recognition (OCR) and a Named Entity Recognition (NER) model. TesseractOCR was used to convert scanned documents into digital text, while a fine-tuned BERT-based NER model was trained to identify and classify relevant financial features. A Python script was employed to extract the numerical values associated with these features. The study found that the NER model achieved high performance metrics, with an F1-score of 0.95, demonstrating its effectiveness in identifying financial entities. The full pipeline successfully extracted over 99% of features from balance sheets with an accuracy of about 90% for numerical data. The project concludes that combining OCR and NER technologies could be a promising solution for automating data extraction from unstructured documents with similar attributes to annual reports. Future work could explore enhancing OCR accuracy and extending the methodology to other sections of different types of unstructured documents.

Keywords: Artificial intelligence, Feature extraction, Named Entity Recognition, BERT, Optical Character Recognition, financial documents

Sammanfattning

Hantering av ostrukturerade och fysiska dokument inom vissa områden, såsom finansiell rapportering, medför betydande ineffektivitet i dagsläget. Detta examensarbete fokuserar på utmaningen att extrahera data från ostrukturerade finansiella dokument, specifikt balansräkningar i svenska årsredovisningar, genom att använda en AI-driven pipeline. Syftet är att utveckla en metod för att automatisera datautvinning och möjliggöra förbättrad dataanalys. Projektet fokuserade på att automatisera utvinning av finansiella poster från balansräkningar genom en kombination av Optical Character Recognition (OCR) och en modell för Named Entity Recognition (NER). TesseractOCR användes för att konvertera skannade dokument till digital text, medan en BERT-baserad NER-modell tränades för att identifiera och klassificera relevanta finansiella poster. Ett Python-skript användes för att extrahera de numeriska värdena som är associerade med dessa poster. Projektet fann att NER-modellen uppnådde hög prestanda, med ett F1-score på 0,95, vilket visar dess effektivitet i att identifiera finansiella poster. Den fullständiga pipelinen lyckades extrahera över 99% av posterna från balansräkningar med en träffsäkerhet på cirka 90% för numerisk data. Projektet drar slutsatsen att kombinationen av OCR och NER är en lovande lösning för att automatisera datautvinning från ostrukturerade dokument med liknande attribut som årsredovisningar. Framtida arbeten kan utforska att förbättra träffsäkerheten i OCR och utvidga utvinningen till andra sektioner av olika typer av ostrukturerade dokument.

Nyckelord: Artificiell intelligens, Datautvinning, Named Entity Recognition, BERT, Optical Character Recognition, finansiella dokument

Acknowledgements

I would like to express my gratitude to the individuals and organizations that have supported me throughout this thesis.

Firstly, I extend my thanks to my supervisor at Bolagsverket, Johannes Lindén, for presenting this project and for his assistance in setting up the necessary resources for this project's execution. His feedback and guidance have been valuable in the successful execution of this project.

I would also like to thank the AI-hubben department at Bolagsverket for their feedback and ideas during the project.

Furthermore, I am grateful to my university supervisor, Qing He, for her guidance throughout the creation of this project and the writing of this thesis. Her insights and guidance have been greatly appreciated.

A Thank you to all for your contributions and support during this project.

Table of Contents

Terminology

Acronyms/Abbreviations

AI	Artificial intelligence
Corpus	A dataset for training or testing in AI
CSV	Comma separated values, a common file format with values structure by commas
Inference	In AI, inference is the process for the model to make predictions / generate data
JSON	JavaScript Object Notation
K2	Rulework for financial reporting in annual reports for smaller companies
NA	Not Available
NER	Named entity recognition
NLP	Natural language processing
OCR	Optical Character recognition

Mathematical notation

F1-score $\frac{2 * TP}{2 * TP + FP + FN}$

FN = False negatives

FP = False positives

TP = True positives

1 Introduction

This project was suggested by Bolagsverket that aims to work on the processing of unstructured data and do research and development of using Artificial intelligence in this field. This chapter will introduce the background, problem, research questions and scope of this project.

1.1 Background and motivation

In today's digital climate, where vast amounts of data can be handled and analyzed rapidly, the persistence of physical and unstructured document management in certain fields presents a notable inefficiency leading to possible loss of insights into this data. Financial documents and specifically annual reports are such a document where about 50% of these are still sent through physical mail and handled by the Swedish Companies Registration Office (Bolagsverket). These annual reports, aside from being in physical format, are also not required to follow a standardized structure creating difficulties in efficiently processing and analyzing the data contained within these documents. The Swedish Companies Registration Office (Bolagsverket) wishes to find ways to extract relevant data from these unstructured documents as this could help them with automation in their processing and create the ability to analyze the data to enhance crime prevention by identifying unusual behavior. They could also offer better services to their customers who request these public documents and improve their understanding of their clients by creating profiles with more parameters than are currently available. By making more data accessible and analyzable, it could not only achieve more efficient data handling across numerous fields but also unlock insights from a vast array of data previously left unexamined. This could significantly enhance financial transparency and possibly crime prevention, as well as reveal trends in the various fields with large amounts of data, for example the medical field, among many others.

1.2 Overall aim and problem statement

The overarching aim and goal of this project is to find ways to extract more data and information from unstructured documents previously locking away an efficient use of data in their contents. The goal is to find out through the use of AI-tools if it is possible to extract valuable data that could be used for analysis and/or automation. While this project will focus on extracting financial data from annual reports, the hope is that

through the methods and tools explored during this project, these could be configured to help extract valuable data in a myriad of different types of unstructured documents, whether it be for financial data, medical data or any other valuable data currently being inefficiently analyzed due to limitations of its unstructured and/or physical format. The focused problem and approach tackled during this project will be through the creation of an AI-driven pipeline to extract financial features from balance sheets in annual reports.

1.3 Research questions

The first two research questions are more specific quantitative metrics on the pipeline that will be constructed during this process while the third question is a broader question about extrapolating the results of the previous questions to gain some scientific knowledge that could be generalized from this project.

1. **How many features can be extracted using the constructed pipeline?**

   There are several financial posts in balance sheets, this question aims to answer how many of them can be extracted through the constructed AI-driven pipeline. The goal is to be able to extract the vast majority of them.

   This question can help determine if an adequate amount of data is able to be extracted using an AI-pipeline, to be able to use for analysis or automation. The amount of data extracted can then be compared to how many financial posts were actually available in the report.

2. **How accurate are the numbers extracted?**

   Since the visual extraction can make mistakes it's important to measure that the numbers connected to a feature are accurate to reality, this could be affected by the visual-AI making mistakes in either mistaking a number for another or structuring the data wrong, so the numbers extracted don't accurately represent the feature.

   It's important to have a high accuracy if insights into the numerical data is the part of the goal.

3. **Are the methods of the constructed pipeline an efficient solution for feature extraction in general unstructured documents?**

   This research question asks to evaluate the knowledge gained from the constructed pipeline to ask the question if constructing a similar pipeline for another use-case could be an efficient method of feature extraction in another type of document.

   This question can help to determine if the knowledge gained during this project is very highly specific to balance sheets in Swedish annual reports or if it could also possibly be generalized to other documents.

1.4 Scope

While the aim of this project is to explore the possibility of developing a general pipeline methodology that could be adaptable to various types of documents, due to time constraints, the scope will be specifically limited to the extraction of financial data from balance sheets in annual reports. Moreover, this project will concentrate on reports adhering to the K2 rulework of Swedish annual reports.

The K2 rule work is for "smaller" companies which have less than 50 employees, less than 80 million Swedish crowns in net sales and less than 40 million Swedish crowns in their balance sheet. The K2 rule work is the reporting rulework used by the majority of Swedish companies.

Focusing on the rulework the majority of companies adhere to avoids the possible additional complexities that alternative reporting rules might introduce which time-limitations might stop the project from addressing.

Image quality improvement is beyond the scope of this project. Although enhancing image quality could potentially increase the accuracy of data extraction from documents with significant noise, the dataset intended for this project is of reasonably good quality, with only a few exceptions. Accurate denoising of unstructured images, each with unique noise structures, could require significant work.

For the visual aspect of data extraction, this project will use existing frameworks and models for Optical character recognition., with only minor configurations. No focus will be dedicated to building or training AI models for visual extraction. Some degree of programming will be

done to transform the extracted data into a format suitable for further processing by the language model.

Significant attention will be towards the language component of the pipeline. The project aims to enhance or fine-tune a Language Model for efficient data extraction and structuring into identifiable features, each associated with one or more numerical values.

In essence, the scope consists of configuring and utilizing pre-existing tools for the initial stages of visual data extraction, followed by the creation and training of a fine-tuned Language model designed to process this data. Additionally, programming and scripting will be done for connecting the models and managing data flow throughout the extraction process, lastly structuring the output generated by the pipeline.

1.5 Outline

Chapter 2 describes the Theory in the area and will introduce basic knowledge on the scientific and technical areas with relation to this project. Chapter 3 (Method) will introduce the scientific methods and present how this project will be performed and evaluated. Chapter 4 (Approach) will explore and present possible approaches for constructing the project and compare these while lastly presenting the chosen approach for this project. Chapter 5 (Implementation) will present the different parts of the project and in some detail explain how the different parts were constructed, the coding required for implementing the different building blocks of the project will be presented here. Chapter 6 (Results) will present the datasets and their contents, the metrics from different parts of the pipeline as well as metrics on the full pipeline. Some examples of balance sheets processed through the pipeline will also be presented here. Chapter 7 (Discussion) will give the authors perspective and analysis on the results presented in the previous chapter and discuss the method, ethical considerations and lastly give a recommendation to Bolagsverket. Chapter 8 (Conclusions) will Aim to answer the research questions asked in the introduction chapter and give an overall conclusion of the project's result based on all previous sections. Lastly a section covering some suggested future works will be presented.

1.6 Division of work

All new work and results that were created during this project will have been made by the author of this report. Some datasets from a previous project (SIMS 2023), the author was participating in may be used. During the project: feedback, ideas and guidance from supervisor Johannes Lindén at Bolagsverket and Qing He at Mid Sweden University was also given.

2 Theory

This theory chapter goes through and provides introductions to what AI is and some fields that can be used in processing unstructured documents. The chapter then gives some general introductory knowledge on OCR and examples of tools in this area. The next section will be on language models and named entity recognition and tools used in this area. Lastly, the theory chapter will present some related work in the area ranging from similar work to research and projects in the surrounding areas.

2.1 The area of Artificial intelligence

Artificial intelligence (AI) is a technology that enables machines to perform tasks traditionally requiring human intelligence, such as learning, reasoning, and problem-solving. The field of AI was formally established in 1956 during the Dartmouth Conference, where initial research focused on using AI to solve algebraic problems and prove geometric theorems. Since then, AI has expanded in scope and application, developing across many sectors. Technologies like neural networks have advanced quickly, leading to rapid developments in areas such as image recognition and natural language processing. [1]

2.2 Artificial intelligence in processing of unstructured documents

In a systematic literature review on artificial intelligence in processing unstructured documents published in IEEE Access 2021 [2], the authors identify three main areas of methods where artificial intelligence has been and can be utilized. These areas are optical character recognition (OCR), robotic process automation (RPA), and named entity recognition (NER). These three areas all involve artificial intelligence approaches. The forthcoming sections will focus on OCR and NER, as they are the main areas utilized in this project.[2]

2.3 OCR

The following section will introduce OCR and some OCR tools in relation to processing physical and unstructured documents:

As a first step to automating and processing unstructured documents, organizations began implementing Optical Character Recognition, which helped digitize documents into digital text. While these tools

enabled organizations to convert their physical documents into text, they were often limited in their early stages. For example, early OCR was often limited to one type of text font. [2] As the development of OCR has continued, new techniques have been developed, and machine learning and artificial intelligence are often utilized in character recognition and classification. Common techniques include Naive Bayes classification, Support Vector Machine, and neural networks. [2]

2.3.1 PaddleOCR

PaddleOCR is an open-source Python library that offers OCR tools for both applying and training models. PaddleOCR supports multiple models for over 80 languages, including Swedish. The library provides frameworks for OCR and layout analysis through PP-Structure. PP-Structure also offers functions for extracting tables from images and converting these into Excel documents. [3]

2.3.2 Tesseract

Tesseract is an OCR-engine first developed in 1985 as a proprietary tool by Hewlett-Packard Co which was then open sourced in 2005 with Google picking up development in 2006. Since version 4, Tesseract uses a neural network to do OCR.[4]

2.4 Language models

The coming sections will cover language models and tasks that language models can perform as well as some popular frameworks, tools, models and evaluation metrics for language models.

Language models are used in natural language processing (NLP) to perform a varied amount of tasks relating to human language. Examples of these tasks include grammatical correction, information retrieval, text generation, and more. Language models are often trained on large corpora, which are collections of texts. They are frequently used for word prediction, taking a sequence of words and predicting the next word in the sequence.[5]

2.4.1 Transformers

Transformers have become the most dominant architecture for today's language models due to their ability to surpass earlier models that used convolutional and recurrent neural networks. This superiority is because the transformer architecture scales with training data and model size and

can be easily adapted to specific tasks with strong performance [6]. In Figure 2 the daily unique downloads of some popular transformer models from huggingface.co are shown, showing that over 30,000 transformer models are downloaded daily, with GPT and BERT models being popular models in the field.

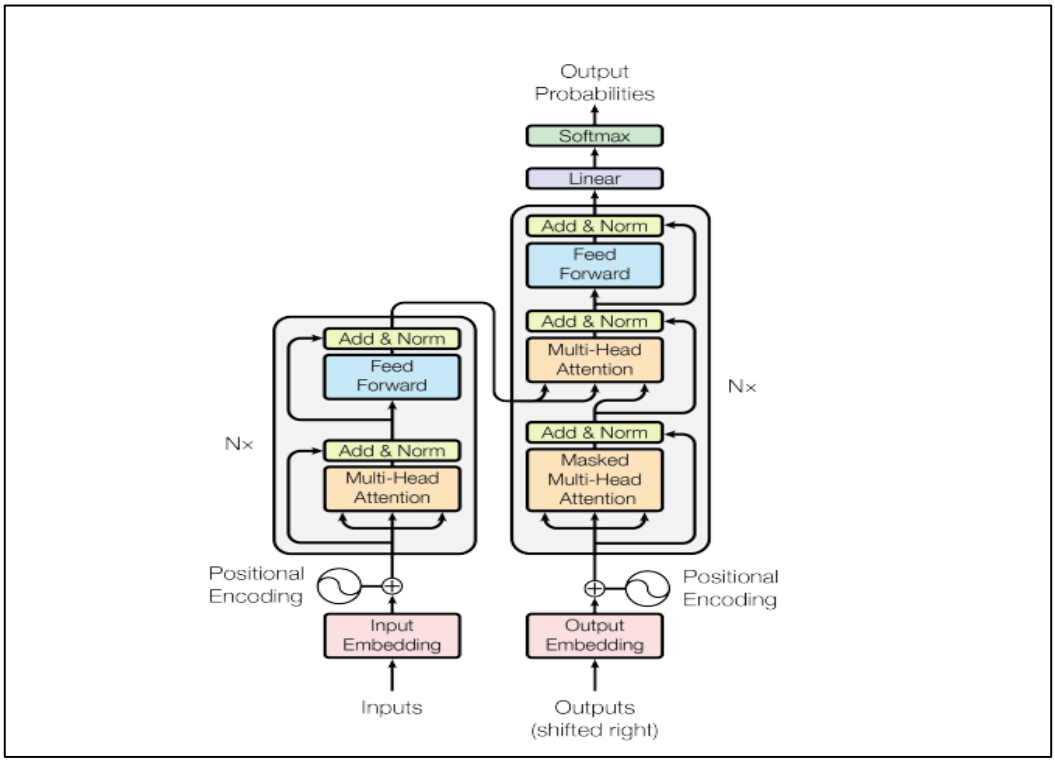


Figure 1: Transformer model architecture from original paper.[7]

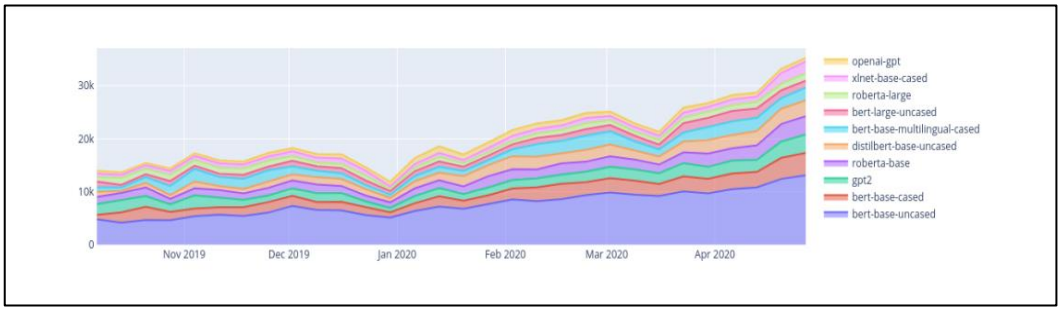


Figure 2: Daily downloads of most popular models from huggingface.co [6]

The transformer architecture was first proposed by Google in the paper 'Attention is All You Need' [7]. This architecture abandoned traditional

recurrence and convolution methods and instead relied on the 'attention' mechanism. The attention mechanism works by allowing the model to focus on specific elements while filtering out those of less relevance, by assigning attention weights. [8, p. 45-48].

The transformer uses an encoder-decoder architecture, as can be seen in Figure 1, which shows the encoder on the left and the decoder on the right. The encoder takes an input sequence and maps it onto a new continuous sequence, which is then processed by the decoder. The decoder generates an output one element at a time. Each step in the output takes the previously generated symbol as a new input [7].

2.4.2 Tokenizers

In natural language models and natural language processing, tokenization is used to convert text into smaller units called tokens. An example method of tokenizations is WordPiece.[9]

WordPiece is a subword tokenization technique used in models like BERT. It starts with all characters as its base tokens and merges pairs of tokens to create smaller and more efficient sets of tokens based on common subwords. This method helps to manage the trade-off between the amount of tokens possible and the length of token sequences. It segments text using a left-to-right, longest-match-first strategy.[9]

After tokenization each token is given a numerical id through a lookup-table to create a format of data the language model can work with [10].

2.4.3 Named Entity Recognition

Named Entity Recognition (NER) is a part of the field of Natural Language Processing (NLP) and is used to identify and classify types of named entities in text. Common types of named entities include names, locations, and times, which are identified and classified within unstructured text. In Figure 3, an example is shown where NER has identified and classified words/tokens that belong to the entity classes 'Person' and 'Location'.

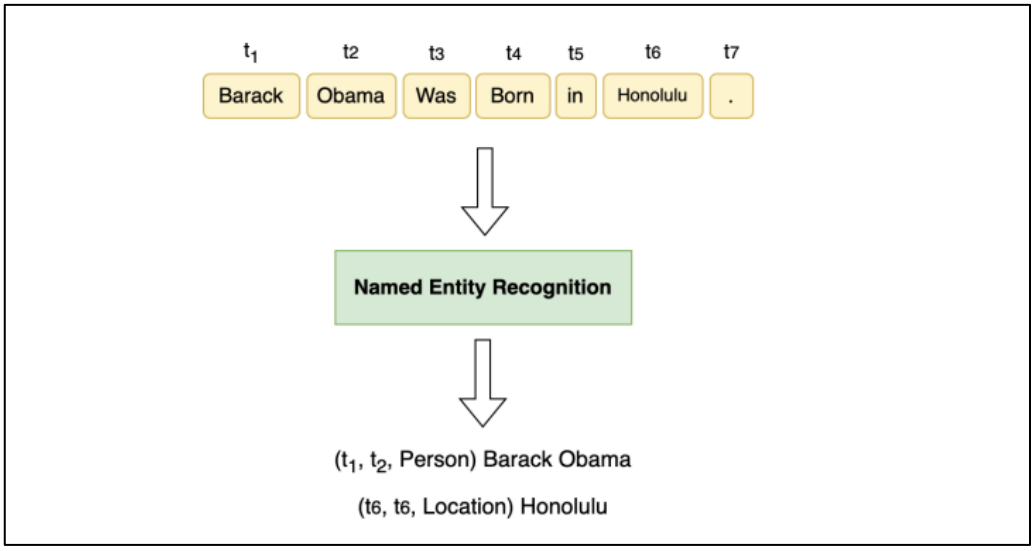


Figure 3: Named entities extracted from tokenized text.[6]

NER systems have existed for quite some time, and many different systems using various approaches have been developed. Early systems often used a rule-based approach, utilizing rules, lexicons, and spelling features to identify and classify entities. Recent developments in NLP and NER utilize artificial intelligence, with research and developments focusing on the use of neural networks, deep learning, and transformers.[11]

One of the more modern developments in NER is the use of language models as they are now a significant part of natural language processing. Many language models are built using the transformer architecture. Transformer models can be used either directly by themselves for NER or to create the input for other methods. The survey 'A Survey on Recent Advances in Named Entity Recognition' mentions BERT and variations of the BERT model (Roberta, DistilBERT) as showing high efficiency in utilization for named entity recognition.[11]

2.4.4 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a model first created by Google in 2018 [12] based on the original transformer implementation 'Attention Is All You Need' [7]. It is designed to be pre-trained in a bidirectional manner on unlabeled text at every layer, meaning that text context is considered from both left to right and right to left. Unlike the original transformer, BERT only uses the

encoder part of the encoder-decoder stack and includes more and larger encoder layers [13, p. 61-65]. This bidirectional approach contrasts with other common language models such as the Generative Pre-trained Transformer (GPT), which is unidirectional, built on a left-to-right architecture. The BERT model reduces a significant amount of engineering and training on specific tasks while still achieving high performance on downstream sentence and token-level tasks.

BERT is typically pre-trained on two unsupervised tasks:

- **Masked LM:**

  A percentage of the input tokens are randomly chosen and masked with either a special [MASK] token (80% of the time), randomly replaced by another token (10% of the time), or not replaced at all (10% of the time). Then, the model is asked to predict the original token.[12]

- **Next Sentence Prediction (NSP):**

  NSP is used to train an understanding of the relationship between sentences. The model is given two sentences (A and B) and asked to determine if sentence B was the next sentence after sentence A. In 50% of cases, B was the following sentence, and in 50% it was a random sentence from the training corpus.[12]

2.4.5 KB BERT

KB BERT is a BERT model trained for the Swedish language, developed by KBLab at the National Library of Sweden. The paper 'Playing with Words at the National Library of Sweden - Making a Swedish BERT' compares their model, KB BERT, to other models such as the one developed by the Swedish Public Employment Service (Arbetsförmedlingen) as well as Google's multilingual M-BERT. The results of the paper show that KB BERT outperforms these in multiple NLP tasks such as NER and part-of-speech tagging (POS).[14]

The KB BERT model was trained on a Swedish corpus which focused on modern Swedish language and the corpus consisted of the dataset that can be seen in table 1. The corpus includes digitized newspapers, official reports of the Swedish government, legal e-deposits, social media comments, and Swedish Wikipedia articles.[14]

Table 1. Corpus for KB BERT [9]

Corpus	Words	Sentences	Size
allnews.txt	2997 M	226 M	16783 MB
sou.txt	117 M	7 M	834 MB
e-deposit.txt	62 M	3.5 M	400 MB
social.txt	31 M	2.2 M	163 MB
sw-wiki.txt	29 M	2.1 M	161 MB
Total	**3497 M**	**260 M**	**18341 MB**

2.4.6 Finetuning a language model, huggingface transformers

The Hugging Face Transformers library is an open-source library designed to provide access to pre-trained models for building and experimenting with [6]. The Hugging Face libraries offer capabilities to download models, set up datasets, and train models. The Trainer class in the Transformers library is an API that can be used for complete training in the PyTorch framework. With the help of the TrainingArguments class, many parameters related to training or fine-tuning a model can be modified. A full list of the training arguments can be found in the Hugging Face Transformers documentation. [15]

2.4.7 Evaluation metrics on named entity recognition

Evaluating a language model can be done quantitatively when it comes to classification tasks. This is typically accomplished by using a subset of the annotated dataset as a test set, which the model should not have been trained on. The model is then tested by making predictions on this test set and comparing these predictions to the annotations to determine how often the model gives the correct output. This measurement is called 'accuracy', or 'error rate' when the inverse is measured [16, p. 103-104].

Although accuracy is a simple measurement that can provide a good indication of how well the model performs, it may not be sufficient for all use-cases. For instance, certain types of mistakes may be more consequential than others. For example, as outlined in the book 'Deep Learning' by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, “an e-mail spam detection system can make two kinds of mistakes: incorrectly classifying a legitimate message as spam, and incorrectly allowing a spam message to appear in the inbox. It is much worse to

block a legitimate message than to allow a questionable message to pass through." [16, p. 422-424]

Further, consider a model that always predicts an email as legitimate, if out of 1,000 emails only one was spam, the model would have achieved 99.9% accuracy, even though it missed every spam email. Because accuracy alone does not account for this type of performance, other metrics such as precision and recall are used. Precision is the fraction of classifications that were correct, and recall is the fraction of 'events' that were detected. Using precision and recall, there is a metric called the F-score or F-measure, which is the harmonic mean of precision and recall and can be expressed as seen in equation 1 below.

$$\text{F1-Score} = \frac{2 * TP}{2 * TP + FP + FN} \quad (1)$$

TP = True positives

FP = False positives

FN = False negatives

Another metric and visualization tool for assessing a model's performance is the confusion matrix, which helps visualize the types of errors a model makes. The matrix displays the model's predictions compared to the true values in the test set.[18] This is illustrated in an example of page-classification in Swedish annual reports as seen in figure 4.

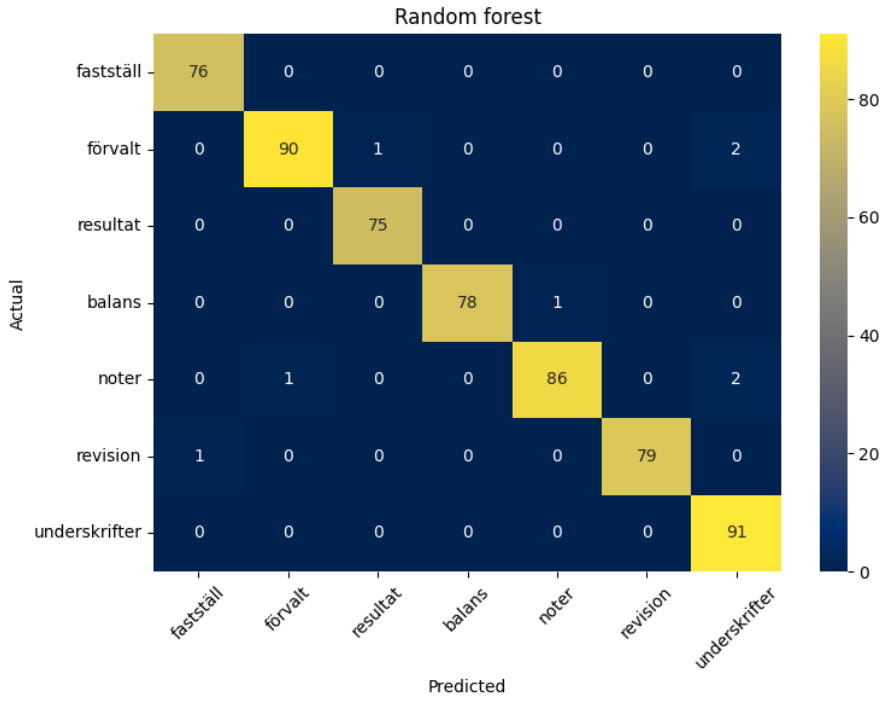


Figure 4: Confusion matrix for text-classification using a random forest model. [19]

2.5 Related work

The following sections introduce 2 works that share similarities with this project and what these similarities are and what differs this project from them. After the 2 first works, a list of other relevant works that are in the surrounding research area will be presented but these have a less clear similar direction as this project.

2.5.1 Reading key figures from annual reports

In her master's thesis from 2021[20], Sara Nordin Hällgren collaborated with AI-hubben at Bolagsverket to extract key figures from Swedish annual reports. The work in her thesis is similar to the work that will be conducted during this project, as both examine methods that could be utilized to extract data from Swedish annual reports. In her thesis, Hällgren focused on extracting four specific key figures from the balance sheet and found that, after OCR, the best method achieved between 89.6% and 92.9% accuracy using a fuzzy string match.

This project will also attempt to extract features from the balance sheet. However, unlike Hällgren's thesis, it will not focus on specific figures but instead will use a language model to try to identify different types of features as some category of financial post.

Hällgren's work also included a larger focus on improving the image quality of the reports to enhance results, which is something this project will not focus on.[20]

2.5.2 RNN based question answer generation and ranking for financial documents using financial NER

In this 2020 article [21], the authors proposed a system for generating a summary report in the form of financial questions and answers about a financial document. Their main contribution was the development of a Named Entity Recognition (NER) model focused on financial entities, which was trained on texts from online financial articles.

The financial NER component of this system shares similarities with what will be done in this project, allowing for a comparison of results. However, there are important differences to note: their work was based on English texts, which generally have larger language models available, and it was not specifically focused on annual reports.

One notable difference is that the article worked directly with digital text, whereas this project will work with text outputted from OCR on images, which introduces potential errors. Additionally, this article used the NER to generate question-answer summaries, a step further than this project will explore.[21]

2.5.3 FiNER

The FiNER project introduces the finer-139 dataset, containing 1.1 million sentences with XBRL tags from SEC filings, focusing on financial numeric entity recognition (NER). The study demonstrates that models like SEC-bert, pre-trained on financial filings, perform well in this task, achieving an F1 score of 82.1% by using numeric pseudo-tokens to handle fragmentation issues.[22]

This work is similar in that it also deals with NER in the finance field using language models, this project does however work on a significantly larger dataset with more specialized tags. Bolagsverket does also work with XBRL-tagging in their digital annual reports, possibly

giving this work more relevance if a focus were ever to be put on those in the future.

2.5.4 FETILDA

FETILDA is a framework created in 2022 for creating embeddings from long financial texts, like annual reports. It splits these documents into smaller chunks and processes each chunk with language models (BERT, FinBERT), and pools the results using a Bidirectional Long Short-Term Memory layer. The framework was tested on 10-K reports from US banks and companies which showed FETILDA outperforms baseline methods in predicting financial metrics and stock volatility.[23]

This work relates closely to using language models on annual reports to be able to extract data for analysis with language models. If future work based on this project were to try to extend the scope to the K3 rule work this paper could reveal insights since the K3 reports are much larger and extensive. The paper presenting Fetilda does however not work on scanned reports, so consideration about structure and OCR errors must be accounted for.

2.5.5 Post-OCR Correction of Digitized Swedish Newspapers with ByT5

The paper presents a model based on the ByT5 model for correcting OCR errors in Swedish newspapers from the 19th and 20th centuries. ByT5 operates at the character level, which helps handle out-of-vocabulary words caused by OCR errors. The model was trained on a mix of newspaper and other text data and achieved a 36% reduction in character error rate (CER).[24]

This work is in the field of digitizing text through OCR and correcting output, which is relevant when dealing with scanned annual reports as well. This project does not focus on the visual parts, they are however important for future improvements and since this project focuses on Swedish text its results might be able to be used in the future for financial documents as well.

3 Methodology

The methodology chapter will present the scientific methods and how the research questions will be answered. The chapter will present the 4 different phases the project has been split into and lastly present the project evaluation method.

3.1 Scientific method description

The scientific method of this project involves the construction, implementation, and training of available tools and models to evaluate their efficiency for a specific use-case. The method will employ quantitative measures to some degree, as there will be clear quantitative measurements that can be made and compared to benchmarks for named entity recognition, although these benchmarks might not involve the exact same entities. While quantitative measurements are utilized, the overarching question, whether the proposed solution is 'good enough' or shows potential for the specific use-case as well as its general potential is inherently more reflective and will be discussed in the discussion chapter.

The first two research questions will be addressed through quantitative measurements while the last one will be a reflective assessment based on the knowledge gained during the project:

- **How many features can be extracted using the constructed pipeline?**

  The output of the constructed pipeline will be compared to a test set with manually counted potential features.

- **How accurate are the numbers extracted?**

  The test set will have been manually checked for correct numbers and will be compared to the pipeline's outputs.

- **Are the parts of the constructed pipeline an efficient solution for feature extraction in general?**

  This question will be based on the results from tests on the earlier questions which will be analyzed and discussed to try to extrapolate some general knowledge.

3.2 Project method description

The methodology for this project will consist of four major parts. The first part is exploratory and experimental in nature, aimed at understanding the problem space and potential solutions and approaches. The second part focuses on the construction and implementation of the necessary tools and models. The third part involves connecting the constructed components and collecting preliminary results as well as collecting measures through testing. The final part entails evaluating the tests on individual components and the entire proposed pipeline to produce a qualitative overview based on the result. These discussions will assess the effectiveness of different parts of the project as well as the overall value of the full pipeline.

3.2.1 First phase: Theory and exploring approaches

During this initial phase, various tools, frameworks, and programming libraries will be researched and experimented with to establish a foundation for subsequent phases. This phase focuses on gathering the theoretical knowledge necessary for constructing the project. Although this phase is not directly connected to the stated goals and questions outlined in the introduction chapter, it is essential for enabling the later stages of construction and result gathering, which are linked to the research questions.

The theoretical framework developed during this phase will cover areas related to processing unstructured documents, including computer vision techniques such as OCR. Various tools in this area, such as PaddleOCR and TesseractOCR, will be explored. Additionally, language models, specifically those related to named entity recognition like KB BERT, will be examined. This phase will also determine the specific categories of features to be included in the project.

3.2.2 Second phase: Construction and implementation

The second phase of this project will focus on construction and implementation, utilizing the results and knowledge gathered during the first phase. This phase will involve constructing the OCR to gather the dataset and for later use in the pipeline. The phase will also include annotating a dataset and start training for NER on a Language model.

For OCR, the libraries PaddleOCR [3] and PyTesseract [25] will possibly be used to implement an OCR engine through Python to extract text from annual reports. The OCR will be developed both to extract text for training data and for use in the completed pipeline.

A dataset of annual reports annotated for NER will be created from reports provided by Bolagsverket. Since this project focuses only on specific pages, a page-classification model from the SIMS 2023 project, developed in collaboration with Bolagsverket, will be used to extract the relevant pages [19]. The open-source tool Label Studio [26] will be employed to annotate the extracted pages with the relevant entities.

The construction and training of the language model will primarily use Python libraries from Hugging Face: Transformers [15] and Datasets[27]. The Datasets library will be utilized to convert the annotated dataset into a format suitable for training a language model. The Transformers library will facilitate the download of a pre-trained language model and tokenizer, which will be used to fine-tune the model on the specific dataset for NER. The likely candidate for this is KB BERT, 'bert-base-swedish-cased,' which will be fine-tuned on the collected dataset for named entity recognition based on the entity categories identified.

3.2.3 Third phase: Connecting constructed parts and gathering results

In this phase the constructed parts from the earlier phase will be connected to create a full pipeline for feature extraction in balance sheets from annual reports. After the Pipeline has been constructed testing on different parts will be conducted and gathered for the results section. The Language model itself will be tested independently to form results and measurements such as accuracy, precision, recall and F1 scores. Confusion matrices will also be visualized as part of the results of the NER-model.

Lastly measurements on the whole pipeline will be made on a manually created test-set to relate to the research questions asked in this project.

3.2.4 Fourth phase: Evaluation and discussion

Based on the results and measurement gathered in phase three evaluation can be compared to the works mentioned in related work as well as common benchmarks for named entity recognition. Discussion

about the generalizability and a qualitative assessment on the metrics will be done.

3.3 Project evaluation method

To assess the overall outcome of the project, an evaluation will be conducted, with some focus on the methodology and structure of this project. This evaluation will involve an examination of both successful aspects and areas for improvement.

The results and measurements gathered will be evaluated in relation to their intended use-case as well as their potential generalizability. This analysis will also evaluate if changes in what the project focused on could have altered its outcome.

4 Approach

This chapter of the report will present possible approaches and analyze them. The approaches are split into 2 main areas, the visual aspect and the language aspect as these should be considered separately but with considerations of their interactions. Lastly a chosen approach will be presented.

4.1 OCR approach alternatives

This section will introduce 3 possible approaches for OCR, compare them and present the chosen approach.

4.1.1 Tesseract, line by line OCR

Previous work on scanned annual reports at Bolagsverket have used tesseract as its main OCR-engine [19][20]. A degree of work has been constructed using tesseract on these reports meaning that there are multiple functions to do the pre-processing, configuring tesseract and post-processing the result to clean the output result. The page-classification models from SIMS 2023[19] are also built atop the output from the configured tesseract-OCR. The Tesseract engine also has a focus on reading text on a line by line basis [4]. Tesseract comes with native support for the Swedish language.

Considering the previous work already made and support for the Swedish language using tesseract to do the OCR is an option for the visual parts of this project getting text output in a clear line by line basis to then be processed.

4.1.2 Paddle OCR / PPocr

Using paddleOCR is a newer approach that hasn't been explored as much for use on Swedish annual reports. Preliminary tests during this project show PaddleOCR having a higher accuracy on character-by-character basis and especially considering accuracy on numbers compared to tesseract. PaddleOCR as default outputs clusters of texts based on their distance from each other, whether this be a single word, single number or a whole line of text [3]. The PPocr model which integrates with the paddleOCR library has support for the Swedish language.

Using paddleOCR as an approach during the project could be a possibility since it has native support for Swedish language. For this approach, some work needs to be made to structure the output. The output would need structuring that could be used in the later parts of processing the text for the language model. PaddleOCR also allows training and fine-tuning models within its framework [3].

4.1.3 PPstructure layout analysis/ table extraction.

PPstructure is a part of the paddleOCR library but rather than only focusing on text it allows for also extracting its structure and extracting tables and converting them into excel sheets [3]. PPstrcuture does not have native support for Swedish, currently only supporting English and Chinese.

Using PPstructure as an approach during this project would allow for more data than just pure text to be extracted from the text during the visual phase, allowing to identify the structure such as titles, text and tables and giving an output of these elements in a structured manner. Since PPstructure does not have native support for Swedish, some work would either have to be made here by post-processing the output through some sort of language correction or through training the model on a new Swedish dataset.

4.2 Comparison of OCR approaches

The main clear comparison is between tesseract and paddleOCR while paddleOCR contains 2 approaches, with or without PPstructure. The clear considerations here are how much work to get the approaches working, with tesseract mostly already ready to implement while the paddleOCR methods needs configuration and work to be able to be applied to Bolagsverkets dataset.

During the comparison of these approaches some preliminary testing was made on the different approaches, showing Tesseract is easiest to implement with both paddleOCR approaches being much more difficult to implement because they do not return the output as a clear text string and that PPstructure does not have native support for Swedish characters. The Pugh matrix in table 2 shows some important attributes that were considered and how they were weighed.

Table 2. Weighted Pugh matrix, total calculated by the sum of each cell multiplied with the attributes weight, for each approach.

Pugh Matrix				
Attributes	***Weight***	**Tesseract**	**PPocr**	**PPStrucuture**
ease of implementation	8	1	0	−1
accuracy	5	0	1	1
Swedish	3	1	1	−1
structured tables	7	−1	−1	1
trainable	1	−1	1	1
Total:	NA	3	2	2

4.3 Chosen computer vision approach

This project will take the approach of using Tesseract as its OCR-engine. This is because this project does not mainly focus on OCR even if it is a key pillar and requirement for the other parts. Tesseract allows for easier implementation while both approaches using PadddleOCR show a high degree of promise in preliminary testing because of their accuracy on numbers and possibility to structure data. The PaddleOCR approaches were discarded after some exploration but will be discussed as a possibility in future works in the discussion chapter.

4.4 Named entity recognition approach alternatives

This section will provide a presentation for 2 different possible approaches for extracting features in balance sheets after OCR has been performed.

4.4.1 Fuzzy string

There are many common financial posts in balance sheets as most annual reports report on the same financial posts. A couple of examples below that are very common in balance sheets:

- yearly result
- tax-debts

- customer debts

Based on this, an approach could be taken where a very large dictionary of possible financial posts/words could be collected with a relevant category/label.

If such a dictionary were made, the OCR-results could then be compared to this dictionary with percentage-based matching, sometimes referred to as fuzzy string matching. This approach is similar to the one taken by Hällgrens work mentioned in the similar works in section 2.12.1, though in that work, only highly specific posts were searched for while here it would need to be a large dictionary of posts sorted into categories/labels.

4.4.2 Language model

Language models can be tasked with an array of different tasks, one which is named entity recognition. Considering this, one approach would be to train a language model to be able to identify the specific types of financial posts present in balance sheets. This approach would require annotation of a set of annual reports for the relevant entities. The model would then be trained on this set, to also be able to identify patterns and words relating to the named entities in new data input.

4.5 Comparison of Named entity approaches

The attributes considered and how they are weighed is seen in table 3 "considers context*" refers to the possibility that the same entity when it comes to wording still might be of a different category/ label based on its context in the balance sheet. As an example, consider the word "faktura" meaning invoice in Swedish, this could be an invoice that the company owes money to a supplier or possibly an invoice to a client owing the company money, this would change the category from an asset to a debt depending on the context of where in the balance sheet it would appear.

"Possibility to classify unseen entities**" refers to words/entities never seen before in the dataset / dictionary.

The weights of attributes were given considerations based on that annual reports are not standardized in Sweden and as well as the performance of the resulting pipeline and on time restrictions in this project.

Table 3. weighted Pugh matrix, total calculated by the sum of each cell multiplied with the attributes weight. for each approach.

Pugh Matrix			
Attributes	***Weight***	**fuzzy string search**	**Language model**
time to gather data	7	0	−1
time to implement	5	1	−1
time performance (during classification)	5	1	0
considers context*	7	−1	1
possibility to classify unseen entities **	10	−1	1
Total:	NA	−7	5

4.6 Chosen Named entity approach

The project will use a language model as its approach for NER. The main reasons for this are because of the fact that the Swedish annual reports aren't standardized leading to needing a method that can identify entities based on their context as well as recognizing completely new wordings as possible entities. While implementing a language model requires more time than a fuzzy string matching against a dictionary, if focus is laid on this part it is a feasible assignment for this project but would require a large degree of this project's focus on implementing a language model for NER.

5 Implementation

This implementation chapter will be split into three main parts. The first part will present how the implementation of the parts necessary to gather and construct a dataset for the training of the language-model was done. Part two will present the implementation and training of the language model. Part three will cover the implementation of the full pipeline to extract the features of the balance sheet from start to finish. Parts of the source-code will be available through a GitHub repo linked in appendix A. The datasets from Bolagsverket as-well as the parts of code that directly interact with resources from Bolagsverket will however be kept hidden to protect their property.

5.1 Data gathering and construction of datasets.

This section will present the implementation for all parts necessary to gather and annotate datasets necessary for this project. The full overview of this process can be seen in figure 5 below.

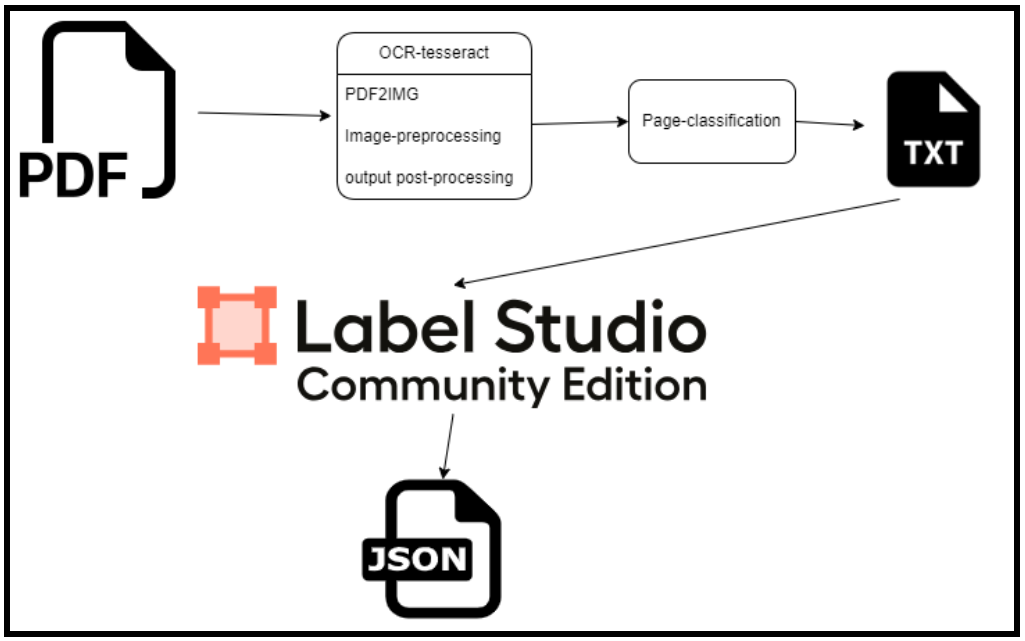


Figure 5: Process overview for constructing dataset

5.1.1 Scanned pdfs and OCR

This section will cover the first two parts of figure 5, the process of turning scanned physical annual reports into digital data. Firstly, the project was offered a large directory of annual reports in both pdf

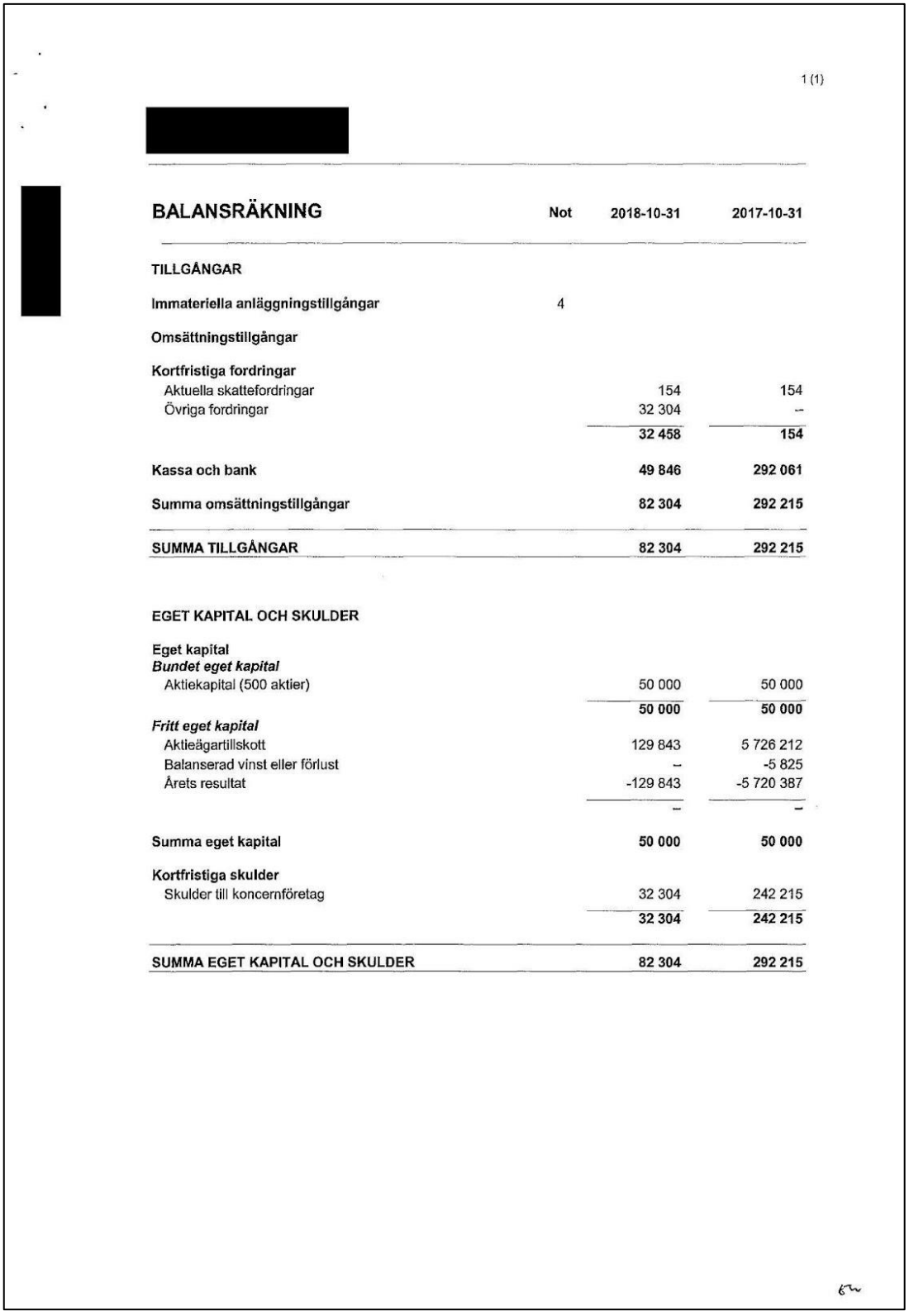
and .tiff format from Bolagsverket. From this Directory an amount of pdfs was chosen without considerations on image quality or contents in an attempt to correctly represent the typical variance in reports that Bolagsverket receives.

After the pdf's have been collected they go through pre-processing to prepare the pdf's for the OCR-engine, most of this source code is not shared since it was part of the SIMS 2023 project with Bolagsverket[19] which has closed source-code but it's functions are explained more in detail in the "Automating annual report control" report[19]. In short the preprocessing consists of using the library pdf2image[28] to convert the pdf into images and cropping the image to exclude noise such as stamps and punch holes which are common in certain positions.

After preprocessing the report, the report is now in the format of a list of images represented by ndarrays from the numpy library[29]. Every Image in this list is processed through tesseract-OCR through the pytesseract library[25] and returns each image as a string of text. Tesseract is configured to allow for Swedish characters and to keep some structure such as horizontal spacing of characters and numbers to be able to differentiate their context.

When the OCR has returned a List of images, 2 functions are run, Firstly connect_numbers() which utilizes the configuration in tesseract which kept the horizontal spacing to connect numbers such as "100 000" to "100000" since they represent one value while keeping unconnected values separated such as " 300 000 400 000" based on their distance from each other. After numbers have been connected all trailing horizontal spacing is reconnected to just one space character.

In figure 6 and 7 the balance sheet from an annual report and the text which is the post processed output is shown as an example of the processing. black bars are included to censor any identifying information in this report.



1 (1)

[redacted]

[redacted]

BALANSRÄKNING

	Not	2018-10-31	2017-10-31
TILLGÅNGAR			
Immateriella anläggningstillgångar	4		
Omsättningstillgångar			
Kortfristiga fordringar			
Aktuella skattefordringar		154	154
Övriga fordringar		32 304	–
		32 458	**154**
Kassa och bank		**49 846**	**292 061**
Summa omsättningstillgångar		**82 304**	**292 215**
SUMMA TILLGÅNGAR		**82 304**	**292 215**
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital (500 aktier)		50 000	50 000
		50 000	**50 000**
Fritt eget kapital			
Aktieägartillskott		129 843	5 726 212
Balanserad vinst eller förlust		–	-5 825
Årets resultat		-129 843	-5 720 387
		–	–
Summa eget kapital		**50 000**	**50 000**
Kortfristiga skulder			
Skulder till koncernföretag		32 304	242 215
		32 304	**242 215**
SUMMA EGET KAPITAL OCH SKULDER		**82 304**	**292 215**

Figure 6: A balance sheet from an annual report

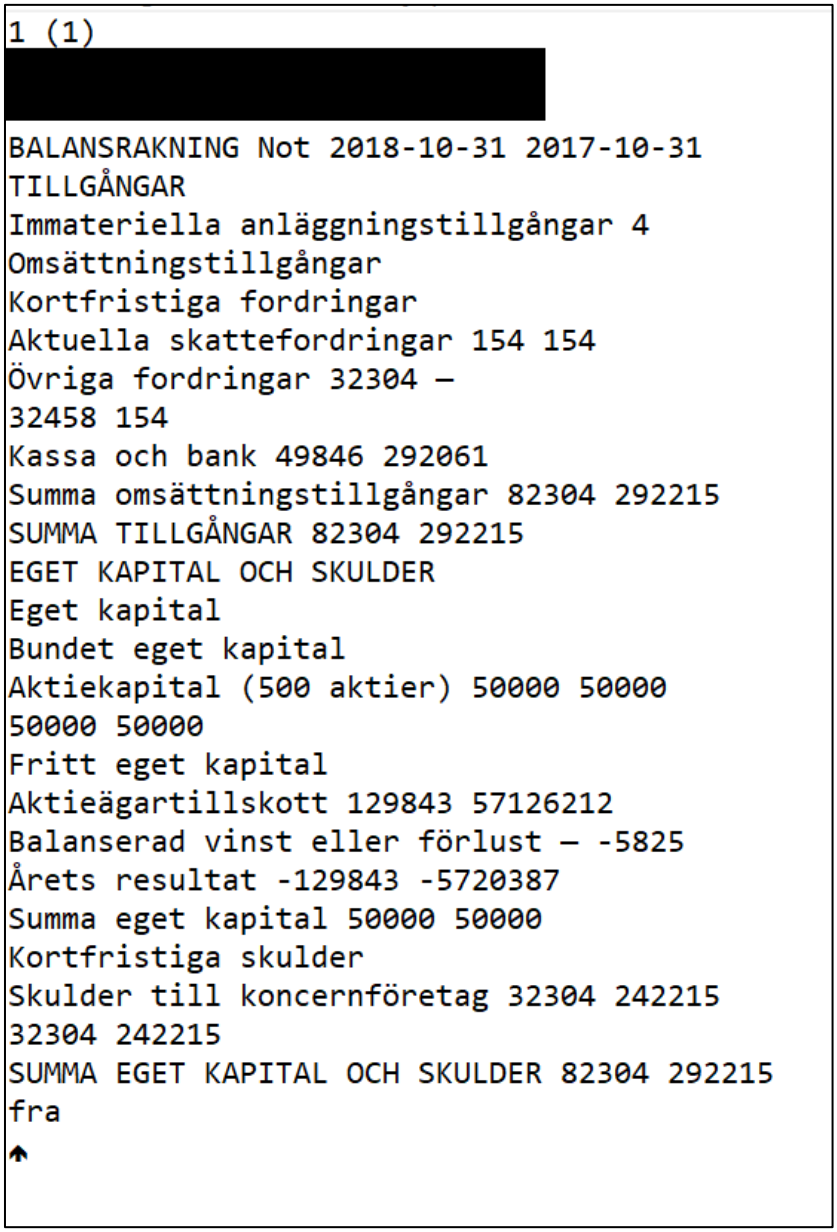


Figure 7: Output string after pre-process, OCR and post-process

5.1.2 Page Classification.

Since this project focuses on the balance sheet, the specific pages containing the balance sheets needed to be extracted. This was made using the classification model from the SIMS 2023 project" Automating annual report control" [19] which is a gradient boosting model that is able to classify 7 classes of pages from annual reports represented as strings. This model shows about 96% accuracy in predicting a string as the correct class overall, while looking at the balance sheet the model

shows 0% false positives on balance sheets [19, p. 56]. This model is used on the list of string representations of the report to classify the strings, after which the correct page's strings are saved as .txt files.

5.1.3 Annotation and Label-studio.

After a large amount of .txt files were collected it was necessary to annotate all these with the entities present in a balance sheet. There are 4 main entities that a balance sheet can consist of which are: Assets, debts, equity and deposits.[30] there is also possible other special entities which are used to exclude values from the balance such as the "obeskattade reserver" [31] or "untaxed reserves" in English.

In this implementation it was chosen to have 5 entities, where the 4 main ones are implemented as well as a fifth one referred to as miscellaneous to cover untaxed reserves and possibly some other outliers, figure 8 shows the chosen entities.

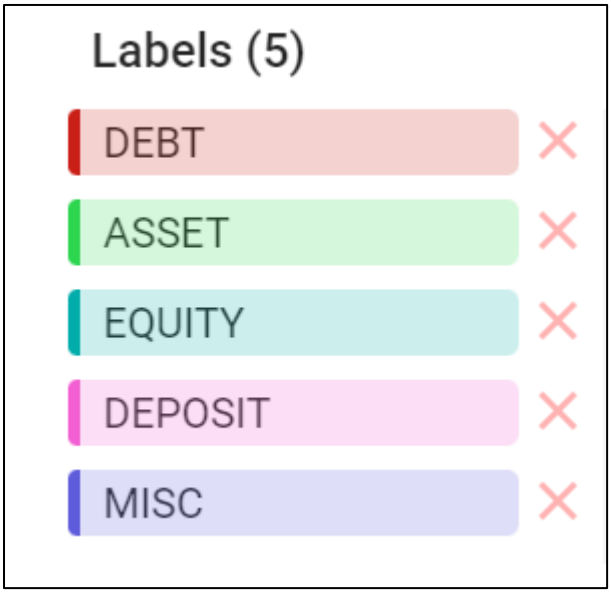


Figure 8: The 5 entities shown in Label studio label interface.

The open-source software Label studio [26] was used to manually annotate the dataset. All the txt files created from the previous section are uploaded to the label-studio server and each txt file is considered an annotation task. Label studio then allows to show the contents of the txt file with an interface to label sections included in a named entity as seen in figure 9.

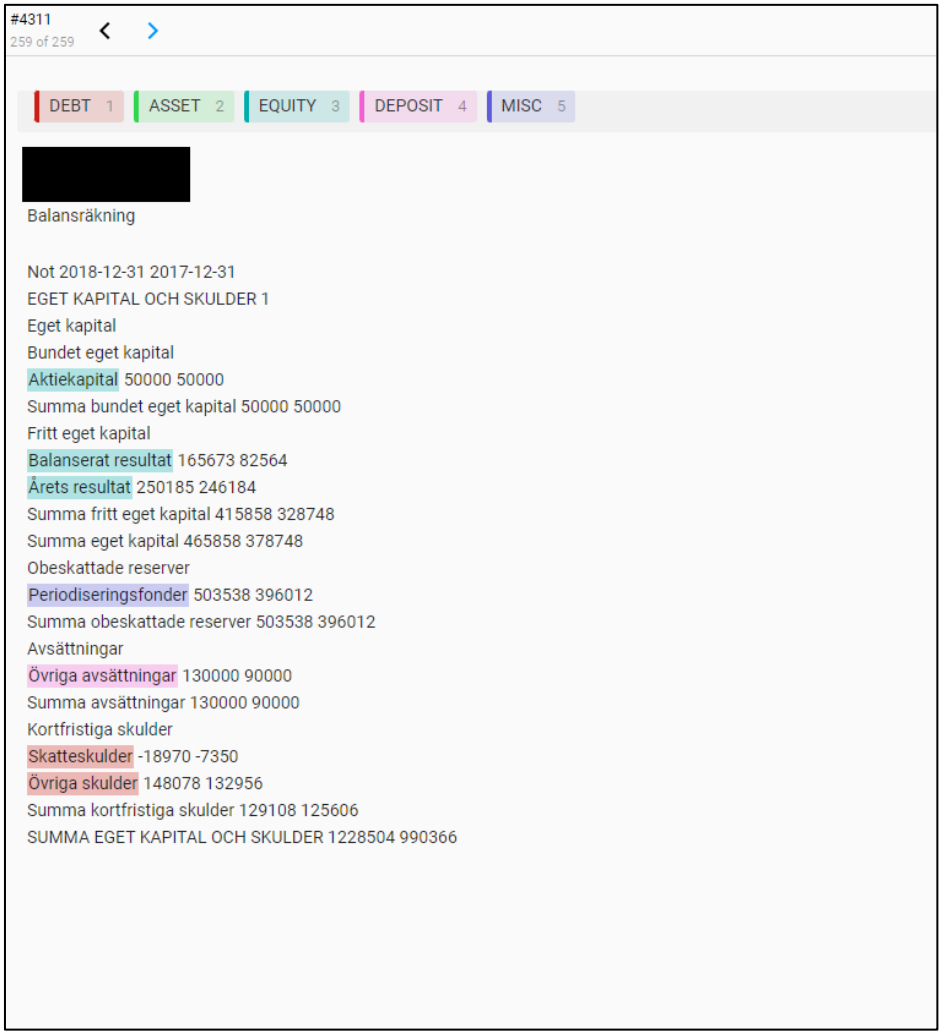


Figure 9: Annotation interface on a balance sheet as a .txt file in Label studio, black bar censoring identifying data

After all the .txt files have been annotated in label studio they are exported as a list of JSON objects in one large JSON file, each object containing a reference to the .txt file with the contents as well as a list of labels/ annotations, each of which contains a start and end index for its label. The JSON object also contains some metadata not needing to be considered here. Figure 10 shows an example of one of these JSON objects.

```
{
    "text": "Å9046767-19_190211_121915_ARSREDOVISNING.pdf0.txt",
    "id": 4053,
    "label": [
      {
        "start": 162,
        "end": 186,
        "labels": [
          "ASSET"
        ]
      },
      {
        "start": 206,
        "end": 244,
        "labels": [
          "ASSET"
        ]
      },
      {
        "start": 415,
        "end": 432,
        "labels": [
          "ASSET"
        ]
      }
    ],
    "annotator": 1,
    "annotation_id": 1585,
    "created_at": "2024-05-07T08:00:27.863760Z",
    "updated_at": "2024-05-07T08:00:27.863792Z",
    "lead_time": 35.713
  }
```


Figure 10: JSON object referencing text file with annotations.

5.2 Training the model

This section will go over the implementation of all necessary parts to finetune the BERT model to the dataset constructed during this project. Figure 11 shows a full overview of the parts for training. There will be 4 sections: Preparation of dataset, preparation of pre-trained model, optimizing training parameters and lastly the final finetuning/training of the model. This section has full source-code available through Appendix A as a Jupyter notebook named train_model_on_pages.ipynb.

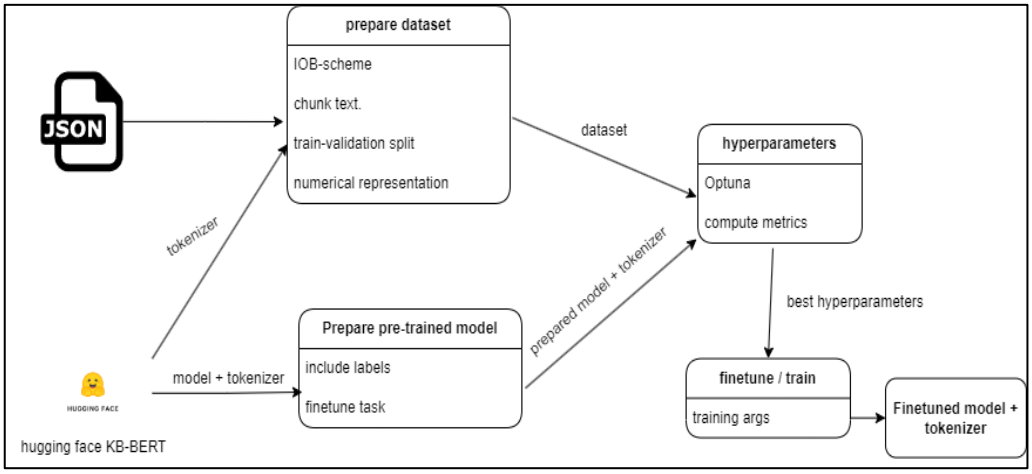


Figure 11: Overview of training process.

5.2.1 Prepare dataset

To prepare the dataset the relevant tokenizer for the later used model is downloaded through the Hugging face library Transformers and assigned to a variable.

Afterwards the relevant Labels in a balance sheet are declared and given a numerical value through 2 maps, one mapping label to number and the other being the inverse map. The labels and their mapped number are as follows:

- O = 0
- ASSET = 1
- DEBT = 2
- EQUITY= 3
- DEPOSIT = 4
- MISC = 5

note that "O" represents the lack of an entity.

The script then sends the Json-file containing the label-annotations for the dataset to the function process_json_file(). This function reads the file and extracts the list of JSON-objects and sends the data to the function label_data().

Label_data() loops through each JSON object, it extracts the file-path to the original text and all labels with their start and end index in the original text.

The text is then tokenized into a list of tokens with inclusion of "offset_mapping" to map each token to its index in the original text. A separate list of labels is made of the same length as the token list to map a label to each token, all tokens are mapped to "O" as default. Then the function has an inner loop to check each label's indexes and check which tokens are within its bound using the offest_mapping, the indexes of these tokens are used to apply its label in the label list. After all the loops are done the function returns 3 lists of lists: a list of token lists, a list of label lists and a list of attention mask (all 1's) lists.

A data-collator from hugging face Transformers is created from the tokenizer to configure the dataset for training, for example padding all the entries in the dataset to be of the same length before training. The dataset is also randomly split into 80% training data and 20% validation data.

The full code for preparing and labeling the dataset is available in the first cell of train_model_on_pages.ipynb in Appendix A.

5.2.2 preparing model for optimizing

For optimizing: a function called objective is the main function. The model is prepared within a sub-function of this function called model_init(): This function downloads the model from hugging face and configures it for token classification. The model is configured for the 6 labels and is given the mapping function between their numbers and labels. This is Available in cell 3 of train_model_on_pages.ipynb in Appendix A.

To optimize the model, it needs to be able to determine its own metrics during training and optimizing trials. This is done through a function

called compute_metrics(). Compute metrics takes a variable both containing the model's predictions and true values from the validation dataset and uses functions from sci-kit learn [32] to calculate precision, recall, F1-score and accuracy of the predictions. Special tokens such as padding are ignored here. Code available in cell 2 of train_model_on_pages.ipynb in Appendix A.

5.2.3 Optimizing for hyperparameters

Optuna [33] is the main framework used to run multiple training trials with different parameters to determine which parameters result in the best performance on a validation set. An Optuna study is created to run 50 Trials trying to maximize the F1-score of the model.

Optuna runs the function objective() 50 times which starts with declaring 6 different training parameters that Optuna assigns a value for during each trial. The training parameters which are tested on are:

- learning rate
- batch size
- training epochs
- warmup steps
- weight decay
- learning scheduler type

TrainingArguments from huggingface Transformers are given these parameters.

A Trainer object from huggingface Transformers is declared with the model, trainingArguments(hyperparameters), a training dataset, a validation dataset, the data collator and the function compute_metrics to determine performance.

The Trainer runs a training cycle and then evaluates the performance with the compute_metrics function. The function returns the F1 score to the Optuna study.

After all 50 Trials, the Optuna study object now contains a list of all trials, which hyperparameters they had and what F1-score they resulted in. Through Study.best_trial.params the parameters causing the highest F1-score can be accessed.

In this implementation of optimizing hyperparameters, manual memory management also had to be implemented because of issues with Optuna accumulating memory on the graphics processing units (GPU's) even with parameters explicitly telling it to use garbage collection in between trials.

Source code for this is available in cell 3 of train_model_on_pages.ipynb in Appendix A.

5.2.4 Training/finetuneing the model

To train/finetune the finished model, TrainingArguments are declared with the best parameters found in the optimizing trials. These are then sent to a Trainer object alongside the model "KB/bert-base-swedish-cased-ner", the datasets for training and validation, the data collator, the tokenizer and the compute_metrics function. The trainer then runs a training cycle after which the model and its tokenizer are saved in a directory "model/", this being the finished finetuned model for this project.

Source-code for this section is available through cell 4 and 5 in train_model_on_pages.ipynb in Appendix A.

5.3 Full feature extraction pipeline

This section covers the implementation of the full pipeline which should be able to take a scanned annual report in pdf format and extract features from it and save these as a comma separated values (csv) file. The full process is visualized in figure 12. The python code for the full pipeline is available through the file full_pipeline.py in Appendix A.

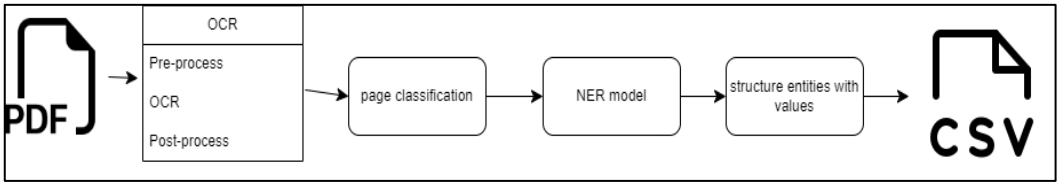


Figure 12: Process overview of full pipeline for extracting features from balance sheet.

The sections of pdf, OCR and page classification in figure 12 is identical to the one covered in the dataset construction, see section **5.1.1** and **5.1.2.**

After the visual components and extraction of the balance sheet the next part of the process uses the finetuned NER-model to do named entity recognition on the balance sheets.

The function apply_ner() present in full_pipeline.py in Appendix A does this:

The text from the balance sheets is tokenized with the tokenizer saved with the model, the output from the tokenizer is both the input_ids and the offset_mapping to keep track of their original positions in the text.

The NER-model then runs inference on the balance sheet and the predicted labels are gathered.

The function then returns the tokens, the labels, and the offset_mapping.

The last part of the pipeline will use the results from the NER-model predictions alongside some common structures in the balance sheet that helps us extract features.

The functions extract_features() once it has collected the predictions from NER, uses the function find_line_indices() to find the indexed bounds of each line in the original text.

With the help of line_indices and offset_mapping, the function align_tokens_to_lines() splits all tokens and labels into multiple lists, all representing a line in the original text. With each line being treated as a possible feature, the aligned tokens and labels are then sent to the function convert_to_features().

Convert_to_features() takes all lines that contain some label that isn't "O"(lack of labels) and finds the most common one. It then extracts that full entity in the text and its label.

Since the chosen entities have numbers associated with them in a balance sheet, a function: extract-trailing_numbers() is used to extract the numbers associated with the entity on that line based on how balance sheets are typically structured.

After entities, their labels and their numbers have been collected they are formatted as CSV rows and a full CSV file is constructed to represent the features in the annual report.

5.4 Measurement setup

The following section will discuss how testing for collecting metrics was set up.

5.4.1 OCR

The OCR will not be directly tested since there has been no focus on improving the default results from a premade OCR-engine. Errors in the OCR will however lead to downstream errors, so when analyzing results on down-stream measurements, OCR-results will be discussed.

5.4.2 NER-model

The NER-model will be tested on a smaller annotated test-set to create a number of measurements.

- accuracy
- precision
- recall
- F1-score
- confusion matrices

These measurements are done on an annotated test-set of annual reports through calculations made with the sci-kit learn metrics library [32].

5.4.3 Full pipeline

From a test-set, the full correct output csv is manually constructed and compared to the output of the when the reports are processed through the pipeline.

- amount of features
- accuracy of features (entity corresponds to original, spelling errors allowed)
- accuracy of numerical values

The full pipeline will be used on the 20 reports in the test set, and then their output .csv files will be compared to actual values in manually created correct csv files.

6 Results

This chapter will present the results of constructed resources and parts of the pipeline and the metrics gathered while performing evaluation and testing on them. The first section will present the gathered datasets and the distributions of labels in these. The second part will present the results of the NER-model and its performance metrics. Lastly the results and metrics of the full pipeline will be presented as well as some examples of publicly available reports processed through this pipeline.

6.1 Resulting datasets

The following sections will present the 3 datasets constructed during this project and their contents.

6.1.1 Main dataset

The main dataset consists of 230 different reports and a total of 398 pages of balance sheets. In table 4 the total occurrences of the 5 labels and unlabeled training data included after annotation is shown.

Table 4. Amount of each label, both full and token count.

LABEL:	AMOUNT FULL ENTITIES:	TOKEN AMOUNT:
O (UNLABELED)	NA	57958
ASSET	912	4166
EQUITY	646	1737
DEBT	718	2746
MISC	76	418
DEPOSIT	3	64

The main dataset is randomly split into 80% training and 20% for validation during the training phase.

6.1.2 NER test dataset

The NER test-set consists of 20 reports and 30 pages of balance sheets, entity distribution shown below in table 5.

Table 5. Test-dataset for NER, amount of full entities and tokens for each label.

LABEL:	AMOUNT FULL ENTITIES:	TOKEN AMOUNT:
O (UNLABELED)	NA	4332
ASSET	60	247
EQUITY	55	176
DEBT	49	204
MISC	6	27
DEPOSIT	0	0

6.1.3 Full pipeline test dataset

This test set consists of 20 .csv files manually created to represent the correct feature extraction of balance sheets; these will be used to compare to the full pipeline output. The original files/annual reports for this dataset overlaps with the NER test dataset but are not identical. Label distribution displayed in table 6.

Table 6. Amount of labels in .csv files for correct features from annual reports.

LABEL:	AMOUNT FULL FEATURES:
ASSET	63
EQUITY	63
DEBT	50
MISC	4

6.2 Named entity recognition (Finetuned KB BERT)

This following section will present the results on the finetuned NER-model trained during this project, it will present the base model, the parameters used in training and lastly the evaluation metrics will be presented and the F1-score shown in comparison with some other models in the field.

6.2.1 Base-model

The base model with pretraining on the Swedish language used is from the National Library of Sweden, with their model named "bert-base-swedish-cased-ner." This model is available through the Hugging Face Hub [34]. The original classification head of this model is discarded, as it contains classifications for a different set of named entities. The configuration and metadata about the model are available in the config.json file in Appendix A.

6.2.2 Training parameters

The training parameters found as optimal on the validation set by Optuna can be seen in table 7.

Table 7. Training parameters for training transformer model.

PARAMETER:	VALUE:
LEARNING_RATE	8.118900895030543e-05
PER_DEVICE_TRAIN_BATCH_SIZE	8
NUM_TRAIN_EPOCHS	9
WARMUP_STEPS	361
WEIGHT_DECAY	0.21913316520609985
LR_SCHEDULER_TYPE	'cosine'

6.2.3 NER metrics

The metrics during each training epoch with the Optuna parameters from previous table 7 is presented in figure 13.

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	1.152400	0.821278	0.142376	0.166625	0.153549	0.854070
2	0.460100	0.397077	0.142381	0.166667	0.153569	0.854285
3	0.178400	0.148824	0.604334	0.594939	0.595166	0.967013
4	0.036600	0.052035	0.812236	0.810214	0.811118	0.990391
5	0.015400	0.043819	0.813548	0.815381	0.814189	0.992184
6	0.010000	0.041705	0.799286	0.817209	0.807378	0.992112
7	0.009100	0.033925	0.984479	0.882406	0.910078	0.993905
8	0.006800	0.030007	0.994060	0.986075	0.989947	0.995339
9	0.008600	0.036299	0.993393	0.982475	0.987715	0.994478

Figure 13: Picture of training output during each training epoch

Table 8 below shows the finished metrics on the test dataset using the finetuned BERT model.

Table 8. Metrics for NER-model on test dataset

Precision	Recall	F1-Score	Accuracy
0.975	**0.943**	**0.955**	**0.993**

6.2.4 F1 benchmark comparisons

There are currently no perfect benchmarks or testing sets for the entities chosen during this project. Benchmarks will just be used as existing benchmarks in the finance field though these do not mirror the exact entities from this project either but there are some examples of similar entities. The base model KB-BERT's benchmarking is also included but this is not on financial data and does not specify recall and precision, only F1-score. Comparisons are seen in table 9.

Table 9. NER metrics comparisons, KB-BERT's recall and precision is not available.

	THIS PROJECT	FINANCIAL NER [21]	SEC-BERT [22]	KB-BERT [14]
F1-SCORE	**0.955**	**0.92**	**0.82**	**0.93**
PRECISION	**0.975**	**0.93**	**0.81**	**NA**
RECALL	**0.943**	**0.91**	**0.83**	**NA**

6.2.5 Confusion matrix

Figure 14 shows the confusion matrix on classifications of tokens in the NER test-set with comparing predicted and true values, the confusion matrix was normalized to proportions to create a clearer visualization, otherwise the amount of labels are unbalanced with unlabeled data(O) completely overwhelming the matrix.

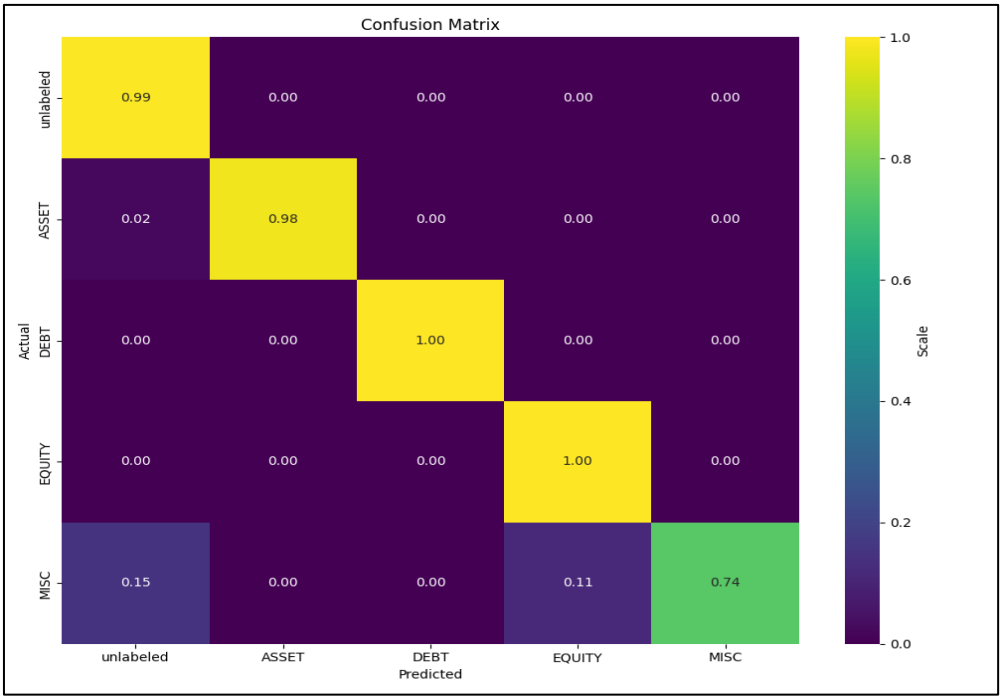


Figure 14: Confusion matrix on token classification by the NER-model

6.3 Full pipeline

This section will present the metrics after performing testing on the full pipeline for feature extraction. The metrics will be split into metrics on finding the features, metrics on number extraction and distribution of errors in number extraction.

6.3.1 Evaluation metrics

In the following table 10: metrics on entities are shown, these are not token based, if a line of text has the majority of tokens labeled as the correct token (excluding unlabeled tokens) it gets extracted as a feature and this would be considered as the correct feature extracted if the label matches the true value.

Table 10. Metrics on correct entries/lines considered a feature from pipeline output.

Accuracy	Precision	Recall	F1-score	Missed features / total	False positives / total lines
99,5%	**0.984**	**0.994**	**0.989**	**1 / 180**	**3 / 693**

When it comes to the numbers associated with feature there can be up to two periods in a balance sheet so each feature can contain up to two related values, but it's also possible for there to only be one period and a period can contain a null or zero value for some feature.

The metrics on numbers are calculated through checking each number to see if it matches the true number. dashes, zeroes and empty values are considered as identical all referencing a zero/null value. The metrics are shown below in table 11.

Table 11. Metrics on extraction of correct numbers.

total number amount	correct numbers	accuracy %
289	**261**	**90%**

The errors on numbers have been sorted into two types of errors, errors caused by failures in the OCR part of the pipeline and for errors caused

by a variance in structure or presentation in annual reports causing the python script to fail in extracting the numbers correctly. Error distribution displayed below in table 12.

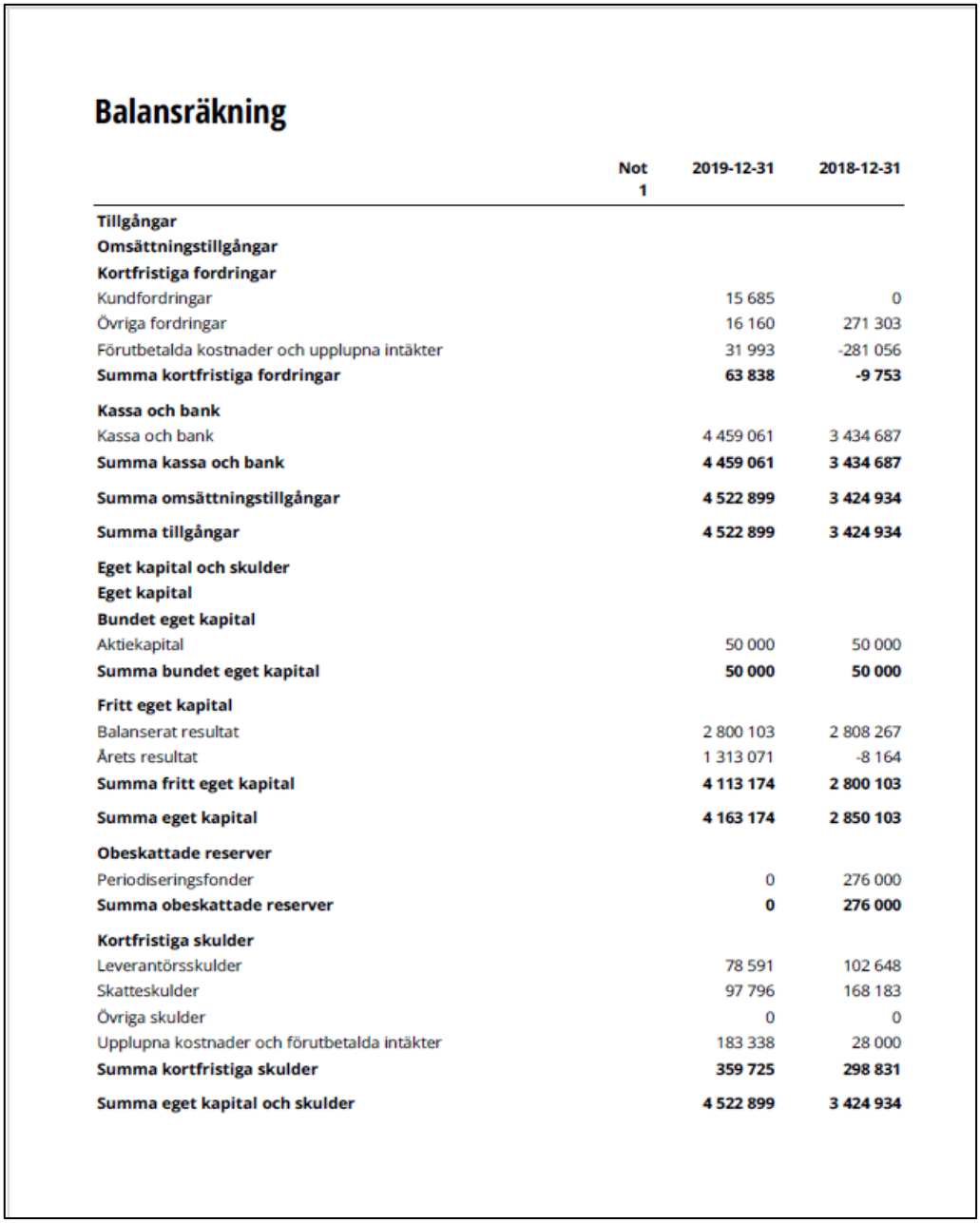
Table 12. Distribution of error types when extracting faulty numbers

total errors	OCR errors	Script errors
28	**18**	**10**

6.3.2 Examples

This section will show a picture of a balance sheet and the resulting .csv file after the original pdf from which the balance sheet comes from has been processed through the pipeline. First an example of a perfect extraction will be shown, then an extraction with errors.

A balance sheet is presented in figure 15, and after the report with this balance sheet has been processed its result is displayed in table 13. This first example is of a perfect feature extraction.



Balansräkning

	Not 1	2019-12-31	2018-12-31
Tillgångar			
Omsättningstillgångar			
Kortfristiga fordringar			
Kundfordringar		15 685	0
Övriga fordringar		16 160	271 303
Förutbetalda kostnader och upplupna intäkter		31 993	-281 056
Summa kortfristiga fordringar		**63 838**	**-9 753**
Kassa och bank			
Kassa och bank		4 459 061	3 434 687
Summa kassa och bank		**4 459 061**	**3 434 687**
Summa omsättningstillgångar		**4 522 899**	**3 424 934**
Summa tillgångar		**4 522 899**	**3 424 934**
Eget kapital och skulder			
Eget kapital			
Bundet eget kapital			
Aktiekapital		50 000	50 000
Summa bundet eget kapital		**50 000**	**50 000**
Fritt eget kapital			
Balanserat resultat		2 800 103	2 808 267
Årets resultat		1 313 071	-8 164
Summa fritt eget kapital		**4 113 174**	**2 800 103**
Summa eget kapital		**4 163 174**	**2 850 103**
Obeskattade reserver			
Periodiseringsfonder		0	276 000
Summa obeskattade reserver		**0**	**276 000**
Kortfristiga skulder			
Leverantörsskulder		78 591	102 648
Skatteskulder		97 796	168 183
Övriga skulder		0	0
Upplupna kostnader och förutbetalda intäkter		183 338	28 000
Summa kortfristiga skulder		**359 725**	**298 831**
Summa eget kapital och skulder		**4 522 899**	**3 424 934**

Figure 15: Picture of balance sheet.

Table 13. Table from .csv, perfect output from pipeline on annual report containing balance sheet seen in figure 15.

LABEL	ENTITY	NUMBER 1	NUMBER 2
ASSET	Kundfordringar	15685	0
ASSET	Övriga fordringar	16160	271303
ASSET	Förutbetalda kostnader och upplupna intäkter	31993	−281056
ASSET	Kassa och bank	4459061	3434687
EQUITY	Aktiekapital	50000	50000
EQUITY	Balanserat resultat	2800103	2808267
EQUITY	Årets resultat	1313071	−8164
MISC	Periodiseringsfonder	0	276000
DEBT	Leverantörsskulder	78591	102648
DEBT	Skatteskulder	97796	168183
DEBT	Övriga skulder	0	0
DEBT	Upplupna kostnader och förutbetalda intäkter	183338	28000

For the next example a flawed feature extraction is displayed, the balance sheets from the annual report are shown in figure 16 and the results are shown in table 14 with errors colored in red and “semi-errors” in orange such as misspellings because of the OCR.

Balansräkning

Belopp i kr	*Not*	*2019-12-31*	*2018-12-31*
TILLGÅNGAR			
Anläggningstillgångar			
Materiella anläggningstillgångar			
Inventarier, verktyg och installationer	2	7 344	12 240
Summa materiella anläggningstillgångar		7 344	12 240
Finansiella anläggningstillgångar			
Andelar i andra långfristiga värdepappersinnehav	3	1 950 400	1 950 400
Summa finansiella anläggningstillgångar		1 950 400	1 950 400
Summa anläggningstillgångar		1 957 744	1 962 640
Omsättningstillgångar			
Kortfristiga fordringar			
Övriga fordringar		6 021	4 596
Förutbetalda kostnader och upplupna intäkter		-	-
Summa kortfristiga fordringar		6 021	4 596
Kassa och bank			
Kassa och bank		144	1 764
Summa kassa och bank		144	1 764
Summa omsättningstillgångar		6 165	6 360
SUMMA TILLGÅNGAR		1 963 909	1 969 000

Balansräkning

Belopp i kr	*Not*	*2019-12-31*	*2018-12-31*
EGET KAPITAL OCH SKULDER			
Eget kapital			
Bundet eget kapital			
Aktiekapital (500 aktier)		50 000	50 000
Summa bundet eget kapital		50 000	50 000
Fritt eget kapital			
Balanserat resultat		1 586 035	1 610 708
Årets resultat		-12 216	-24 673
Summa fritt eget kapital		1 573 819	1 586 035
Summa eget kapital		1 623 819	1 636 035
Kortfristiga skulder			
Övriga skulder		340 090	332 965
Summa kortfristiga skulder		340 090	332 965
SUMMA EGET KAPITAL OCH SKULDER		1 963 909	1 969 000

Figure 16: 2 pages of balance sheets from annual report

Table 14. Flawed feature extraction from pipeline on annual report with balance sheets seen in figure 16, red marks wrong numbers, orange marks correct entity but misspellings or partial token misclassifications.

LABEL	ENTITY	NUMBER 1	NUMBER 2
ASSET	Inventarier , verktyg och installationer	7344	12240
ASSET	Andelar i andra långfristiga värdepappersinnetrav	−400	1950400
ASSET	Övriga fordringar	65021	4596
ASSET	Förutbetalda kostnader och upplupna intäkter		
ASSET	Kassa och bank	144	1764
EQUITY	Aktiekapital (500 aktier) 50 -	0	50000
EQUITY	Balanserat resultat	−35	1610708
EQUITY	Årets resultat	216	−24673
DEBT	Övriga skulder	−90	332965

7 Discussion

The Discussion chapter will first discuss the constructed parts in this project, analyzing their metrics and results. After the discussion on the constructed parts, the chapter will discuss the methodology, scientific value of the work as well as give a recommendation on the future of what's been presented in this work to Bolagsverket, which this project was done in collaboration with.

7.1 Analysis and discussion of results

The discussion of the results is split into 4 separate sections: OCR, datasets, NER-model, and the resulting full pipeline.

7.1.1 OCR

While this project did not focus on the visual aspects of feature extraction in scanned annual reports, it is important to discuss since it will affect all downstream work on the data. All data for the datasets was collected through OCR and the quality of the data in the full pipeline is highly dependent on correct data being processed.

Looking at table 12 in the results section 6.3.1, it shows that about 65% of errors in the extraction of financial numbers are because of a failure the OCR-engine caused.

With future work and improvements on the OCR, I believe this can cause significant improvements on feature extraction. It is worth noting that it may be close to impossible to remove all OCR errors as there are limitations with working on scanned annual reports as there may be some of unsalvageable quality. Therefore, some errors in the OCR are to be expected but I do believe significant improvement can be done here.

In the early stages of this project some work was committed to exploring possibilities of using other OCR engines that showed higher accuracies as mentioned in the approach chapter, though these had to be abandoned due to time limitations and a larger focus being applied to the language model. This preliminary work did show a high degree of promise and will be further mentioned in some possible future works in the conclusion chapter.

7.1.2 Datasets

The created and annotated datasets seem to be effective based on the results of the NER-model trained and tested on them but there are a couple of caveats that should be considered in if the results are generalizable:

- **Representative data**

  Bolagsverket has millions of annual reports in their possession, it easy to imagine that a training set of 230 reports and testing set of 20 reports do not cover all the possible variance and outliers that could exist in the broader dataset. The annual reports were picked somewhat randomly to try to combat any biases in the datasets but can probably not be considered a fully representative dataset.

- **Time-period**

  All the data used in the dataset are from reports covering periods in between 2017-2019. This means it should be considered that testing metrics could differ on annual reports from other periods. For example, there can be changes in software companies use to write their annual reports resulting in differing structures. Older reports might have used older scanning machines resulting in more noise in the images.

- **Deposit**

  Worth mentioning here is also the lack of Deposit entities in the test-dataset, this is because this category of financial posts seems to be a rare occurrence in the scope of K2 annual reports. Only 3 of them exist in all the datasets combined and none in the test-dataset, so any results in this report cannot be considered to have included this label even if it is a possible label in balance sheets.

If future testing and or improvements should be done in this area, I believe that creating a larger dataset to try to be representative of the overall data would benefit analysis of results.

7.1.3 NER-model

The NER model shows very promising results with very high metrics as seen in the result section with the limited dataset it was finetuned on. In

this section I will discuss these performance metrics and analyze the results.

Let's start by discussing the F1-score as this is the classic metric used for determining performance in classifications tasks. The F1-score of 0.955 is a very good result, but as mentioned previously, it should be recognized that the testing dataset of 20 annual reports is limited and might not be fully representative. However, I do believe that these results do show a high degree of promise based on the fact that balance sheets do have similar structures and contents in most cases. Therefore, a NER-model trained on a limited dataset could be an efficient solution for feature extraction of financial posts.

In the result section 6.2.4 in table 9 some comparisons to other NER-models in the finance area are shown and the original KB-BERT model which this model is a finetuned version of. The NER-model created in this project significantly outperforms the other models, but this should not be considered as a full fair comparison based on three main reasons.

- **The entities**

  These models do not all use the same set of labels, meaning they are not all trying to identify the same entities, this should be considered since some entities might be more difficult to classify.

- **The test set**

  Some of the models in the comparison have their F1-score decided based on large popular testing datasets for benchmarking. This project works on a smaller dataset constructed during this project making it difficult to compare the two in a fair manner.

- **Variance**

  As mentioned previously balance sheets have some variance since they are not standardized, but most of them do share similar structures and contents. Some of the models compared to, do testing on more varied data, for example testing on news articles, social media posts and more, allowing for more variance.

The closest comparison that could be made is with the one named "Financial NER" with an F1-score of 0.92 which is also mentioned in related works section 2.5.2. This is because like in this project its testing set is also constructed based on the use case, which in their case was financial articles. This project shares two entities/labels that reference a similar concept/category which is "F-ASSET" and "F-Liability" in their model being similar to "ASSET" and "DEBT" in mine. These are not an exact match however with theirs being broadened to a larger area where they can be applied.

Comparing the F1-score with this, mine being 0.95 and theirs being 0.92 I think this should not be considered as my model being superior, rather they might be getting close to similar performance considering that I think financial articles do have a larger degree of variance compared to balance sheets.

Moving on to the confusion matrix for the NER-model, it gives a clear indication that classifications on most labels have a very high performance, with the exception being MISC. The reason MISC performs significantly worse than the other labels only seeming to reach about 75% accuracy is most likely due to being severely underrepresented in both the training dataset and test-sets leading to the model performing worse on it. The MISC category makes up only about 5% of the 5 non-O-labels. To Improve performance on the MISC entity, the dataset should either be balanced by overrepresenting the MISC entity somewhat, or a larger dataset may help.

To summarize and generalize: I would consider the metrics and results here to be of a high degree of performance and that NER is an efficient way to extract types of financial posts and classify them correctly. when it comes to the generalizability outside of this highly specific use case: I feel somewhat confident based on the findings during this project to claim that a finetuned NER-model on a smaller dataset can be an efficient solution to extract an amount of entities from data with limited variance but still requiring a dynamic approach because of lacking strict structuring or standardizations.

7.1.4 Full pipeline

The full pipeline is the amalgamation of leveraging the OCR alongside the NER model to fully extract features and structure these into a csv of features with the added extraction of the financial numbers connected to the features.

When it comes to the labels and features, the results of this are highly dependent on the 2 earlier parts of OCR and NER which also shows in the result, as when it comes to extracting the entities and considering them features, the result metrics are close to the results from the NER-model with the NER-model having an F1-score of 0.95 and the full pipeline having 0.989 with the increase being due to the design choice of middle tokens being misclassified would still be able to be extracted based on the majority label. This part of the feature extraction can be considered equal to slightly better than the metrics of the NER-model itself.

The number extraction reached about a 90% accuracy which I would consider inadequate since if these are to be considered financial figures to possibly be analyzed, they would need to be highly accurate. 65% of these errors were because of the OCR failing and as mentioned earlier, works on improving the visual parts were not part of the scope of this project.

Instead let's focus on the script failing to extract numbers. The extraction of numbers is based on the common structure of a balance sheet which does cause some issues. The balance sheets do not have a strict structure and can in theory be structured however the company wishes to present their balance sheet.

This caused issues with the python script extracting faulty numbers. I think a solution for extracting numbers requires a different method because of this reason which will be further discussed in recommendations for Bolagsverket and in future work.

Another remark that can be done here is that 90% accuracy on numbers are similar to the found accuracies in Hällgrens work on annual reports mentioned in similar work 2.5.1 which reached in between 89.6 % and 92.9% accuracy. This could indicate a flaw in the underlying dataset leading to difficulties in reaching much higher accuracies. Both this

project and Hällgrens project also used TesseractOCR as its OCR engine so it could also indicate that this OCR-engine seems to at most reach these accuracies.

To summarize: The full pipeline can with a high efficiency extract which features are present in the balance sheets but is inadequate to also extract the financial values corresponding to these features. In its current state this full pipeline could be considered a prototype and suggestion for feature extraction on balance sheets if number accuracy could be improved.

7.2 Project method discussion

This section will discuss each phase, if they were successful and if some improvement could have been done to their methodology.

7.2.1 First phase: Theory and exploring approaches

The theory gathered in the first phase helped give the basic understanding of the theory in the areas needed to be able to construct all the parts when it came to OCR and a NER-model based on the transformer architecture.

Exploring the approaches in the approach section helped through some preliminary testing and exploration to decide the path forward for the rest of the project. In the approaches section a significant time was spent on the approach of using paddleOCR and PPstructure since their preliminary testing showed significantly improved results compared to tesseract, but this was unfortunately abandoned due to time limitations and these tools requiring significant work to implement. Because of this I think the approach section could have benefited from earlier setting up a clearer focus on the language model approach as the base of focus during this project.

7.2.2 Second phase: Construction and implementation

The construction and implementation were successful with coding and constructing all the parts needed for gathering results and analyzing them. There are certainly improvements to the source code that could be made but in general the implementation of everything included in this project was successful and is fully functional.

7.2.3 Third phase: Connecting constructed parts and gathering results

Building the full pipeline was a successful part of the project. After the NER was tested independently to gather its results for comparison with other NER benchmarks, the full pipeline was tested using similar metrics. This helped analyze whether the full feature extraction introduced more errors compared to the NER model on its own.

7.2.4 Fourth phase: Evaluation and discussion

This phase relates to this chapter which is a more qualitative view on the results of all phases of these projects which helps us analyze the results on both a specific use-case view as well on if there is some generalizable knowledge gained. To be able to analyze the results, metrics were chosen based on the standards in the field which helps determine performance and analyze generalizability.

7.2.5 Summarized

Overall, all the phases were successful, if there was to be given one improvement to the methodology and project It would've been a more specific focus/smaller scope earlier on in the project which could have helped with some time restrictions such as not spending valuable time on the visual aspects of the projects which later was discarded. If the focus was more focused there might have been some more time to analyze the issues with number extraction and possibly improve them.

7.3 Scientific discussion

The main clear scientific knowledge gained in this project can firstly be talked about in its very specific use-case which is what Bolagsverket wanted to observe. Two of the research questions were highly specific to Bolagsverkets use-case: how many features can be extracted and how accurate are they. Those questions have been able to be answered for their use-case in this project through how the methodology and construction was set up.

When it comes to more general scientific knowledge there are limitations, this project cannot answer if the methods and tools used for this use-case would be effective in general feature extraction in documents for any type of document possible. Some broad scientific knowledge may however be considered based on the general attributes found in the use-case for Bolagsverket. If instead the question is altered slightly to: Would

these tools and methodology be an effective solution for feature-extraction in documents which share these attributes?

- A set of known entities to be extracted, exists/can exist in the document.
- The document is not strictly structured and requires a dynamic extraction of features.
- Limited variance, most documents share similar structure and patterns.

I believe the results of this project suggest that for documents with these attributes, the method of using OCR and a finetuned NER-model can help perform feature extraction quite efficiently.

7.4 Recommendation

My recommendation to Bolagsverket based on the results found during this project would be to further explore the methodology used on the balance sheet during this project for other parts in annual reports as well. The pipeline shows promise and if it could be broadened to also extract other features this could help with the problem statement from Bolagsverket. Another suggestion is to look at alternative methods of number extraction as this was the part with inadequate results during this project, some suggestions on this will be given in the section further work in the conclusion chapter.

7.5 Ethical and societal discussion

Two mentions of ethical considerations will be presented in this section, one that was undertaken during the project and one for the possible future use-case.

7.5.1 Censored or not shared data

The ethical considerations that had to be taken during the project mainly involved the sharing of data. The entire project was conducted in collaboration with Bolagsverket, which provided datasets and insights into their processes. As a result, some data, such as datasets and certain surrounding code and processes, are not shared. A non-disclosure agreement was signed between the author and Bolagsverket.

Since examples from Bolagsverket's datasets were not disclosed, some examples for the thesis and subsequent presentation were collected from publicly available sources. These contain some identifying information about the companies and data on persons connected to the companies. Although these are technically covered by the principle of public access (offentlighetsprincipen), it was decided to censor any identifying data from this report to avoid sharing details about individuals or companies that could affect them in any way.

7.5.2 AI bias

While the results and pipeline built during this project should not directly affect ethical considerations or societal impact, the purpose of this tool should be considered. How could these results potentially be used in the future, which could have larger impacts. In their problem statement, Bolagsverket mentions that this project was meant to help in processing and analyzing data in annual reports. Two of their examples in data analysis were related to crime prevention and the identification of unusual behavior.

When it comes to using AI for both feature extraction and possibly for making predictions of crime, identifying unusual behaviors, and profiling companies, one should consider the possibility of AI bias. There have been examples of AI making predictions based on less relevant features because of the introduction of human biases into the datasets on which AI models are trained. Because of this, careful consideration of what data such an AI model gets access to should be made. Which features are actually relevant to the task at hand?

Careful monitoring of predictions should also be conducted to ensure an AI model does not start making predictions based on non-relevant features that discriminate against some demographic.

8 Conclusions

The overall problem I aimed to address was to find a way to extract more data from scanned physical unstructured documents, specifically annual reports but with the question if the same methods could be used for other types of documents. I believe the answer is positive, through the results and pipeline presented in this project it has shown a way to extract features in a part of the annual reports being the balance sheet. The solution can extract all financial posts in a balance sheet in most cases with further work being needed on the financial numbers (90% accuracy). The results also suggest that the same methods should be able to be applied to other areas and documents if they share similar attributes such as limited variance, known entities and a need for a dynamic extraction but further testing should be done on other types of documents for a more definitive answer.

For the specific research questions, they will be answered here with specifics for this use case on the first two and generalized knowledge on the third question:

- **How many data points can be extracted using the constructed pipeline?**

  The scope of this project was on the balance sheet where 99%+ of features could be extracted.

- **How accurate are the numbers extracted?**

  The numbers relating to a feature were attempted to be extracted and an accuracy of 90% was found in testing.

- **Are the methods of the constructed pipeline an efficient solution for feature extraction in general unstructured documents?**

  The results from this project can't answer this question for every single type of unstructured document, it does however suggest the methodology for feature extraction in the pipeline could be effective in documents with known entities, a limited variance in structure and a need for dynamic extraction.

8.1 Future Work

In this section some ideas of future work will be presented, these could be to improve the results found here, to further increase the scope on the same area as this project or some other future work discovered during this project.

8.1.1 Number extraction for given entity/feature

One of the limitations in this project was the extraction of correct numbers belonging to the relevant entities, this is partly because the current approach tried to build a python script to do this which couldn't properly handle the amount of possible variants and structures of a balance sheet. Instead of trying to improve the python script there are two approaches that could possibly be explored to handle this in a more dynamic way.

- **Approach 1, Relation Extraction (RE):**

Currently the language model used in this project was a BERT-model for the task "named entity recognition". There are language models that can also handle the task "Relation Extraction" which is a task meant to also classify a relation between extracted entities, if the dataset was also annotated with numbers and their relation to the entity this could lead to a solution where the language model can both extract the final post and its relevant figures. Another interesting relation to explore is to also relate the financial number to a TIME entity in the cases where there are multiple time periods presented in a balance sheet.

- **Approach 2, Layout analysis and table extraction:**

Another approach is to train some computer vision models to structure the data in a balance sheet to easily extract the features and numbers as a clearly structured table. This possibility was discovered during the approach exploration in this project. PaddleOCR and PPstructure showed preliminary promise to accurately extract a table with rows and columns. The PaddleOCR framework also allows for training and fine tuning the models on specific datasets which gives the opportunity to improve its results on annual reports.

8.1.2 Further entity extraction on annual reports

This project focused on balance sheets, but there are multiple other common pages in annual reports, for example the result sheet which has a very similar structure to the balance sheet but with other entities which could easily be extracted in the same way as done in this project.

The approaches could differ here depending on what the desired result is, should feature extraction be to support possible crime prevention, if so discussion with Bolagsverket could be initiated to analyze which features could be relevant. Another approach could be wanting to extract features from annual reports for financial analysis to inform investment decisions, if so what features are relevant for such fields would have to be decided.

8.1.3 Improving OCR results

Since a majority of the errors on the numbers attempted to be extracted during this project were caused by OCR error, there is room for improving OCR results. There have been previous attempts on improving image quality on annual reports at Bolagsverket through denoising.

I see 2 approaches available here, one direct and one using post-processing based on a previous work mentioned.

- **Approach 1, direct:**

This project currently uses tesseractOCR as its engine, but some preliminary work showed PaddleOCR being more accurate, so one approach could be to implement PaddleOCR to provide a structured output similar to that of tesseract.

Another direct approach could be to create a dynamic choice of denoising algorithms and parameters. A machine learning model could be trained on a dataset of images where the image gets classified with the denoising algorithms and parameters resulting in the best OCR-output.

- **Approach 2, post processing**

In the previous work section, a work using a transformer model for OCR post-processing is mentioned. This work uses a pre-trained

transformer model as a base to be applied to translating OCR-output into a corrected version in Swedish texts, a similar model could possibly be trained on annual reports to correct OCR errors, however how functional this would be on numbers would have to be explored.

References

[1] Lu Y. Artificial intelligence: a survey on evolution, models, applications and future trends. Journal of Management Analytics [Internet]. 2019.Available from: http://dx.doi.org/10.1080/23270012.2019.1570365

[2] Baviskar D, Ahirrao S, Potdar V, Kotecha K. Efficient automated processing of the unstructured documents using artificial intelligence: A systematic literature review and future directions. IEEE Access [Internet]. 2021 Available from: http://dx.doi.org/10.1109/access.2021.3072900

[3] PaddleOCR, [Internet]. github. [Accessed 2024 april 29]. Available from: https://github.com/PaddlePaddle/PaddleOCR

[4] tesseract-ocr [Internet]. github. [Accessed 2024 april 29]. Available from: https://github.com/tesseract-ocr/tesseract

[5] Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, et al. A survey of large language models [Internet]. arXiv 2023. Available from: http://arxiv.org/abs/2303.18223

[6] Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Stroudsburg, PA, USA: Association for Computational Linguistics; 2020. Available from: https://aclanthology.org/2020.emnlp-demos.6.pdf

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In “Advances in Neural Information Processing Systems” 2017., pages 6000–6010. Available from: http://arxiv.org/abs/1706.03762

[8] Kamath U, Graham K, Emara W. Transformers for machine learning: A deep dive. Philadelphia, PA: Chapman & Hall/CRC; 2022.

[9] Mielke SJ, Alyafeai Z, Salesky E, Raffel C, Dey M, Gallé M, et al. Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP [Internet]. 2021. Available from: http://arxiv.org/abs/2112.10508

[10] Summary of the tokenizers [Internet]. Huggingface.co. [Accessed 2024 May 18]. Available from: https://huggingface.co/docs/transformers/tokenizer_summary

[11] Keraghel I, Morbieu S, Nadif M. A survey on recent advances in named entity recognition [Internet]. arXiv 2024. Available from: http://arxiv.org/abs/2401.10825

[12] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding [Internet]. arXiv 2018. Available from: http://arxiv.org/abs/1810.04805

[13] Rothman D. Transformers for Natural Language Processing. 2nd ed. Birmingham, England: Packt Publishing; 2022.

[14] Malmsten M, Börjeson L, Haffenden C. Playing with words at the National Library of Sweden -- making a Swedish BERT [Internet]. arXiv 2020. Available from: http://arxiv.org/abs/2007.01658

[15] Transformers [Internet]. Huggingface.co. [accessed 2024 april 29]. Available from: https://huggingface.co/docs/transformers/

[16] Goodfellow I, Bengio Y, Courville A. Deep Learning. London, England: MIT Press; 2016.

[17] Sklearn.Metrics.F1_score [Internet]. scikit-learn. [accessed 2024 May 1]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[18] Murel J, Kavlakoglu E. What is a confusion matrix? [Internet]. Ibm.com. 2024 [Accessed 2024 May 1]. Available from: https://www.ibm.com/topics/confusion-matrix

[19] Berglin C, Larsson H, Nilsson J, Ellström J, Persson L, Åsberg O. Automating annual report control, SIMS 2023 Bolagsverket. 2023.

[20] Nordin Hällgren S. Reading Key Figures from Annual Reports Chalmers.se. 2021. Available from: https://odr.chalmers.se/items/fb1d1535-4a47-4dc7-a16a-3e349e2ab38c

[21] Jayakumar H, Krishnakumar M.S, Peddagopu V.V.V, Sridhar R. RNN based question answer generation and ranking for financial documents using financial NER. Sādhanā 45[Internet]. 2020 Available from: http://dx.doi.org/10.1007/s12046-020-01501-3

[22] Loukas L, Fergadiotis M, Chalkidis I, Spyropoulou E, Malakasiotis P, Androutsopoulos I, et al. FiNER: Financial numeric entity recognition for XBRL tagging [Internet]. 2022 . Available from: http://arxiv.org/abs/2203.06482

[23] Xia B "namir" Rawte VD, Zaki MJ, Gupta A. FETILDA: An effective framework for fin-tuned embeddings for long financial text documents [Internet]. 2022 . Available from: http://arxiv.org/abs/2206.06952

[24] Löfgren V, Dannélls D. Post-OCR correction of digitized Swedish newspapers with ByT5. Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024). St. Julians, Malta: Association for Computational Linguistics; 2024. p. 237–42. Available from: https://aclanthology.org/2024.latechclfl-1.23/

[25] Pytesseract [Internet]. PyPI. [cited 2024 May 2]. Available from: https://pypi.org/project/pytesseract/

[26] Label Studio [Internet]. Label Studio. [Accessed 2024 May 2]. Available from: https://labelstud.io/

[27] Datasets [Internet]. Huggingface.co. [Accessed 2024 May 2]. Available from: https://huggingface.co/docs/datasets/

[28] pdf2image [Internet] github.com [Accessed 2024 May 3] Available from: https://github.com/Belval/pdf2image

[29] numpy.ndarray [Internet] numpy.org [Accessed 2024 May 3] Available from: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html

[30] Balansräkning i årsredovisningen [Internet] Bolagverket.se [Accessed 2024 May 3] Available from: https://bolagsverket.se/foretag/aktiebolag/arsredovisningforaktiebolag/

[31] Årsredovisning i mindre företag (K2) [Internet] Bokföringsnämnden. Available from: https://www.bfn.se/wp-content/uploads/vl16-10-k2ar-kons.pdf

[32] Metrics and scoring: quantifying the quality of predictions[Internet] scikit-learn.org . Available from: https://scikit-learn.org/stable/modules/model_evaluation.html

[33] Optuna: A hyperparameter optimization framework [Internet] optuna.readthedocs.io . Available from: https://optuna.readthedocs.io/en/stable/index.html

[34] bert-base-swedish-cased-ner [Internet] Huggingface.co. Available from: https://huggingface.co/KBLab/bert-base-swedish-cased-ner

Appendix A: Source Code

All source code that can be publicly shared can be accessed through the following link to a GitHub repository:

https://github.com/Jesper-Nilsson/feature-extraction-balance-sheghvfets