<Corinna Grabner> <BSc>

<Training of the Language Model> <PikabERT>

MASTERARBEIT

zur Erlangung des akademischen Grades
Diplom-Ingenieurin

STUDIUM
Informatics

Alpen-Adria-Universität Klagenfurt
Faculty of Technical Science

BEGUTACHTER
<Assoc. Prof. Dipl-Ing. Dr. Konstantin Schekotihin>

Institut für Angewandte Informatik

Version <0.01> Klagenfurt am Wörthersee, September 2, 2022

<Vor der Abgabe
mit Version der AAU abgleichen oder besser
Version der AAU einbetten!>

Zitat, falls gewünscht
von wem?

Affidavit

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated;
- I have fully disclosed all assistance received from third parties during the process of writing the paper, including any significant advice from supervisors;
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g. in footnotes);
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- the digital version of the paper submitted for the purpose of plagiarism assessment is fully consistent with the printed version.

I am aware that a declaration contrary to the facts will have legal consequences.

<Corinna Grabner, BSc> e.h. Klagenfurt, **<01.08.2021>**

<Please check against the AAU version of this page or embed the AAU version!>

)

Acknowledgements

Unser Dank gilt Raphael Wigoutschnigg, da diese Vorlage für wissenschaftliche Arbeiten der Gruppe Systemsicherheit (syssec) auf den LaTeX-Files seiner Diplomarbeit aufbaut.

Abstract

Infineon's Failure Analysis department examines the produced chips for errors in production and also deals with customer complaints. There are over 2,000 types of chips, which differ only minimally in layout. For the analysis, microscopes, lasers and X-Ray equipment is used. The department makes a very important contribution to the quality assurance of products at Infineon. If a product is not working properly, the resulting error code can be determined and the now called target product is sent to the failure analysis (FA) laboratory. Detection and localization of faults in semiconductors is a knowledge-intensive and tedious task. To increase the chances of success, an electrical engineer should be able to get all available information about the samples of similar past jobs. Various support systems used in Failure Analysis (FA), like databases, wikis, or file shares, often have this information stored as documents describing previous analysis reports of similar samples, best practices, specifications, customer reports, etc. However, accessing knowledge contained in these documents can be problematic, since in most cases, such support systems only provide rudimentary search functionality, like keyword matching. As a result, to find relevant information about jobs similar to the considered one, an engineer must query multiple systems, manually evaluate returned reports looking for similar characteristics, and asserting the value of each document for the current problem.
Modern Natural Language Processing methods (NLP) already showed their efficiency in various applications, including automatic translators, Recommender Systems or chatbots. Among these applications, text classification is one of the most promising to solve the FA search problem by automatically associating labels with a report denoting physical or electrical faults described in it, applied methods and tools, etc. The engineers can then use these labels to perform various tasks, like identifying similar jobs or getting statistics on possible faults, tools, or methods.
This is why one of the first applications of Artificial Intelligence (AI) tools at the FA laboratory of Infineon consisted on a classifier of the FA reports, using word2vec embeddings and clustering models, which satisfactory results.
With the apparition of more sophisticated tools in NLP during the last few years, the development of Transformer models until the publication of BERT, the results obtained by these models surpass anything utilized in previous times in almost every field where Transformers have been tested.

Zusammenfassung

Die Fehleranalyseabteilung von Infineon untersucht die produzierten Chips auf Produktionsfehler und kümmert sich auch um Kundenreklamationen. Es existieren über 2.000 Chiptypen, die sich nur minimal im Layout unterscheiden. Für die Analyse werden Mikroskope, Laser und Röntgengeräte verwendet. Die Abteilung leistet einen sehr wichtigen Beitrag zur Qualitätssicherung der Produkte bei Infineon. Funktioniert ein Produkt nicht richtig, kann der resultierende Fehlercode ermittelt und an das FA-Labor gesendet werden. Das Erkennen und Lokalisieren von Fehlern in Halbleitern ist eine wissensintensive und langwierige Aufgabe. Um die Erfolgschancen zu erhöhen, sollte ein Elektroingenieur in der Lage sein, alle verfügbaren Informationen über ähnliche vergangene Jobs zu erhalten. Verschiedene Unterstützungssysteme, die in der Fehleranalyse (FA) verwendet werden, wie Datenbanken, Wikis oder Dateifreigaben, haben diese Informationen oft als Dokumente gespeichert, die frühere Analyseberichte ähnlicher Proben, Best Practices, Spezifikationen, Kundenberichte usw. beschreiben. Jedoch kann das Zugreifen auf das enthaltene Wissen in diesen Dokumenten problematisch sein, da solche Unterstützungssysteme in den meisten Fällen nur rudimentäre Suchfunktionen, wie z. B. Keyword-Matching, bereitstellen. Um relevante Informationen zu Jobs zu finden, die dem betrachteten ähnlich sind, muss ein Ingenieur daher mehrere Systeme abfragen, zurückgegebene Berichte manuell nach ähnlichen Merkmalen auswerten und den Wert jedes Dokuments für das aktuelle Problem bestätigen.
Moderne Methoden der Verarbeitung natürlicher Sprache (NLP) haben ihre Leistungsfähigkeit bereits in verschiedenen Anwendungen unter Beweis gestellt, darunter automatische Übersetzer, Recommender-Systeme oder Chatbots. Unter diesen Anwendungen ist die Textklassifizierung eine der vielversprechendsten, um das FA-Suchproblem zu lösen, indem Etiketten automatisch mit einem Bericht verknüpft werden, der darin beschriebene physikalische oder elektrische Fehler, angewandte Methoden und Werkzeuge usw. bezeichnet. Die Ingenieure können diese Etiketten dann zur Durchführung verwenden verschiedene Aufgaben, wie das Identifizieren ähnlicher Jobs oder das Abrufen von Statistiken über mögliche Fehler, Werkzeuge oder Methoden. Aus diesem Grund bestand eine der ersten Anwendungen von Tools für künstliche Intelligenz (KI) im FA-Labor von Infineon in einem Klassifikator der FA-Berichte unter Verwendung von word2vec-Einbettungen und Clustering-Modellen, die zufriedenstellende Ergebnisse lieferten.
Mit der Entstehung immer ausgefeilterer NLP-Tools in den letzten Jahren, der Entwicklung von Transformer-Modellen bis zur Veröffentlichung von BERT, übertreffen die Ergebnisse dieser Modelle alles, was in früheren Zeiten in fast allen Bereichen, in denen Transformers getestet wurden, verwendet wurde.

Wichtige Hinweise zur vorliegenden LaTeX-Vorlage

- Diese Vorlage kann ohne Änderungen mittels PDFLaTeX von MikTeX compiliert werden. Daher müssen auch die Abbildungen als PDF vorliegen.
- Muss zur Compilierung aber LaTeX genutzt werden, da psfrag (oder Ähnliches verwendet wird),dann darf das Package hyperref nicht verwendet werden → Package hyperref (inkludiert am Beginn dieser Datei) auskommentieren und \newcommand{\href}[2]{#2} definieren. Abbildungen müssen dann als eps vorliegen.
- Damit das Literaturverzeichnis erzeugt wird, muss "bibtex thesis" aufgerufen werden.
- Damit das Abkürzungsverzeichnis erzeugt wird, muss "nomenclature.bat" aufgerufen werden.
- Einige Dokumente die den Umgang mit LaTeX und diversen Packages beschreiben, finden Sie im Verzeichnis "README".
- Textstellen (wie Datumsangeben oder Nafkmen) die anzupassen sind, wurden teilweise mit Platzhaltern der Form <**Bechreibung**> versehen! Nicht markierte Stellen der Titelseite, die gegebenenfalls anzupassen sind, umfassen
  - Bezeichnung des Studiums und
  - Name der betreuenden Assistentin bzw. des betreuenden Assistenten.
- Bitte Sicherstellen, dass die Titelseite und die Eidesstattliche Erklärung der aktuellen Fassung der AAU entsprechen. Die zugehörigen Seiten können auch auf der Upload-Seite der AAU erzeugt und in dieses Dokument eingebunden werden.
- Vor dem Ausdruck für die gebundene Masterarbeit ist die boolesche Variable "FINAL" (siehe Zeile 49 in dieser Datei) auf "true" zu setzen! Nicht ersetzte Platzhalter – das sind Kommandos der Form \ph{...} – werden dann beim Aufruf von PDFTex zu einer Fehlermeldung führen.
- Die aktuelle Version dieses Dokuments finden Sie auf der Website des Instituts für Angewandte Informatik `https://www.aau.at/en/ainf/teaching/templates`.

Table of Contents

1 Introduction

Infineon technologies Austria is one of the leading semiconductor companies worldwide. With around 4,820 employees, the company makes an important contribution to shaping the digital and networked future. Through the development of microelectronics and their constant improvement, Infineon enables efficient energy management, intelligent mobility and secure, seamless communication.

Introduction to failure analysis

Failure Analysis process?

Detection and localization of faults in semiconductors is a knowledge-intensive and tedious task. To increase the chances of success, an electrical engineer should be able to get all available information about the samples of similar past jobs. Various support systems used in Failure Analysis (FA), like databases, wikis, or file shares, often have this information stored as documents describing previous analysis reports of similar samples, best practices, specifications, customer reports, etc. However, accessing knowledge contained in these documents can be problematic, since in most cases, such support systems only provide rudimentary search functionality, like keyword matching. As a result, to find relevant information about jobs similar to the considered one, an engineer must query multiple systems, manually evaluate returned reports looking for similar characteristics, and asserting the value of each document for the current problem.
Modern Natural Language Processing methods (NLP) already showed their efficiency in various applications, including automatic translators, Recommender Systems or chatbots. Among these applications, text classification is one of the most promising to solve the FA search problem by automatically associating labels with a report denoting physical or electrical faults described in it, applied methods and tools, etc. The engineers can then use these labels to perform various tasks, like identifying similar jobs or getting statistics on possible faults, tools, or methods.
This is why one of the first applications of Artificial Intelligence (AI) tools at the FA laboratory of Infineon consisted on a classifier of the FA reports, using word2vec embeddings and clustering models, which satisfactory results.

goal of the project

1.1 Problem Definition

The goal of this project is to develop a FA report classifier with a BERT model. In order to do so, the task has been divided into two phases:

First, we have developed a Language Model based on the state-of-the-art model BERT. Therefore we had to consider our specific domain and select the most appropriate model for our domain. Since many successors of BERT have been developed but none of them aimed at the electrical domain, we had to select a model as close as possible to our field of study in order to achieve a satisfying performance later.
Second, we focused on defining the structure of the classifier, which consisted on the BERT network and additional classification layers. To test the results, we have defined a series of classification problems based on the FA reports.
Given the nature of this project the two phases were developed in parallel so the joint performance hasn't been tested yet.

1.1.1 Research Questions

1.2 Research Question 1: Finding the best BERT Model

1.3 Research Question2: Named Entity Recognition

The goal of this project is to develop a FA report classifier with a BERT model. In order to do so, the task has been divided into two phases:
First, we have developed a Language Model based on the state-of-the-art model BERT. Therefore we had to consider our specific domain and select the most appropriate model for our domain. Since many successors of BERT have been developed but none of them aimed at the electrical domain, we had to select a model as close as possible to our field of study in order to achieve a satisfying performance later.
Second, we focused on defining the structure of the classifier, which consisted on the BERT network and additional classification layers. To test the results, we have defined a series of classification problems based on the FA reports.

general structure of the project

In order to fulfill this goal, an innovative language model trained on the semiconductor domain had to be developed. Therefore it was necessary to find an existing language model to use it as entry point and further train it on a domain specific dataset to achieve the desired results. This specific dataset had to be collected by myself and transformed into a shape which allows to train a BERT based language model.

Structure of the thesis

The following chapters

2 Literature Review

2.1 Natural Language Processing

Communication and the sharing of knowledge is an essential part of human society. History has shown that the best way to transmit knowledge is in writing. It is not difficult for a person to learn to read and to understand the context in texts. For computers, this task is much more difficult. Natural Language Processing (NLP) is a subfield pf Artificial Intelligence (AI) which deals with teaching computers to understand and interpret human language and has emerged in 1940. The initial need of NLP was the translation from one language into another which was used during the second world war. Nowadays it is widely used in health care, spam detection, sentiment and cognitive analysis. NLP also provides computers with the ability to read text, hear speech, and communicate with humans. Every person had contact with a NLP device at least once in their life. The best example everyone might know are personal assistants like Siri or Alexa. These are applications which work with NLP and have learned to communicate with humans and nearly seem humanely.
Generally speaking, NLP breaks down language into shorter, more basic pieces, called tokens (words, periods, etc.) which get saved as vectorial representations. The technique of mapping words to real vectors is called word embedding. To understand the relationships of the tokens, the model gets trained by predicting some hidden part of the text using some other part of their surrounding text. One way to calculate the representations of the vectors is the method *Word2Vec*. Until BERT has emerged, Word2Vec was the most common technique to create word embeddings. It is a two-layer neural network which processes text and turns it into a numerical form that can be understand by deep neural networks. Given enough data, Word2vec can make highly accurate guesses about a word's meaning based on past appearances. Those guesses can be used to establish a word's association with other words. The diference between Word2Vec and BERT will be covered in Section 2.3.1.

2.1.1 Tokenization

Texts cannot be processed by ML models directly. Tokenization is a method that transforms sequences of characters into a sequence of integers. As an example we can take the sentence:

"This is a cat."

A tokenization transforms this sentence into [‘This’, ‘is’, ‘a’, cat’]. Punctuations will be removed. This step is important to help the model understanding the meaning of a text. The model can

- Count the number of words in the text
- Count the frequency of the word, that is, the number of times a particular word is present

In modern LMs, tokenizers are trained together with the model to identify the best possible transformations, which can happen on both word and sub-word levels. The set of obtained tokens determines the vocabulary of an LM. If a word appears in the vocabulary of LM, it can be represented as a vector in the target space. Otherwise, a word is split into parts until each of them can be mapped to a set of tokens from the vocabulary. In the worst case, a word is split into individual letters. Such splitting may significantly reduce the quality of word embeddings in the vector space, thus reducing the quality of features extracted for the classification layers of a network. Therefore, it is essential either to select an LM whose vocabulary provides good coverage of the main terms used in an application domain or to extend the vocabulary of an LM with these terms and fine-tune it on domain-specific texts.

2.2 Transformer

The transformer in the field of NLP is a new architecture which is able to solve sequence-to-sequence tasks while handling dependencies in the text. Transformers basically consist of an encoder and a decoder. The encoder reads the input text and the decoder produces a prediction. Transformers work in small increments. In each step, an attention mechanism is applied to understand the relationships between the words in a sentence, regardless of their positions.

2.3 Evolution of Language Models

2.3.1 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It makes use of a transformer and an attention mechanism that learns contextual relations between words or sub-words in a text. The BERT model was developed by Google and is already pre-trained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia. In contrast to previous language models which looked at a text sequence either from left to right or combined left-to-right and right-to-left training, BERT is using the bidirectional approach. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. To do this, the authors introduced a new technique which is called Masked Language Modelling (MLM).
A Transformer usually includes two separate mechanisms, an encoder that reads the text input and a decoder that produces a prediction for the task. For producing a language model, only the encoder mechanism is necessary. Previous models have read the text input sequentially either left-to-right or right-to-left. BERT's transformer encoder reads the entire sequence of words at once. This allows the model to learn the context of a word based on all of its surroundings [**?**].

The input to BERT's encoder is a sequence of tokens previously converted into vectors. Later, these tokens can be processed in the neural network. In order to be able to do this, however, a few steps must first be carried out:

1. CLS and SEP tokens at the beginning and end of each sentence
2. Segment embeddings for each token to distinguish between sentences.

3. Position embeddings for each token to identify the position in a sentence.

2.3.2 Difference from BERT and word2Vec

Word2Vec is contex-independent, so it only has a numeric vector to represent a word. If a word has several meanings, these are combined into a vector.

BERT, on the other hand, is context-dependent and thus allows multiple numeric vectors as a representation for a word, depending on the context in which the word occurs.

An example of the difference is the word bank, which can appear in a financial context as well as in a beach or park context. Word2Vec will always generate the same vector for this word and can therefore lead to an inaccurate representation. BERT can distinguish the two different semantic meanings and thus also generate two different vectors.

1. The next difference is that Word2Vec doesn't care about the position of words in a sentence. BERT, on the other hand, uses the position (index) of a word as input for calculating the vector.
2. Word2Vec only needs one word as input and delivers a vector as output. BERT, on the other hand, needs an entire sentence as input because it needs the context of the sentence to calculate the vector.
3. Word2Vec can have problems if a word is not stored in the vocabulary and no vector can be generated. BERT can also create a vector for subwords that are not stored in the vocabulary and is therefore not limited by the vocabulary.

2.3.3 From BERT to SciBERT

The development of BERT was a milestone in Natural language processing. Since the vocabulary of BERT is restricted, the application of the model in some domains does not lead to a satisfactory performance. To achieve this goal in different scientific fields, the model of BERT got adapted to SciBERT. The SciBERT model follows the same architecture as BERT but is instead pre-trained on scientific text. The corpus consists of 1,14M papers derived from *Google Scholar*, 18% papers from the computer science domain and 82% from the broad biomedical domain. The authors used the full text of the papers instead of only the abstracts. The average paper length is 154 sentences which results in 2.769 tokens. The final corpus size has 3.17B tokens, similar to the 3.3B tokens on which BERT was trained [**?**].

The S2ORC-SciBERT is a SciBERT model which was further trained on the S2ORC dataset presented above. It got published simultaneously with the dataset itself from the same developers. Compared to SciBERT it has a wider corpus including 20 different domains like Computer Science, Mathematical Science and Engineering which could be beneficial for the FA domain. However it was also trained on Medical and Biology data which again could be a drawback for the model as with SciBERT.

2.4 Masked Language Modelling

2.5 Named Entity Recognition

3 Data Collection

In order to train a language model, a sufficiently large amount of data is needed. Since the model presented in this paper will be used for textual classification, the data must be available in written form. Most of the data used for the report classification come from the FA laboratory. The laboratory has two sources to save information: an ontology which is the formalization of FA knowledge in a computer-friendly format, and the FA reports which comprise information that describes all diagnostic steps, results, and observations for a job analysis. These two resources would have a large enough amount of data to train the model. However, experiments with previous models indicated that these documents are too specific to serve as the sole data source for a model in the semiconductor domain. The content of the documents in the ontology and the FA reports contains too specific vocabulary and too little general vocabulary about electronics and semiconductors. Models which were trained only with this data did not have sufficient performance. For this reason, additional sources were sought to train the model more extensively. This chapter starts by introducing the inddividual datasets and how they got collected before covering the data cleaning process to prepare it as an input for the PikaBERT model.

3.1 Data Sources

The remaining dataset was created out of different data sources which got categorized in four datasets for an easier use in later experiments. The four datasets are: the *FA ontology and reports*, the *S2ORC Dataset*, the *Infineon Dataset* and the *Additional Dataset*. The following chapters introduce every dataset with its content, size and chatateristics.

3.1.1 Failure Analysis Ontology and Reports

The FA laboratory stores most of its knowledge as free texts, thus they might be ambiguous and cannot be processed by the software automatically. To avoid these issues, standard definitions of FA concepts used in the domain are required. Moreover, these definitions must be stored in a way that they can be used by both engineers and software tools alike. One possible solution to this challenge is to formalize the knowledge about the FA domain as an ontology, a knowledge base specifically designed to store terminological definitions. This is the case for the FA laboratory, where an ontology is being developed, currently storing hundreds of failures, tasks, tools, etc. The structure of an ontology includes classes, instances of these classes and properties. Individuals are descriptions of real-world entities, like sample integrated circuits of a job or tools available in a lab. Classes are defining parts of the world by summarizing properties of a collection of individuals.

For training the classification models, we have considered the historical data of the FA laboratory. These reports contain a series of fields, including the job identification number, cus-tomer comment regarding the issues found in devices, an anal-ysis report describing applied methods, found physical defects, their locations, electrical characterization, and other details. Some of this data is structured and can be retrieved from corre-sponding databases. However, the essential parts relevant to the fault identification process are described only in textual form and, therefore, cannot be processed automatically. Consider three samples taken from fault analysis reports of an FA Laboratory presented in Error! Reference source not found.. The examples show selected sentences describing the findings of an engineer and the labels indicating the physical faults and their electrical signatures. Both types of labels are organized in an ontology, which is a hierar-chical structure rep-resenting a taxonomy of faults. This ontology allows for the de-velopment of software tools helping to label reports manually and working with the classification results. Along with the job summary, the reports contain a series of fields including, an identification number of the device, lists of tasks performed on the device, the images obtained by certain tools such as X-Ray,the description of such images attached, etc. A subset of the reports also includesfault labels, both of the electrical signature of the faulty device and of the final physical failure.For the classification phase, these two faults are considered the target output while the job summary and image descriptions will constitute the input.

3.1.2 S2ORC Dataset

The S2ORC Dataset was collected by the Allen Institute of Ar-tificial Intelligence and it has a corpus of 81.1M English-language academic papers from different domains. Around 40% of the papers belong to the biomedical field, as in Figure 3.1. The texts got extracted from PDFs including abstracts and inline mentions of citations. The S2ORC-SciBERT is a SciBERT model which was further trained on the S2ORC dataset presented above. It got published simultaneously with the dataset itself from the same developers. Compared to SciBERT it has a wider corpus including 20 different domains like Computer Science, Mathematical Science and Engineering which could be beneficial for the FA domain. However, it was also trained in Medical and Biology data which again could be a drawback for the model as with SciBERT.

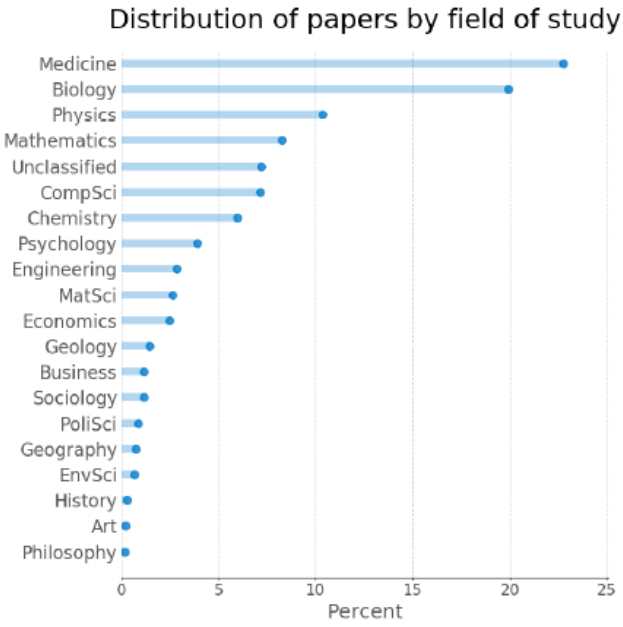

Fig. 3.1: Different Domains in the S2ORC Dataset.

3.1.3 Infineon Dataset

In addition, we created an Infineon Dataset comprising rele-vant textual data available on the company's intranet. This dataset comprises 1.687 papers covering research topics and best-practices methods in the semiconductor domain. It was collected by employees and engineers of Infineon . Be-cause these papers got collected from engineers in different countries, we had to filter out non-English language papers. After filtering and extracting the raw text, we used approxi-mately 31.5MB of text data to create a dataset.

3.1.4 Additional Data

To increase the specificity of the training data for the FA domain, we also searched the Web for FA and electrical en-gineering papers. In this work we were using open search engines, like FreeFullPDF and GoogleScholar , as well as specific sources, like IEEE, to collect the data. The text was extracted from the papers and converted into a text represen-tation. The resulting dataset comprises 12.31MB of raw text, where 2.33MB got collected from FreeFullPDF and Google-Scholar and the other 9.98MB from IEEE.

3.2 Data Preprocessing

For later usage during training, we combined the Infineon and the Additional Datasets in one. Moreover, since the biggest part of the S2ORC Dataset was medical and biomedical data, we filtered this dataset by using the FA ontology keywords, retaining only documents that comprise at least one keyword. The resulting filtered dataset contains approximately 542MB of raw text. For later discussions, we introduce the following abbreviations for the datasets:

- **inf**: Infineon Dataset + Additional Dataset;

- **s2**: a part of the documents from S2ORC dataset fil-tered on keywords extracted from the ontology.

After finishing collecting the papers for the dataset, it was necessary to extract the raw text out of the documents. Therefore we got access to an text extractor API developed by Infineon collegues in Bangalore. The raw texts got extracted and UTF8 converted to be saved as excel sheets.
Not all of the extracted text is important or useful for the dataset. To remove unuseful data, a notebook was created were the text got prepared before being cleaned. Authors, metadata about the images, tables and other not contiguous text was remoed. The resulting text got again saved as excel sheets.

3.2.1 Data Cleaning

To use the FA reports for AI purposes, their content must be pre-processed and cleaned to erase all data that can disrupt the training process. One of the biggest issues regarding the FA reports is the wide variety of styles in which they are written, as they come from several laboratories around the globe, some of them dealing with specific purposes, while others deal with a broader range of failures. It is also essential to consider that although all they are written in English, this is not a standard language as engineers are only rarely native speakers.
The data cleaning process was the same for all datasets. A notebook was created were the excel sheets got loaded into dataframes. The methods used for cleaning the text are imported from the Natural Language Toolkit (nltk) package which implements functions for normalization, stemming and lemmatization. The following steps describe the procedure of text cleaning:

1. **Remove Punctuations and Special Characters**: Punctuatins and special characters got removed inbetween and at the end of every sentence by using regular expressions because a language model is not able to interpret those characters.
2. **Remove Numbers**: The input got checked if it is a number or not. Also all words which are not strictly all letters got removed to avoid confusing the model.
3. **Normalization**: After removing undesired words and characters, the entire text got normalized. This means that the resulting text contains only out of lowercased letters.
4. **Remove Stop-Words**: The nltk package contains a collection of english stop-words like "and", "or", "but", "for" etc. Due to their low entropy they are useless for the training and therefore removed from the text.
5. **Lemmatization**: Lemmatization considers the context of a word and converts it to its meaningful base form, which is called Lemma. For instance, lemmatizing the word "Caring" would return "Care". This has been done for the entire dataframe by iterating over the words.
6. **Remove Empty Lines**: Due to the previous cleaning some cells or lines of the dataframe may be empty now and represented as NaN values in the dataframe. These NaN values have to be replaced with spaces, otherwise they will cause errors during further processing.

3.2.2 Dataset Formatting

In comparison to traditional neural networks, BERT-based models are like a black box for the engineer. For training the model we used the so-called "transformer" class from Huggingface. The trainer method of the transformer class needs a model, a tokenizer and the training dataset as input. This input dataset needs to have a specific format and datatype otherwise the function would rise an error. To do so the previous created dataframes got converted into dataframes of two cells, an index and a text cell. An example is shown in Table 3.1.

	text
0	The following parameters are most often considered...
1	Nitrogen fertilization stands out as an important...

Tab. 3.1: Example of transformed dataframe shape

The base class *Dataset* from Huggingface implements a Dataset backed by an Apache Arrow table. This type of object can be created out of the dataframe just created with the following code:

```
train_dataset = Dataset.from_dict(df)
datasets = datasets.DatasetDict({"train:" train_dataset})
```

The dataset object allows to specify a training and a test dataset. Since training the language model is an unsupervised learning method, only the training dataset is needed. The resulting dataset objects were saved and could be loaded later for training.

4 Analysis

Introduction to chapter

All of the following steps to prepare the dataset for training the model were taken from an online notebook created by *Google*.

4.1 Training Environment

4.2 Training Pipeline

4.2.1 Finding the Model Entrypoint

Since BERT models studied in this paper were trained on different text corpora, their tokenizers might provide different coverage of important terms used in the FA domain, such as the ones stored in FA ontology. To determine the coverage, the first task was to analyze the tokenization of the ontology keywords with the three language models: traditional BERT, SciBERT and S2ORC-SciBERT. With this analysis it was possible to gain a first impression about the vocabulary of the models and how well they fit to the electrical domain.
To receive the keywords, we executed an existing function provided from our supervisor Christian Burmer. It executes a script with a request to the SparQL Database receiving the ontology keywords in a list. This keywords also occur in the reports as indicators for the content.
After receiving the keywords from the ontology and having a deeper look at them, we found some misspellings like in the word *flasover*, which misses a h to become *flashover*. These typos could confuse the models and lead to some mistokenization. To prevent this, we will first run some spell checking and correction methods on the keywords before further processing it. Therefore we tried different packages like the *Pyspellchecker* or *SymSpellPy* which are based on the *Peter Novig's method*. As both of the two packages did not work, we skimmed the keywords manually and corrected misspellings.
With the correct spelled keywords we started the experiment of tokenizing them with the three different BERT models. The code for tokenization was the same for every model. We created a loop which hands every word over to the tokenizers and also analyzes if the resulting word is included in the models vocabulary or not. If it is not included in the vocabulary, we counted the number of good and badly tokenized words

4.2.2 S2ORC-SciBERT Fine-Tuning

Also the tokenizer itself can be improved before using it in the training method together with the model. Training a to-kenizer can be done in two ways:

1. Extending the vocabulary manually
2. Pre-training the tokenizer

Tokenizers of BERT-derived models usually have empty positions allowing us to add required tokens directly. Extend-ing the vocabulary of a tokenizer means that it won't get trained explicitely but important words like our ontology keywords can be added to the vocabulary of the tokenizer and therefore recognized as one single token by the model later. Pre-training the tokenizer means that the tokenizer will get trained on a prepared dataset without using the model. Again the engineer has no impact on the behavior of the tokenizer during the training loops. We created four types of trained tokenizers. For later discus-sions we introduce the following abbreviations:

- **ext**: A tokenizer with the ontology keywords added to its vocabulary manually
- **small**: A tokenizer trained on inf dataset
- **large**: A tokenizer trained on inf+s2 dataset;
- **flt**: A tokenizer trained on the s2 dataset

Tokenizing dataset split into blocks MLM Method different models

5 Results

5.1 Performance Measures

5.2 Discussion of Results

Inhalt...

6 Conclusion

6.1 Future Work

Bibliography

[Ashb00] J. D. M. Ashbourn: Biometrics : Advanced Identify Verification: The Complete Guide. Springer Verlag (2000).

[BeKH01] K. Beuth, G. Kurz, R. Hanebuth: Nachrichtentechnik. Vogel Fachbuch, Vogel-Verlag KG., Wrzburg (2001).

[BeKP91] A. Beutelspacher, A. G. Kersten, A. Pfau: Chipkarten als Sicherheitswerkzeug. Springer Verlag, Berlin (1991).

[BiSh91] E. Biham, A. Shamir: Differential Cryptanalysis of DES-like Cryptosystems. *In: CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, London, UK (1991), 2–21.

[Breu84] R. Breuer: Computer-Schutz durch Sicherung und Versicherung. Karamanolis-Verlag (1984).

[Chen00] Z. Chen: Java Card Technology for Smart Cards: Architecture and Programmer's Guide (The Java Series). Addison-Wesley (2000).

[DaRi00] J. Daemen, V. Rijmen: The Block Cipher Rijndael. *In: CARDIS '98: Proceedings of the The International Conference on Smart Card Research and Applications*, Springer-Verlag, London, UK (2000), 277–284.

[JoMV01] D. Johnson, A. Menezes, S. A. Vanstone: The Elliptic Curve Digital Signature Algorithm (ECDSA). In: *International Journal on Information Security*, 1, 1 (2001), 36–63.

[MeVO96] A. J. Menezes, S. A. Vanstone, P. C. V. Oorschot: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA (1996).

[NIST77] NIST: Data Encryption Standard, *Federal information processing standards publication*, Bd. 46. National Institute for Standards and Technology (1977).

[NIST00] NIST: FIPS PUB 186-2 Digital Signature Standard (DSS). National Institute for Standards and Technology, Gaithersburg, MD, USA (2000), `http://www.itl.nist.gov/fipspubs/fip186-2.pdf`.

[RaEf02] W. Rankl, W. Effing: Handbuch der Chipkarten. Carl Hanser Verlag Mnchen Wien (2002), 4., berarbeitete und aktualisierte Auflage.

[Rank06] W. Rankl: Chipkartenanwendungen. Carl Hanser Verlag Mnchen Wien (2006), entwurfsmuster fr Einsatz und Programmierung von Chipkarten.

[RiSA78] R. L. Rivest, A. Shamir, L. M. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the ACM*, 21, 2 (1978), 120–126.