Enhancing insights into spending through aggregation with automated document clustering of a large-scale multilingual corpus

Bachelor Thesis

Part of the Examination for the
Bachelor of Science (B.Sc.)
of
International Business Administration and Information Technology
at the University of Business and Society Ludwigshafen

by

Lisa Rebecca Mirjam Schmidt
Sternstraße 93
67063 Ludwigshafen am Rhein

Date of submission:	07.06.2022
Company Supervisor:	Dr. Karthik Muthuswamy
Academic Supervisor:	Prof. Dr. Joachim Melcher

Contents

List of Abbreviations

AI	Artificial Intelligence
AI BUS	SAP AI Business Services
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CBOW	Continuous Bag of Words
CPU	Central Processing Unit
CRISP-DM	Cross Industry Standard Process for Data Mining
CoE	Center of Excellence
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DR	Dimensionality Reduction
ERP	Enterprise Resource Planning
FFNN	Feed-Forward Neural Network
GIL	Global Interpreter Lock
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
KDD	Knowledge Discovery in Databases
LDA	Latent Dirichlet Allocation
ML	Machine Learning
MLM	Masked Language Model
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
PCA	Principal Component Analysis
SEMMA	Sample Explore Modify Model Assess
sklearn	scikit-learn
SQL	Short Query Language
TF-IDF	Term Frequency - Inverse Document Frequency
t-SNE	T-Distributed Stochastic Neighbor Embedding
TPU	Tensor Processing Unit

List of Figures

List of Tables

Listings

1. Introduction

1.1. Motivation

More than 4 out of 5 financial departments are "overwhelmed by the high numbers of invoices they are expected to process" [7]. Already swamped departments of course struggle with providing information on savings potential. A solution for processing large amounts of invoice data with minimal human interference is desirable. Processing this information should transform unstructured data into a useful format, further the data should be enriched in a way giving insight and provide value for financial advisors. With analysis results of this automated processing, financial advisors can enjoy two improvements. Firstly, they have a factual base for recommendations. Secondly, the financial department is able to focus on tasks which are more value-adding, if an automated solution completes this time-consuming task of an in-depth analysis of spending.

An invoice is a document recording the main information on a sales transaction. Usually, an invoice contains the unit cost, a timestamp, and payment terms. Other information, such as shipping terms, shipping address, or discounts may also be included. Apart from their use for tax records, tracking the inventory and legal protection, invoices are essential for a company's financial reporting. Invoices are the main source of information for controlling [18], as they record the complete history of cash flow [49]. In general, internal financial reporting aims to provide information about the health of a company, and supply means for improvement.

1.2. Current Situation

An essential part of economic counseling is the assessment of spending for different company segments. Spending of a firm usually is written down in invoice documents, which have to be grouped to analyze cost types. The global market is estimated to comprise 550 billion invoices annually, but 90% are exchanged paper-based [20]. With

modern technology, these paper-based or digital documents can be transformed into a structured or semi-structured format. According to expert estimates, unstructured data makes up for more than 80% of enterprise data [47]. Companies can not utilize unstructured data to its full potential, as this data is not able to be leveraged with traditional data analysis tools.

1.3. Research Questions

How can information from invoice-like documents be extracted? How can extracted information provide insights?

1.4. Outline

The introductory chapter briefly explains the motivation behind the thesis. Also, the current situation is assessed and research questions are presented. Lastly, it presents the structure of the thesis.

The following chapter about objectives and criteria gives a detailed task description, and established criteria. In the section 2.2 the process model is explained, evaluated and adjusted.

Chapter 3 gives a glossary of terms. Chapter 3 sheds light on the corporate environment with respect to the aspects of history, organization an technological aspects.

The following chapters follow the structure of the research model presented in 2.2.

Chapter 4 explains the process of selecting a dataset. It presents fundamental types of projects and data sources. The chapter explains the sourcing of the used dataset.

Chapter 5 explains the business understanding.

Chapter 6 explains the data understanding.

In chapter 7, the preparation of the data is explained. The process of data cleaning is presented, as well as different means for feature extraction.

Chapter 8 compares algorithms and alternatives for measuring distances.

Chapter 9 presents the evaluation of the result. The chapter evaluated the output from the previous steps.

In chapter 10, a conclusion is drawn, and a further outlook is given.

2. Objectives and Criteria

2.1. Detailed Task Description

The goal of this thesis is adding value to real business documents by transforming unstructured data into a structured format. The structured information can then be used as a base for further analyses.

The task is to perform a full data analysis on the supplied dataset. The dataset is to be prepared for processing with established methods. An evaluation for different means of feature extraction, machine learning, model evaluation and visualization should be performed. With the evaluation a complete flow for the data processing should be presented. The result is an added value to the dataset by creating structured data and providing insights into this data.

2.2. Criteria

The task includes different criteria from a corporate perspective which need to be considered. The analysis should be performed utilizing only available resources, which are the student's company laptop and already available instances for SAP internal services. The dataset which will later be presented is only available to SAP employees, and only after an access request for a specific use case is approved. Therefore, the tasks need to be completed with special attention to data protection.

Additional quality criteria were established before the start of the work. Firstly, the solution should be designed according to industry standards. This also includes the choice for a fitting research model. Secondly, the source code is to be documented in such a way, that an expert third party can understand the workings of it in an appropriate time. Thirdly, the design of the source code should account for existing hardware limitations and should be optimized computationally.

2.3. Research Model

To solve the task described in chapter 1.2, this paper employs the Cross Industry Standard Process for Data Mining (CRISP-DM) [11]. This model puts forward a structure for conducting data mining projects. CRISP-DM was developed in 1996 by three companies, which are now the partners of the CRISP-DM consortium: NCR, DaimlerChrysler AG and SPSS Inc.

A poll [36] conducted among visitors of a data science project management blog found that almost half of all respondents employ the CRISP-DM process model. Followed by Scrum and Kanban with a 18% and 12% of the user share, CRISP-DM is by far the most popular. Other methods such as Knowledge Discovery in Databases (KDD) and Sample Explore Modify Model Assess (SEMMA) are also noteworthy alternatives, but are less popular that CRISP-DM, Scrum and Kanban.

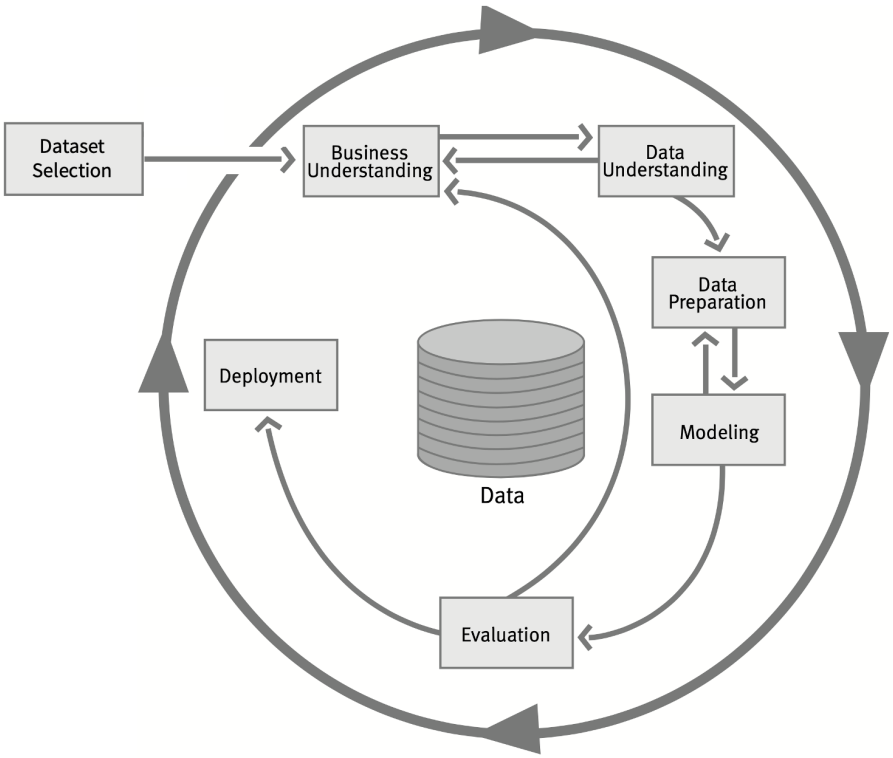

Figure 2.1.: Adjusted CRISP-DM Model

Being the most popular model, CRISP-DM is not necessarily always the best fit for all data science projects. In the case of this particular research effort, CRISP-DM proved

the best suit for several reasons. Firstly, the process model follows the natural intuition of project design for data science tasks. Evaluation has to occur before the deployment, the modeling needs to occur before the evaluation, the preparation needs to occur before the modeling, and an adequate understanding of business and data aspects has to be developed in the beginning of the process. All those elementary dependencies are reflected in the model. Secondly, the CRISP-DM model addresses the iterative nature of data mining. Fundamentally, the model is of circular nature, reflecting the fact that data science projects require continuous improvement. After the deployment of one solution, monitoring can give insights which allow for deeper business understanding, triggering the start of a new circuit of the model. Another model which puts forward an iterative approach is KDD [16]. Thirdly, the model allows to adapt the order of its phases. The free choice of path is more favorable compared to KDD, which allows for loops, but has a fixed order [16]. Fourth, CRISP-DM has no special requirements regarding team size or roles. Instead, a CRISP-DM project can be completed by only one person. This stands in contrast to SCRUM, which needs different roles represented by people in the team to work effectively [17]. Fifth, the CRISP-DM model was publicized in the context of a 70-page guide with generic task descriptions and outputs for each phase. The detailed guide is especially valuable for teams with little experience.

Classically, the reference model consists of six phases. For this thesis the model was adapted to reflect all tasks encompassed. The phase "Dataset Selection" was added, resulting in a total of seven phases. The newly added phase includes the selection of a suitable dataset, the data retrieval, and the data provisioning. This results in a process model adapted to this specific project, and lays the groundwork for a successful undertaking.

3. Corporate Environment

3.1. Historical

SAP was founded in 1972 by five former IBM employees. The original company name was "Systemanalyse Programmentwicklung", which can be translated to "System analysis and program development". In 1976, a second company, the SAP GmbH was founded, where the acronym SAP denoted "Systems, Applications and Products for data processing" [38]. The SAP GmbH is the company, which is today known as SAP SE.

Data processing being part of the company's name shows the importance of this field to SAP since the beginning of the company history. In 2017, SAP entered the Artificial Intelligence (AI) business with the SAP Leonardo Machine Learning Foundation [35]. SAP challenged the market for hyped products in the sectors machine learning, blockchain, big data, and design thinking. The name Leonardo refers to Leonardo Da Vinci, who is renowned for his interdisciplinary innovations [41]. SAP has the goal of driving the digital innovation strategies of customers with the help of SAP Leonardo.

With changes in the underlying hyperscalers for Leonardo, and evolving requirements of customers and partners, SAP adjusted their AI strategy. Two new products were introduced: SAP AI Core and SAP AI Launchpad. Both products are united under the collective name of AI Foundation [35]. With the general availability of AI Foundation in late 2021, SAP Leonardo is officially sunsetted.

3.2. Organizational

SAP SE has an executive board consisting of seven members, each attributed to one area. The AI division falls into the responsibility of Jürgen Müller, Chief Technology Officer and leader of the board area for technology and innovation [39]. Members of the organizational unit for AI are divided in different teams concerned with development, product success, operations and specific AI-services. Development Teams are organized

into Centers of Excellence (CoEs), in which special expertise for designated areas is united. SAP has a team of researchers around the world concerned with state-of-the-art topics, including few-shot learning, sentiment analysis, privacy and fairness [37]. The research teams regularly publish articles on their advancements.

4. Dataset selection

4.1. Choosing a Data Source

With the project type as a decisive factor for the dataset explained, the other evaluation criteria for the dataset is described. In the corporate environment two fundamental sources for data exist.

4.1.1. Potential Data Sources

Firstly, data can be sourced from inside the company. This can include customer data or data generated from observation and monitoring processes inside the company [34]. Data is either directly or very closely related to the company's business. Because internally sourced data is of utter utility and a possible target for industrial espionage, internally sourced data is almost exclusively rated confidential, limiting even intra-company access to it. Authorization processes and more than often not existing registries for data may hinder project progress.

Secondly, data can be sourced outside the company. A vast number of online registries for data exist, both with paid and free of charge service offerings [9]. Data sources include social media data, sensory data and weather data. Because of its publication, the data is sometimes stripped from all parts which could expose confidential information such as corporate secrets. Additionally, data is anonymized for privacy reasons. External Data can give valuable insights into industries and markets especially for data scientists without access to paid databases, or for small businesses without the option for an own data collection.

4.1.2. Selected Data Source

It can be stated, that both sources are suited for different goals and different contexts. For data scientists with internal sources available, this type of source seems more appealing.

Table 4.1.: Comparison of Data Sources

	Internal	**External**
Source	Internal or customer data, bought or generated	Publicly available, generated or supplied by companies
Relevance	Business-relevant	Anonymized, processed
Value	Relevant specific business	Of general relevance
Availability	Authorization processes in place, data protection measures	Ubiquitously available
Examples	Sales records, usage statistics, customer feedback	Historic weather information, consumer statistics, social media data

An approach of connecting both internal and external data is an option, but not within the scope of this project. For this project, internally sourced data is chosen as a source.

4.2. Choosing a Dataset

The dataset is the foundation of a data-science project. The quality, size, and closeness to reality decide the helpfulness of findings made using the data. In this section, a classification for data-science projects is introduced as a guide for dataset selection.

4.2.1. Project Types

The problem-first project type is characterized by a predefined problem statement or research goal. The underlying question is how a specific problem can be solved [24]. For the selection of the dataset, this requires that goal achievement can be measured with existing ground truth. In some cases, also other methods such as expert judgments can be sufficient. In a problem-first project different datasets can and should be considered. [24] give the metaphor of "mining for valuable minerals or metals at a given geographic location where the existence of the minerals or metals has been established". This means

Table 4.2.: Fundamental Data Science Project Types

	Problem-First	**Data-First**
Systematics	Applied Data Science	Exploratory Data Science
Underlying Question	How can a problem be solved?	Which problems can be solved with the solution?
Role of the dataset	Different datasets can be considered for one problem statement.	The dataset is the core of the project, with a new dataset, a new project begins.
Requirements for the dataset	Require a ground truth or established methods for evaluating the goal achievement.	Compared to problem-first projects, larger datasets are required because there is no prior assumption of patterns.
Fixed component	Problem statement.	Dataset.
Innovative Aspect	Defined during formulation of the goal.	Found during the project.

that it is already verified that the business goal can be reached with this specific dataset, hence the metaphor of already discovered mineral occurrence.

The complementary project type is the data-first project. This type is characterized by a more exploratory approach, and the goal to find problems and patterns. Here, the dataset is at the core of the project and the fixed aspect of the project. In turn, this means swapping the dataset is the start of a new project. When turning to the metaphor of mining for minerals, this type of project would be the exploration of different test pits that promise mineral occurrences [24]. It is not clear if a problem can be solved with the investigation, and problem to solve is not known.

4.2.2. Systematization of this Project

Usually, the project type is not decided on, but implicitly arises out of environmental parameters. The goal of the analysis is clearly stated in before chapters. Still, the research

questions (1.3) aim at finding out which insights the analysis can give. Therefore, this project can be classified as a data-first project.

4.2.3. Resulting Requirements for Selecting a Dataset

In a data-first project, there is no assumption of patterns. This for once means that a large dataset is required to cover enough ground for accurate derivation of insights. Second, a data-first project does not require structured data [46]. Instead, unstructured and semi-structured data is also applicable for data-first projects.

4.3. Selected Dataset

With the requirements explained, a dataset fulfilling all criteria was found. The selected document dataset consists of 150.000 invoices. The data contains information about the vendors, billing amounts and a descriptions of the goods.

5. Business Understanding

In the guide complementing the CRISP-DM model, different tasks, and outputs for developing a business understanding are mentioned. The task and respective output will be discussed in the following sections.

5.1. Business Objectives

Businesses without existing an Enterprise Resource Planning (ERP) solution in place can easily be overwhelmed by the number of invoices reaching them daily. Even more, the controlling department can easily lose the overview of spending. To quickly gain a perspective on the most important spending topics, spending should be sorted in categories of similar nature.

The primary goal is the development of a solution for automatic aggregation of documents, based on topics addressed in those documents. The focus is on shorter text segments, such as product descriptions. The business objective is an information gain, on how spending is distributed among cross-cutting topics in a company.

The created solution can be evaluated with the business success criterion: "Does the solution identify and give useful insights in the money pits?". The judgment of goal achievement is a subjective matter. Evaluating the success should therefore be distributed among several stakeholders, including but not limited to the author, the supervisor and the supplier of the data.

5.2. Assessment of the Situation

5.2.1. Inventory of Resources

An inventory of resources (A.1) was created for assessing the situation. Most notably is the availability of experts through excellent inter-corporation cooperation. Also, a large

collection of datasets is available. Hardware platforms include personal machines as well as hosted environments with GPU capabilities. Available software are data science tools included in the Anaconda Navigator, such as Jupyter Notebook. All open-source libraries are of course also included.

5.2.2. Requirements, Assumptions, and Constraints

The project is to be completed the latest on June 7^{th} 2022.

Several assumptions underlay the process of data mining. First, it is assumed that the descriptions of the invoices is speaking enough to identify the product referenced. Second, the analysis assumes that invoices can be logically grouped into clusters, in other words, several invoices referring to similar topics.

From a legal perspective, the project is constrained in the publication of data. While the use and processing of the supplied data is permitted within the context of the thesis, publication and further use is prohibited. The dataset is to be kept only on the local machine and SAP owned hyperscaler instances.

5.2.3. Risks and Contingencies

Dataset too Large for Processing One specific risk that may arise in this project, is a dataset too large to be processed. Only a limited size of data is able to be loaded into memory. If the dataset turns out to be too big, four solutions are proposed.

1. The dataset can be compressed using either a lossy compression (sampling, truncating floating point values) or a lossless compression (choosing only specific columns, using a sparse-column representation or choosing efficient data types) [25].
2. Memory problems often arise out of computations that are not thought through. Most of the time, not the whole data needs to be in memory. Streaming the data or loading it progressively is a great option, if the algorithms permit this [10]. Programming languages often have built in lazy evaluation capabilities, such as Python's generators.
3. Another approach for storing and querying large datasets is the use of a relational database. The database can be queried using Short Query Language (SQL). Again,

this option has the premise of Machine Learning (ML) algorithms permitting iterative learning [10].

4. Finally, using a platform for ML workloads, such as SAP AI Core helps with handling large amounts of data. This approach requires aligning of the source code to fit the specifications and endpoints of the platform.

Messy Data Of course, data collected from operational sources is not perfect and ready to feed into an algorithm. Data cleaning is one of the main activities of data scientist's everyday workload. But what if the dataset is untidy to such a severe extent, that it is not able to be cleaned in a reasonable amount of time? Particular problems can be column shifts in tables, different units for values without naming the unit, or feature encodings without keys for decoding. Depending on the extent of the case, a different dataset should be evaluated. Also, the remark has to be made that this case is highly unlikely as the datasets have been used for other applications in the past.

Inefficient Calculation Calculations can quickly grow into inefficient and obfuscated code, taking hours or even days to complete. To mitigate this risk the following contingencies are proposed:

1. Different methods for calculating the same operation should be identified, evaluated and implemented.
2. Regular bench-marking of smaller chunks of the data helps to extrapolate processing times and decide for one solution.
3. Implementations in libraries should be, in general, favored over self-made implementation. Literature research in industry-specific blogs helps to find even more efficient implementations than those contained in popular libraries.
4. A sophisticated design of data structures is crucial in utilizing optimized code. Even the most elaborate way of calculating operations can turn into a resource-intensive task if the input data is structured poorly. Considering different data structures and selecting the most appropriate one for each specific task is crucial.

Results are of no Value The business objectives were determined in a prior section. But what is the procedure if the business objectives are not reached? Especially the criterion of giving useful insights is the crux. A simple laissez-faire attitude towards the

definition of "useful" is unsatisfactory, although creating the illusion of a positive outcome of the project. With not reaching the original goal of a research project, the work does not automatically become useless. Identifying problems and causes for the failure can facilitate other research.

5.2.4. Terminology

To summarize both relevant data mining and business terminology, a glossary was compiled. This is not yet completed, as the glossary of terms will be expanded while writing the thesis.

Clustering Algorithm A clustering algorithm is a sequence of instructions, which arranges a set of instances into groups, which contain items of high similarity to each other.

Document A document is a unit of data most typically containing natural language text.

Hyperparameter

Hyperscaler A hyperscaler is a company offering architecture to adapt to changing workloads in cloud computing, networking and internet services.

Natural Language Processing (NLP) NLP is often attributed to computer science, but after closer examination, NLP is a discipline comprised of linguistics, computer science, artificial intelligence an mathematics [12].

Overfitting

Pickle

Process

Script A script is a small piece of code contained in a single file, most commonly written in a dynamic high-level programming language.

SAP AI Foundation The SAP AI Foundation is a term describing SAP's offerings of the AI Launchpad and AI Core.

SAP AI Launchpad The SAP AI Launchpad is a tool for the operation of AI content inside an SAP system.

SAP AI Core SAP AI Core allows for training and serving AI scenarios [41].

Token

5.3. Data Mining Goals

The following data mining goals were identified during the phase of business understanding:

1. Identifying and applying appropriate methods for feature extracting tailored to this type of dataset.
2. Identifying and applying appropriate methods for clustering documents in this type of dataset.
3. Identifying and applying appropriate methods for topic modeling with this type of dataset.
4. Aggregating expenses by their clusters and visualizing the output.

The successful outcome is defined by reaching all named criteria. The achievement will be evaluated by the author and the supervisor, also people referenced in the inventory of resources will be considered to evaluate the outcome.

5.4. Project Plan

5.4.1. Project Plan

I don't think I will need this...

5.4.2. Initial assessment of tools and techniques

Programming Language A multitude of programming languages for statistical analyses and machine learning applications exist. The most popular and best supported are Python and R. Both languages are open source and targeted at data science tasks. Following criteria will be used to evaluate the best fit for this project:

1. Understanding, as the availability of learning resources and the ease of reading and writing source code.
2. Availability of resources such as libraries, packages and modules.
3. Compatibility with existing solutions, availability of scaling and deployment options.

Python is a general purpose programming language with a straightforward syntax. Python is regularly taught at schools and suited for beginners. R is a language built by statisticians. Therefore, R is suited primarily for data-science tasks. While a data analysis can be created easily, more complex functionality requires experience and is considered challenging [30]. For understanding, Python is rated as the more favorable choice.

For the availability of resources, R clearly scores. In general, Python has a large number of libraries available. Since python is not exclusively for data science, only a fraction of them are suited for statistical analyses. For R on the other hand, over 13.000 packages are available in the R archives [30].

Deploying a python solution is easily possible via APIs, web apps and containerization technologies. R is also displayable on the web using specific packages. Both R and Python can be deployed in docker containers [31] [29] paving the way to be deployed on all major hyperscalers. It can be stated that python has superior ability to be scaled and deployed as it is a general purpose programming language.

Programming Paradigm With the programming language decided, the basic programming paradigm has to be chosen. Python is an high-level object-oriented language, containing also functional aspects [13, chapter 1.3.2]. Python allows for programming using only scripts, but also supports and encourages the use of modules to separate files. For data scientist, the decision between modular code and code contained in a notebook presents itself in the beginning of every project. Python code in modules can be organized into several files calling each other when needed. This option requires careful documentation, as there is no graphical interface showing dependencies between the modules.

Python notebooks are structured to represent code, descriptions and the output. The most popular notebook format are Jupyter Notebooks, representing code and additional information as JavaScript Object Notation (JSON) data [19]. While notebooks allow to tell a story using visualizations and descriptions, they lack in ability to be easily deployed using common methods. Of course, the deployment of a Jupyter Notebook is not impossible, but disproportionately more difficult than deploying python modules.

Since not every part of a data-science project has to be deployed, a hybrid approach to Juypter Notebooks and modular coding makes sense. The data understanding part of the project will be completed using notebooks. The data preparation and the modeling will be completed in python modules to ensure being able to be deployed, if desired.

6. Data Understanding

6.1. Initial Data Collection

Data Requirements Planning The business objective is the development of a solution for the automatic aggregation of documents based on their topics. To achieve this goal, a sufficient number of documents are required. Also, the documents must contain enough text to be assigned a topic. After clustering the value of each expense in a cluster have to be summed, to gain the desired insight of spending in one category. So the value an expense amounts to has to be included in the data.

Selection Criteria An initial look a the data has shown that each invoice item can easily contain more than 100 pieces of information, which can be considered as attributed in the dataset. The part of the data that is of interest for the research is the description and payment data. Other data needs to be retained for a coherent and informative presentation of the results, such as location information and information on seller and buyer.

Insertion of Data The insertion of data is concerned with the theoretical utilization of the data and problems arising during this process. The encoding and grouping of free text items is referenced as one concern in the CRISP-DM user guide [11, p. 38]. In this research the focus is on free text items, therefore this step will be discussed in detail in the following chapter. Other concerns are missing attributes. The dataset has already undergone a surface-level evaluation, which concludes that all relevant attributes are contained.

6.2. Data Description

The data is available as 152,591 JSON files. Each file represents one invoice document.

Each invoice contains specific header data, the detailed list of the 172 header fields is contained in the appendix (A.2). The items listed in one invoice are line items. The information about line items is contained in 32 features, also detailed in the appendix (A.3).

All documents follow a specific schema (6.1). In the beginning, the metadata of the invoice is described, followed by an array of objects (l.7). Each item in the array contains two key value pairs. First, the label of the feature (e.g. "totalAmount"). Second, the value of the feature (e.g. "$7").

```
1  {
2     "$id": "https://example.com/arrays.schema.json",
3     "$schema": "https://json-schema.org/draft/2020-12/schema",
4     "description": "A representation of information extracted ↙
          ↳ from invoices.",
5     "type": "object",
6     "properties": {
7       "annotations": {
8         "type": "array",
9         "items": { "$ref": "#/$defs/annot" }
10      }
11    },
12    "$defs": {
13      "annot": {
14        "type": "object",
15        "required": [ "label", "text" ],
16        "properties": {
17          "label": {
18            "type": "string",
19            "description": "The identifier of the extracted ↙
                 ↳ information."
20          },
21          "text": {
22            "type": "string",
23            "description": "The value of the extracted ↙
                 ↳ information."
```

Listing 6.1: Shortened JSON Schema of one invoice document representation

6.3. Data Exploration

The exploration focuses on the descriptions and related metadata, as only this part of the data is used for modeling.

Descriptions The descriptions of invoice items vary greatly in their length. While the largest portion of descriptions is under 600 characters, there are outliers with over 1000 characters.

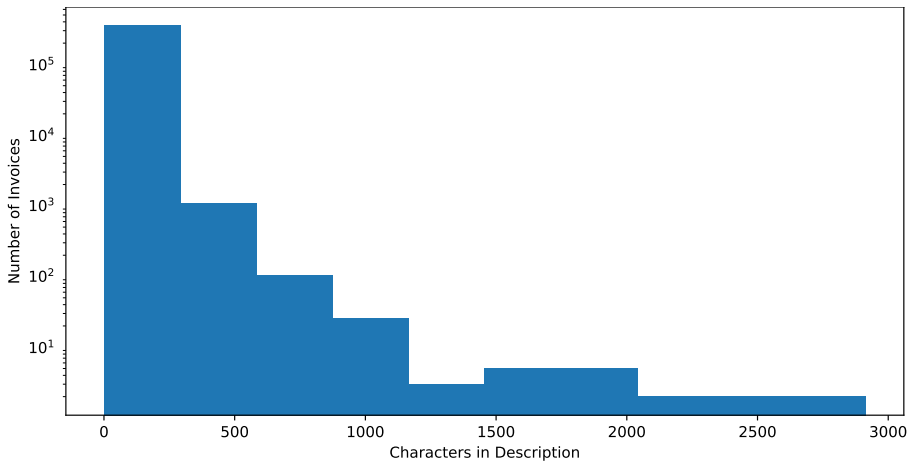

Figure 6.1.: Histogram of Description Length

The top ten most frequent descriptions are led by two values which will be removed during data cleaning. The first value ('nan') is short for 'not a number'. The second most frequent description is an empty value. Unfortunately, these descriptions amount for one fourth of all text elements. The next descriptions are more speaking, for example consulting fees, credit card transaction fees, and payments for room and board.

The frequency distribution of unique descriptions shows that the top third (36%) of the descriptions covers 80% of the invoices. It is notable, that the slope shows an edge at the point of the 24212the description. After this point, each following description only occurs once in the whole dataset. In turn, this means that only 20% of the descriptions occur more than once.

Description	Frequency	Percentage
nan	83716	23.54%
	6030	1.70%
ICO hour s consulting	4144	1.17%
ICO Hour s Premium Services	2269	0.64%
INTERNET MÓVEL	1959	0.55%
NRO TARJETA AMEX aut TRANSACTION FEE	1896	0.53%
Meals	1643	0.46%
BCD TRAVEL	1473	0.41%
BSN Fee Air Europe	1420	0.40%
ICO hour s development	1286	0.36%

Figure 6.2.: Top ten most frequent descriptions

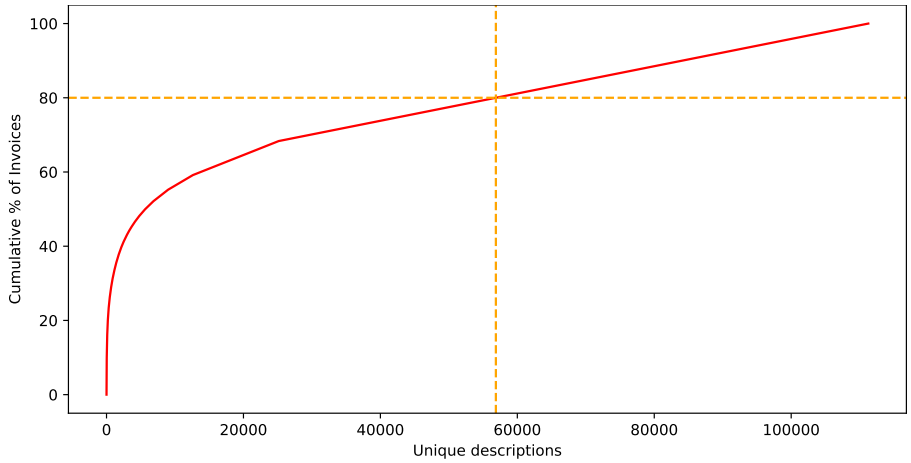

Figure 6.3.: Percentage of Invoices by cumulative unique Descriptions

Language Distribution The description of the invoices is related to the feature 'language', which describes the language of the invoice and its text fields. The most popular languages are English, Spanish, and German. It is noticeable, that there are over 700 languages or combinations of more than one language. 58878 invoices were not assigned a language, which is roughly one third of the total number.

From the chart (6.5) it can be inferred, that 80% of invoices are in the top 16 languages. The slow rise after the top 80 languages indicates, that there are a lot of languages

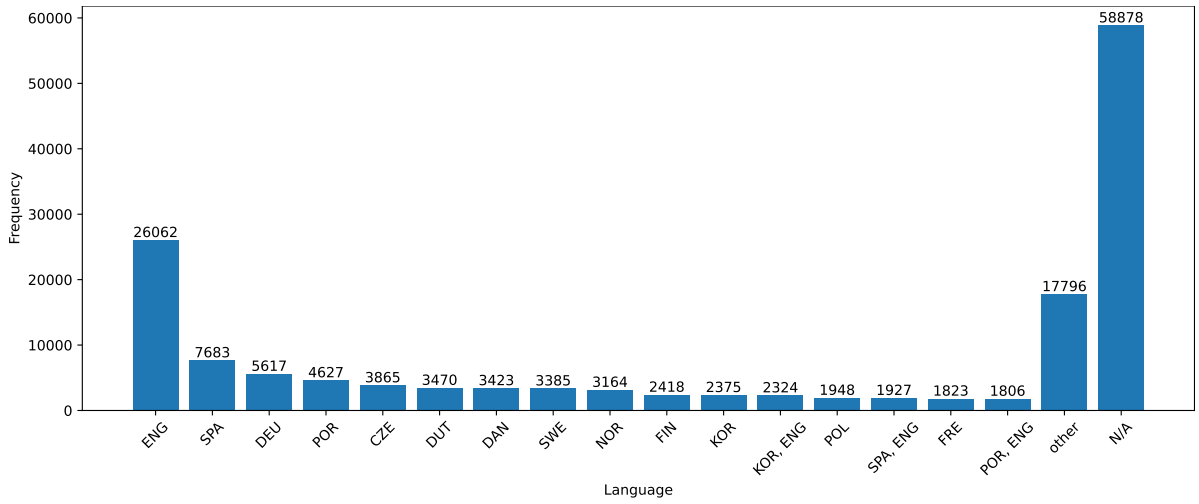

Figure 6.4.: Number of Invoices per Language

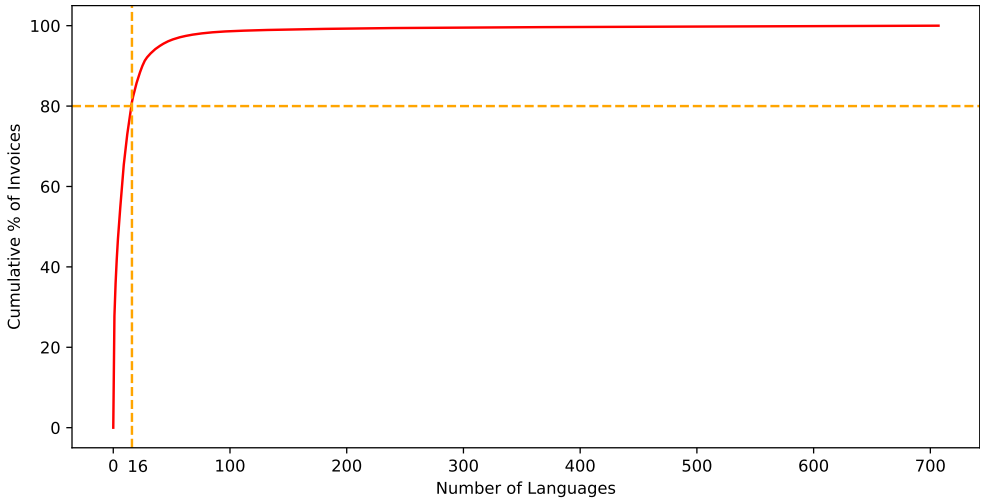

Figure 6.5.: Percentage of Invoices by cumulative Number of Languages

with a small number of invoices. This is also explained with a huge number of possible combinations of more than one language.

Invoice Item total Amount The invoice item total amount stands for the charge corresponding to one line in an invoice. Without very basic data cleaning, this column was not usable. The values were separated by a variety of different decimal delimiters and contained additional symbols. Also, the variation was initially very high due to some outliers in the magnitudes of 10^{217}. Therefore, the values were aligned to one common

decimal delimiter and stripped of the 1st and 99th percentile. This allows an initial insight into the distribution of the values.

Table 6.1.: Five-number summary for Item total Amounts

metric	**mean**	**std. dev.**	**min**	**25%**	**50%**	**75%**	**max**
value	4,749.69	17,687.85	1.64	45.84	276.20	1,520.00	197,721.71

The five-number summary shows that the values are skewed to the right in a considerable manner. The so-called tail of the price distribution reaches almost the 200,000 mark. For the further analysis, it will be interesting if very large charges belong into a similar group of spending.

6.4. Data Quality

In the analyses before several problems with the data could already be identified. Firstly, a lot of descriptions are empty or have the unusable value 'nan'. Secondly, the language distribution has shown that the spread is highly biased towards English. Both the unusable values and the uneven language distribution will be addressed in the data cleaning part of the thesis.

The quality of data for text mining purposes can be evaluated using four elementary dimensions [5, p. 1279] :

Completeness	The extent to which data are of sufficient breadth, depth, and scope for the task at hand
Correctness/free of error	The extent to which data is correct and reliable
Timeliness	The extent to which the age of the data is appropriate for the task at hand
Consistency	The extent to which data are always presented in the same format and are compatible with previous data

Completeness The invoices cover only a part of all billing documents received in the time frame. As a result, the analysis may not give an insight into the whole spending of one company but only into a fraction of it. Additionally, the data exploration has shown,

that about one fourth of the descriptions are either empty or contain a placeholder for a missing value ('nan'). During data cleaning, even more unusable values will turn up. Still, a pessimistic estimate of 100,000 unique descriptions is a large enough dataset for telling analyses.

Correctness The data was created with the help of an annotation service. The vendor specializes on Optical Character Recognition (OCR) and Natural Language Processing (NLP) to extract data from real documents, including invoices. Extractions results by this vendor have a guaranteed accuracy of 99% [40].

Timeliness The dataset consists of invoices in the time frame from 2015 to 2019, which makes it timely enough for the task.

Consistency Annotations supplied by the annotation service always follow the given structure detailed in example 6.1. This ensures consistency of data generated over a time span.

To sum the data quality report up, the data has flaws in the dimension of completeness, but these minor issues with data quality will likely not impact further data analyses.

7. Data Preparation

Data preparation is the process of transforming the acquired dataset into a dataset which can be fed into learning algorithms and is cleaned of impurities in such a way, that the learning results are likely satisfactory. Impurities to some extent were already identified in the prior chapter, additionally, shifted columns, encoding errors or different data formats have to be accounted for. Data preparation includes data wrangling, data cleaning, and feature extraction. What is left should be a uniform dataset, which is in a format that is understandable by the desired algorithms.

7.1. Data processing and data wrangling

7.1.1. Alternatives

The chosen dataset consists of files in JSON format. Several alternatives for storage and transforming the data exist.

Storage Location Within the constraints of the thesis, three options are available for data storage. Firstly, data can be stored locally on the available machine. The data is available without internet access, and the local machine has high read/write speeds. This option requires no additional learning, and is free of charge for the department. A multitude of libraries exist for accessing files on a local file system. A downside is the limited scalability in terms of storage capacity and read/write operations. Additionally, local storage makes the data only available to this specific machine.

Storing data on a cloud file sharing system is easy to operate and free of usage-bound charges. The storage space is virtually unlimited. Accessing files on sharing platforms is feasible but tedious. The access can be shared within the company context, but everyone is subject to the limited accessibility. Using a files storage service could be of use for transferring data without a physical connection. Still, this is the only recommended use case in this context.

The third option is storing the data using a specialized storage service, such as AWS S3. While the learning overhead is higher in the beginning, the scalability is a convincing argument. Billing is according to usage, but adequate because of the high connectivity with other cloud-based ML service offerings inside and outside of SAP.

Table 7.1.: Comparison of available Storage Options

	Local	**Cloud Filesharing**	**Specialized Storage Services**
Example	2.6GHz 512GB SSD	Microsoft OneDrive	AWS S3
Ease of use	high	high	higher learning overhead
Cost	free of charge	free of charge	billing according to used storage space
Scalability	limited	unlimited	unlimited
Data access	local	limited remote access capacity	local, remote, high connectivity to ML service offerings

Data Format The data is still available only in the from JSON documents. Python allows for easy parsing of these objects, but this option is highly inefficient compared to native Python data structures. The equivalent to JSON in Python are combinations between lists and dictionaries. Dictionaries are an implementation of the `Map` data structure.

This option stands in contrast to a data format commonly used in data science tasks: the `DataFrame`. Both formats, the dictionaries and DataFrames have specific advantages and disadvantages.

The `DataFrame` is structured like a table, consisting of labeled rows and columns. This data structure is implemented in the library `pandas`. Different data formats such as strings, floats or boolean values can be stored in the table. Optimized methods for calculating row-wise and column-wise operations are supplied in the library. This makes the `DataFrame` one of the most popular tools for data scientists.

`DataFrames` being suited for various purposes, makes them inflated and slower than low-level data structures. The indexing operation, for example, has to address a multitude of cases. This makes it slower than a simple dictionary access. Appendix A.4 details the comparison in a benchmark.

Surprisingly, storing JSON data in a `DataFrame` is less memory-intensive than storing the same data in dictionaries and lists [48]. This means `DataFrames` are favorable for storage and calculations with large data quantities.

Considering all factors, the best choice is transforming the dataset into a `DataFrame`. Still, the advantages of dictionaries and lists in basic use-cases will be kept in mind for later.

7.1.2. Theoretical Implementation

For the data wrangling the invoices have to be read into memory and then processed into a reusable structure. All invoices will be combined into `DataFrames`, which then are persisted for later use.

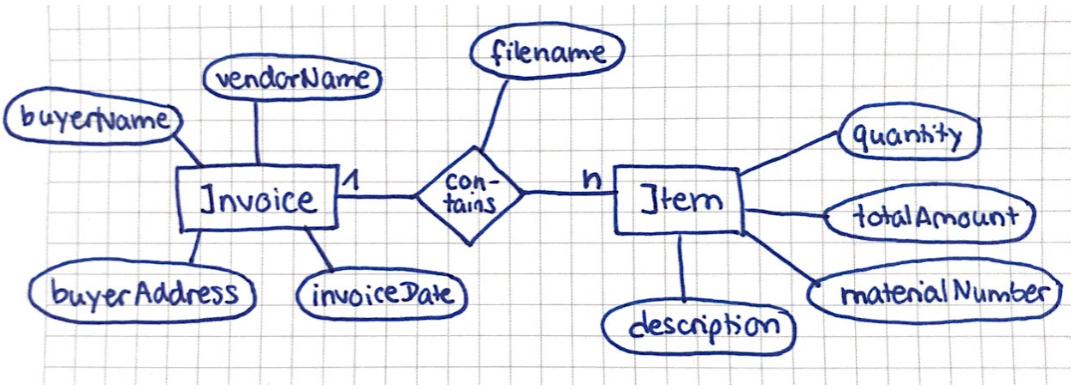

Figure 7.1.: Entity Relationship Diagram for Invoice Documents

The documents can be separated into two fundamental entities: the invoice and the line items. The entity relationship diagram (7.1) shows how invoices and line items are related. The invoice contains at least one line item, and the information in the invoice is relevant to all of its line items. This includes information about the vendor, and the invoice date. The line items of an invoice contain specific information on the products,

such as the material number and the description. An invoice and its line items can be matched through an unique identifier, which is also the filename.

Figure 7.2.: Theoretical Implementation of the Data Wrangling

Now, looking at the technical side of the file composition. The data for an invoice and its line items is stored in an array of objects. The array "annotations" contains objects, each one representing information such as the invoice date or the unit price. A schematic description of how the invoice data should be represented after extraction is shown in figure 7.2: One invoice is modeled as one row in the table. Different attributes are represented as columns. Information about line items have labels containing the prefix "lineItem". One invoice containing several items is represented by duplicate values of one label (in the example **??** the label "lineItem.description.value"). Each new label "lineItem.description.value" denotes a new line item. Here, the order of the labels has to be retained during data wrangling to ensure not mixing up information about different invoice items. Each new line item should be a new row in the table of line items.

The goal of the data wrangling process is to create two `DataFrames`, one for the invoices and one for the line items. Additionally, another data structure will be created to optimize recurring and complex operations.

Data exploration shows, there are only 79.741 unique descriptions for the listed items. By saving only the unique values, the required space is reduced to less than one fourth compared to before. Additionally, this step is required by most machine learning models, as duplicate input values can skew the outcome. The model is chosen later, so this processing step leaves the model selection more open to different kinds of learning algorithms.

After the deletion of duplicate descriptions, the relationship between one line item and its description is not one-to-one, but rather one description belonging to several different line items. Model outputs will be a classification of one description, so to restore the relationship between the classification result and one line item, a mapping is constructed. This dictionary maps one description to all line items (ID) which have this description value.

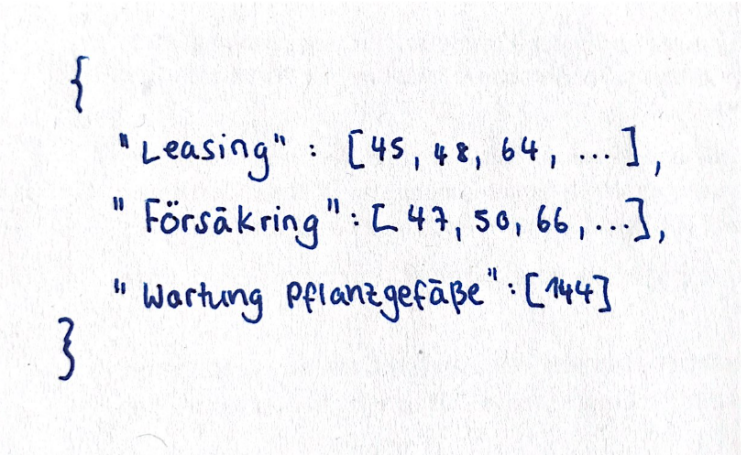

Figure 7.3.: Schematic depiction of the Mapping of Description to their Line Item IDs

After the construction of both tables and the mapping, these artifacts should be serialized. This way, they can easily be loaded into memory without the effort to reconstruct them from scratch.

7.1.3. Practical Implementation

The files were processed using a Python script (A.2). The information on invoices is stored in a tabular data structure, a `pandas DataFrame`. Worth mentioning is that some invoices contain more information than others. In this case, invoices are still appended into one table, but fields for non-existent values are left empty. In figure 7.4, the first invoice does not have a total amount included, but since the second one does, this column is created. The filename of one invoice contains the unique identifier for one invoice.

filename	vendorName.value	invoiceDate.value	totalAmount.value
73d8feb2-b01d-11ec-b909-0242ac120002.json	Example Telephone Company	2020-03-12	
942d7318-b01e-11ec-b909-0242ac120002.json	Example IT Vendor	2020-02-31	24.48

Figure 7.4.: Exemplary depiction for processed invoices in a DataFrame

Similarly, information on the invoice items is retrieved from the documents and stored in a `pandas DataFrame`. Every line item has an unique id and can also be linked to the respective invoice through the filename.

	filename	lineItem.totalAmount.value	lineItem.description.value
0	73d8feb2-b01d-11ec-b909-0242ac120002.json	250.00	Cell phone bill may
1	73d8feb2-b01d-11ec-b909-0242ac120002.json	256.00	Cell phone bill june
2	73d8feb2-b01d-11ec-b909-0242ac120002.json	249.00	Cell phone bill july
3	73d8feb2-b01d-11ec-b909-0242ac120002.json	280.00	Cell phone bill august
4	73d8feb2-b01d-11ec-b909-0242ac120002.json	300.00	Cell phone bill september
5	942d7318-b01e-11ec-b909-0242ac120002.json	12.99	Mouse
6	942d7318-b01e-11ec-b909-0242ac120002.json	4.99	HDMI Adapter
7	942d7318-b01e-11ec-b909-0242ac120002.json	6.50	Shipping Fee

Figure 7.5.: Exemplary depiction for processed invoice items in a DataFrame

The resulting two `DataFrames` are serialized with the python standard library `pickle`. The data can be efficiently loaded into memory and re-serialized again with this library.

This processing step allowed for storing the initial 5.12GB of data in a more usable format and takes up only a total of 393MB.

The process of reading files from the disk is not inherently an expensive one, but in the realms of many thousand documents, processing times soon reach several days. Observing the execution of the python code showed a peculiarity: While the utilization of the used processor was consequently at the maximum (in figure 7.6 at 97.1%), only one process is executed, leaving the total CPU usage at only 20%.

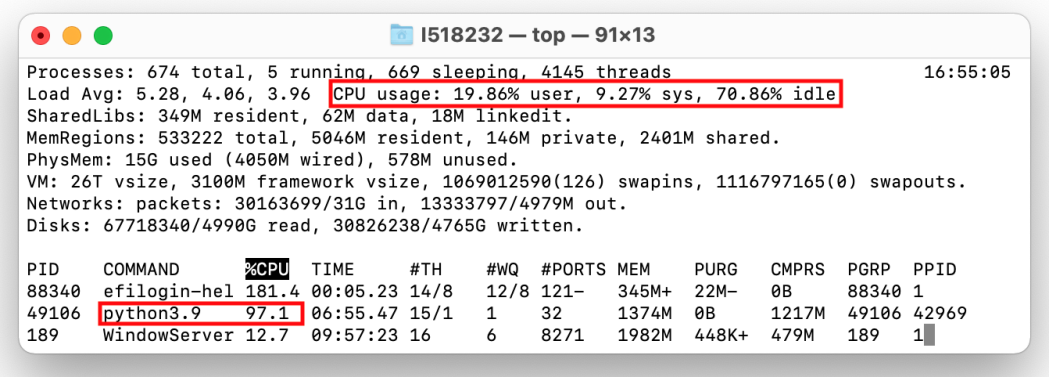

Figure 7.6.: CPU Usage during single-process Python Execution

One question that may now arise is: Why doesn't python split up the workload and employ the full capacity of the machine? This can be explained with the design of the Python interpreter. The Global Interpreter Lock (GIL) controls access to the Python Virtual Machine executing the code. This lock only allows exactly one thread to run at a time [13]. This of course is not favorable, as valuable CPU capacity is idle. Fortunately, the GIL behaves in a special way regarding C routines: the lock is released before executing C code [13]. I/O operations in Python, such as opening files, utilize C code. This allows to bypass the, in this case, inconvenient locking mechanism.

Different python libraries exploit this specialty and allow to spawn a pool of different processes, which then execute calls asynchronously. One example is the `ProcessPoolExecutor`. A notable restriction is that only picklable objects can be submitted for multiprocessing. This concerns both function and its parameters. While this is not a problem in this task, this restriction will become important later on in the section about feature extraction.

```
1 with concurrent.futures.ProcessPoolExecutor() as executor:
2     results = process_map(r.read_invoice, filenames, ↙
      ↳ chunksize=2000)
```

Listing 7.1: Spawning a Process Pool in Python

The `ProcessPoolExecutor` (s. listing 7.4) can distribute a list of input values for functions (`filenames`) onto a function (`r.read_invoice`). Additionally, the `chunksize` as parameter specifies how large the bundles of input values for each process are. Distributing the workload onto all CPU cores, the processing time was reduced from 27 hours to under 4 minutes.

7.2. Data cleaning

Data cleaning concerns removing all impurities from the data. This includes deleting duplicates, using an uniform data type, and removing unnecessary information. Just the descriptions of all the line items will be fed into models, therefore only those have to be cleaned. The Python library Natural Language Toolkit (NLTK) provides useful methods for cleaning textual data.

```
1 from nltk.tokenize import word_tokenize
2
3 def clean(text):
4     tokens = word_tokenize(str(text))
5
6     alphabetic_text = [t for t in tokens if t.isalpha()]
7     return alphabetic_text
```

Listing 7.2: Python Script for extracting JSON Data into a DataFrame

In the code example 7.2, the text is firstly separated into tokens. Then each token is examined whether it contains non-alphabetic characters. Only alphabetic tokens are retained. Again, this process is highly computationally intensive due to the size of the data. Therefore, this task was completed using the same procedure for multiprocessing as detailed in the prior section.

```
1 $ python3 -c 'import cleaner; print(cleaner.clean("500grams of ↙
    ↳ special baking flour type 504"))' \\
2 >>>  ['of', 'special', 'baking', 'flour', 'type']
```

Listing 7.3: A Description before and after Data Cleaning

The code example 7.3 shows the input and output of this procedure. A token is completely discarded if it contains non-alphabetical characters, as these tokens are more than often not an accurate descriptor of the text's meaning.

7.3. Feature Extraction and Feature Engineering

The majority of popular ML algorithms require the input of scalar, vector or matrix data. Several methods for representing text as mathematical object will be discussed in the following.

7.3.1. Alternatives

Firstly, the distributional representations will be explained and evaluated.

Term-Document Incidence Matrix

document	content
1	car repair
2	flight ticket processing and ticket reservation
3	parking ticket

Figure 7.7.: Example Corpus

The three documents in figure 7.9 will be considered to explain the workings of the one-hot encoding. A corpus is transformed into a term-document incidence matrix representation in two steps.

Firstly, the vocabulary is determined. It is a collection of all words occurring in the corpus. Every word is contained exactly once, regardless of the actual number of occurrences. The vector representation of one document is of the same length as the vocabulary. One document being represented by one vector, a corpus of several documents can be

	and	car	flight	parking	processing	repair	reservation	ticket
car repair	0	1	0	0	0	1	0	0
flight ticket processing and ticket reservation	1	0	1	0	1	0	1	1
parking ticket	0	0	0	1	0	0	0	1

Figure 7.8.: Term-Document Incidence Matrix

represented as a matrix. The resulting matrix M has the size $|D| * |V|$, with $|D|$ being the number of documents in the corpus, and $|V|$ being the size of the vocabulary.

Secondly, for each combination of one document d_i and one word v_j in the vocabulary, it is determined whether the word occurs in the document. This information is encoded binary with a "1" at m_{ij} for occurrence and a "0" for no occurrence. The resulting vectors for the three documents are:

$$d_1 = [0, 1, 0, 0, 0, 1, 0, 0]$$

$$d_2 = [1, 0, 1, 0, 1, 0, 1, 1]$$

$$d_3 = [0, 0, 0, 1, 0, 0, 0, 1]$$

The drawback of this method is that no consideration is paid to words being repeated in one document. Additionally, no semantic relationship between words or documents can be inferred. Further, it can be stated, that the Bag of Words (BoW) representation fails to capture the meaning of synonyms. This becomes obvious with an example: document d_1 and d_3 would be considered to belong into the topic of transportation or automotive. The BoW representation suggests topical proximity between document d_2 and d_3 , through the shared word "ticket". "Ticket" here is used in both the meaning of an entrance pass (d_2) and in the meaning of a note for a traffic offense (d_3). Further, this feature extraction method strongly suffers from high dimensionality [**practicalNLP**], since the matrix dimensions depend on the size of the vocabulary.

Bag of Words Model

With the BoW model, the only difference to a term-document incidence matrix is that the occurrences of a word are counted, instead of a binary encoding.

	and	car	flight	parking	processing	repair	reservation	ticket
car repair	0	1	0	0	0	1	0	0
flight ticket processing and ticket reservation	1	0	1	0	1	0	1	2
parking ticket	0	0	0	1	0	0	0	1

Figure 7.9.: Bag of Words Representation of Documents

The result of vectorization are three vectors:

$$d_1 = [0, 1, 0, 0, 0, 1, 0, 0]$$

$$d_2 = [1, 0, 1, 0, 1, 0, 1, 2]$$

$$d_3 = [0, 0, 0, 1, 0, 0, 0, 1]$$

Tf-Idf

The TF-IDF model aims to capture more meaning from the corpus by considering the composition of the whole corpus for the calculation of individual document vectors. TF-IDF makes two assumptions about natural language:

1. A word t_i which occurs very frequently in one document is considered to describe a text very well (term frequency).
2. A word t_i which occurs in a large number of documents does not describe one document well (inverse document frequency).

Term Frequency The occurrences of one word in one document is denoted as $\#(t_i)$. One additional consideration needs to be made regarding the document length. In a document of length $|d_2| = 6$ and a document of length $|d_3| = 2$, the word $t_i$occurring once should be considered more important to d_3, since it accounts for a larger share of the text. The measure resulting from this idea is the term frequency:

$$TF(t_i, d_j) = \frac{\#(t_i)}{|d_j|} = \frac{occurences\ of\ word\ t_i\ in\ document\ \ d_j}{legth\ of\ d_j}$$

Inverse Document Frequency Words occurring in many documents often are articles or pronouns (stop words) which do not provide value when inspecting the content of a text. The inverse document frequency is a measure accounting for this fact. The inverse document frequency of a word is the proportion between the number of documents in the corpus and the number of documents containing the word. The logarithm is applied, as the importance of a word does not increase proportionally to the number of occurrences.

$$IDF(t_i, d_j) = \log \frac{|D|}{\#(d_{t_i})} = \log \frac{number\ of\ documents\ in\ corpus\ D}{number\ of\ documents\ containing\ word\ t_i}$$

Combining both assumptions, the TF-IDF measure is created:

$$TFIDF(t_i, d_j) \ = \ TF(t_i, d_j) * TF(t_i, d_j)$$

For the corpus displayed in the previous section the document vectors calculated with the TF-IDF measure are:

$d_1 = \begin{bmatrix} 0 & 0.707107 & 0 & 0 & 0 & 0.707107 & 0 & 0 \end{bmatrix} d_2 = \begin{bmatrix} 0.39798 & 0 & 0.39798 & 0 & 0.397980 & 0.397980 & .605349 \end{bmatrix} d_3 = \begin{bmatrix} 0 & 0 & 0 & 0.7$

The TF-IDF measure corrects some of the pitfalls of the BoW model. It certainly is less vulnerable to skewing by stop words, as words are ranked by importance to each document and the all over corpus.

The above mentioned methods are distributional representations [**practicalNLP**]. They all suffer from similar problems. The vectors contain one element for each word in the vocabulary, resulting in vectors which are high-dimensional and very sparse. Also, TF-IDF fails to represent the topical relationship between d_1 and d_3. Finally, these methods can not handle an inference for words which are not in their vocabulary [**practicalNLP**].

Following methods are distributed representations. They are designed to alleviate the drawbacks of distributional word representations.

Word Embedding BERT

The most desirable vector representation consists of dense low-dimensional vectors which are close to each other if the words are considered similar. The lack of understanding of related words, and the problem of high-dimensionality is corrected with the third presented option: word embeddings.

An embedding is a translation of a word into a high-quality vector. The model does so, as it has been trained on a large set of natural language data. The first model for continuous vector representations was word2vec [26]. Being published in 2013, the model has been already replaced by new state-of-the-art models such as BERT. This new model builds on the premised of word2vec, but was able to obtain astonishing results at tasks such as answering questions using the Stanford Question Answering Dataset [14].

The Bidirectional Encoder Representations from Transformers (BERT) model can represent unlabeled text by learning on large textual corpora. Its name is assembled from its different components:

B	Bidirectional	The model considers both left context (previous words) and the right context (following words) during learning.
E	Encoder	The model consists of several encoder layers.
R	Representations from	Sequence representations are a projection of a sequence onto a vector which is able to reflect the sequence's meaning.
T	Transformers	BERT belongs into the family of transformer models and utilizes parts of the transformer architecture.

The model is trained on two tasks. Firstly, it is trained on word prediction (**??**). BERT had to predict one missing word in a text unit. The objective of course is to get the right word. The loss in this task is measuring if and how wrong a prediction is. The second task is to predict the next sentence after a text section. The objective is to predict the correct words and their order. The model is trained by adjusting internal weights in such a way, that both losses are minimized. In turn this means, the model can predict missing words and next sentences with minimized error.

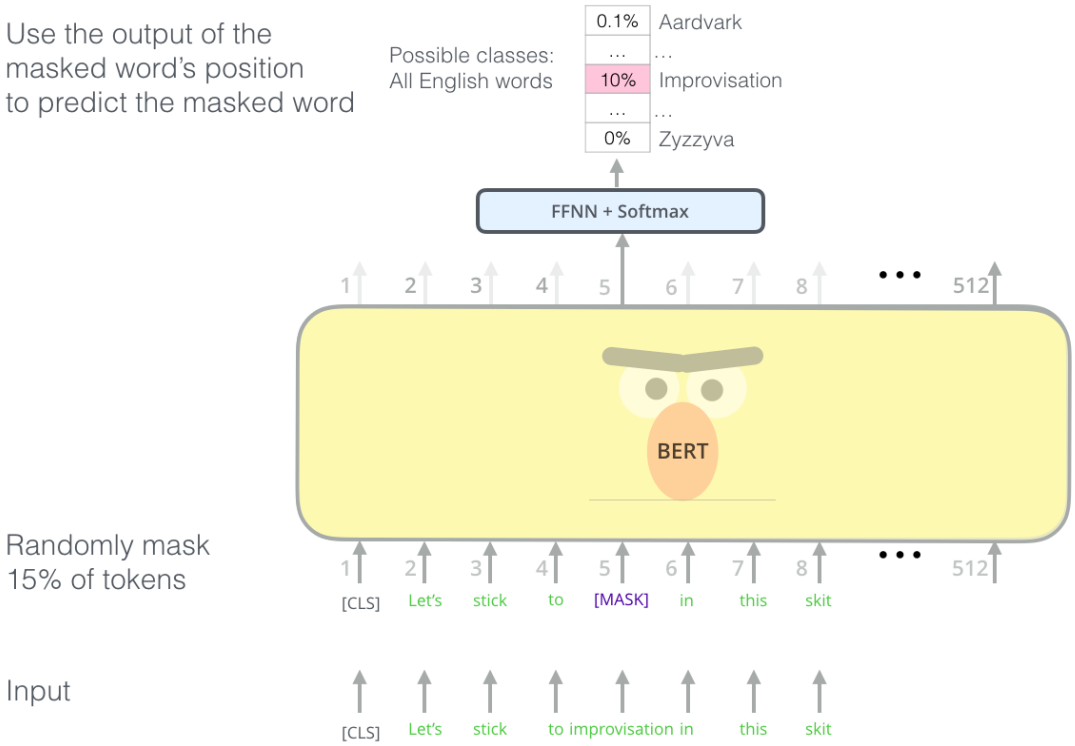

Figure 7.10.: Training BERT with the MLM task [3]

BERT is trained with millions of wikipedia articles, meaning the inputs is a large text database. The text segment is split into tokens, which are then encoded as a vector of length 768 [1]. Next follows the positional encoding. When applying the transformer encoder to a sequence of words, the output should be dependent of the sequence of the input words. E.g. the sentences "Bob is taller than Alice" and "Alice is taller than Bob" should be encoded as two different vectors. This can be realized with encoding the position of each token.

The token embedding and its positional encoding are now added, and fed into the encoder [4]. Inside the encoder, four elements process the input sequentially:

1. The self-attention mechanism allows for the model to incorporate the context of a token into its encoding. The mechanism does so by calculating the importance of inputs with themselves ("self") and returns which tokens are important to each other's meaning ("attention").
2. To reduce the computational requirements for training a deep neural network, the layers of neurons in the networks are normalized with respect to their activation.

This reduced the training time while not impacting the result, which is only dependent on the learning algorithm [6].

3. The Feed-Forward Neural Network (FFNN) takes the results of the layer normalization and the residual connection as input. When the whole transformer is trained on a specific task, the parameters (weights) of this model are also trained. Therefore, the FFNN is one gear in the whole process of reaching the transformer's task.
4. Finally, the results are normalized again.

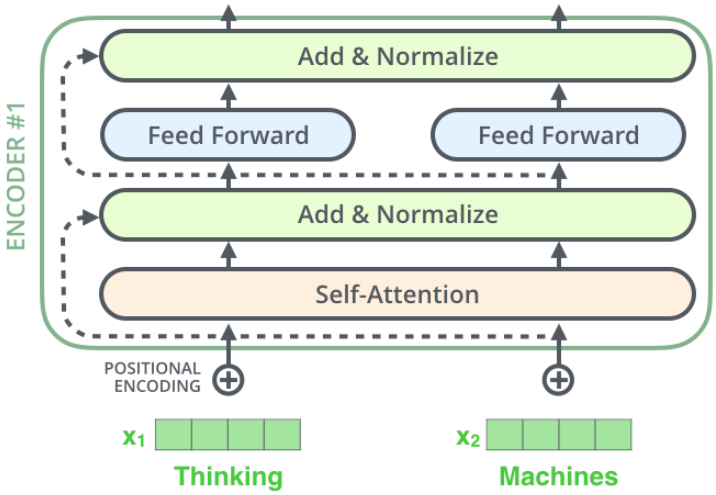

Figure 7.11.: The Transformer Encoder Architecture [4]

Evaluating all different models for word vectorization, BERT clearly outperforms the distribustional representations:

- Word embeddings produce dense and lower-dimensional vectors than distributional representations.
- With word embeddings, the resulting vectors perform a lot better on common NLP tasks, meaning the resulting vectors are of higher quality than with distributional representations.
- BERT can process out-of-vocabulary words, this is a big advantage to previous embeddings.

The advantage of BERT over the other presented options is obvious. But, now there are two unconsidered restrictions: First, BERT only encodes words. Second, BERT is only in one language.

Luckily, researchers around BERT encountered similar tasks and developed a new BERT model to solve this. First, the Sentence-BERT embeddings are generated in a way that requires only minimal additions to the BERT architecture [33]. Sentence-BERT is trained with a dataset of sentences, labeled with their relationship (contradiction, entailment, neutral) [33]. Two sentences each pass through a BERT model. A pooling layer for each of the two calculates a weighted sum of all words in the sentence to create the embedding for the whole sentences. With the labels and the comparison of two sentence embeddings, the pooling layer is trained. The weights in both pooling layers are adjusted over time, creating a so-called Siamese Network.

Secondly, the embeddings need to be able to understand different languages. A multilingual implementation of BERT can solve this. A multilingual embedding should align vectors in different languages in such a way, that translated sentences are very close in the vector space. A multilingual Sentence-BERT model trains on a dataset with sentences and their translation [32]. The weights in the BERT model are adjusted by completing the task of assigning the same sentence in different languages a similar vector [32].

Both the multilingual and sentence embeddings require very little additional complexity compared to the underlying BERT model. But, these additions help to accurately represent the content in the dataset.

7.3.2. Theoretical Implementation

After explaining and evaluating the alternatives, the most favorable one, the word embeddings using BERT will be put into practice. In theory, a BERT model would have to be trained using a large text dataset. This took the developers of BERT at Google 4 whole days, and required 4 Tensor Processing Unit (TPU)s [27]. Fortunately, the researchers published this trained model, making it free to use for the public [14].

From the available pre-trained models, the `distiluse-base-multilingual-cased` model fulfills all the requirements (sentence embeddings, multilingual vector alignment). The prefix 'distiluse' stands for distilled universal sentence encoder, meaning the sentence encoder is compressed in a model with a more portable size. The suffix 'cased' means that words are handled case-sensitive.

Since the description data has already been transformed into a usable format and cleaned, these descriptions can now easily be fed into the `distiluse-base-multilingual-cased` model.

7.3.3. Practical Implementation

The model can be laded using the library `sentence_transformers`. The library can create a model from the name of the transformer. Embeddings can be generated very easily, and are returned as a 2D-array. This inference takes about 30ms. For all 79.741 documents, this means an total time of about 40 minutes.

```
1
2 from sentence_transformers import SentenceTransformer
3 model = SentenceTransformer(
4     'sentence-transformers/distiluse-base-multilingual-cased-v2'
5     )
6 sentence = ["Service Charge for Flight FRA-LAX"]
7 embedding = model.encode(sentence)
8 print(embedding)
9 >>>    [[-0.02105838 -0.01305696 -0.0403975 ... 0.08315323 ↙
      ↳ 0.06774105 0.03354893]
```

Listing 7.4: Generating an Embedding with BERT

Since data cleaning and data wrangling could be sped up significantly with multiprocessing, it stands to reason that this might also work for the inference on BERT. Batching and parallel processing only provides a speed-up if the model can utilize a strong Graphics Processing Unit (GPU) [42]. This is not the case, so we settle with a one-time processing taking 40 minutes.

During processing, the hardware was overheating and shutting down. To prevent losing progress, batches of 1000 documents were processed at a time, serialized and written on the hard disk. Upon completion, the files are loaded into memory again and unified into one collection of 20MB in vectors.

These vectors are now ready for clustering.

8. Modeling

The objective of this thesis is the grouping of expenses with the methods of NLP. Grouping, or clustering, is the practise of sorting data points into groups in such a way that the similarity inside a group (intra-cluster similarity) is high while the similarity between clusters (inter-cluster similarity) is low. The defintion of similarity as well as different clustering algorithms are presented and contrasted in the following sections.

8.1. Alternatives

8.1.1. Similarity and distance measures

A distance measure is a quantification of how near objects in space are. Distance measures can be defined for spaces of arbitrary numbers of dimensions. In the examples, four words (man, woman, king, queen) are compared in their semantic similarity. The word vectors were generated with a word embedding, but the focus is on comparing the distance measures available for clustering later on.

Figure 8.1.: Dot Product as Similarity Measure

Dot Product The dot product is an operation transforming a vector into a scalar through multiplying the vectors element-wise. This operation is easily and efficiently calculated, but has some pitfalls. While every word in the example (8.1) ranks very high in similarity to itself, the distance between the word and itself is not always the same for every word. This happens because the dot product is not agnostic of a vector's magnitude. A vector's product with itself is the square of its magnitude. This results in the similarity of man-man to be lower than the similarity of queen-queen. This fact makes the dot product impractical for accurately representing the relationships of words.

$$\text{dot product} \coloneqq \mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i$$

Euclidean Distance For low-dimensional applications, this distance works well and shows great results. Euclidean distance can be calculated in a highly efficient manner, even for n dimensions. This suggests, that euclidean distance is particularly suitable for high-dimensional data. Unfortunately, euclidean distance falls victim to the curse of dimensionality. In [8, p. 1] it is proven that with increasing dimensions, the distance between data points approaches an uniform value for all data points. This effect could be demonstrated for spaces with as little as ten dimensions. Therefore, it can be said that euclidean distance is not suitable for high-dimensional data. With vectors in NLP ranging from 100 to 800 dimensions with embeddings and hundreds of thousands with TF-IDF, this distance measure is not suitable.

$$\text{euclidean distance} \coloneqq \sum_{i=1}^{n} (A_i - B_i)^2$$

Cosine Similarity The name of this measure already suggests that not distance but similarity of objects is measured. Instead, it quantifies how close two vectors are. Mathematically, the cosine distance is the cosine of the angle between two vectors. A wide angle means a high distance between two vectors, or in this case two words. Narrow angles stand for a low distance and words being closely related. In example **??**, the word vectors for woman and king have a higher angle than those of woman and queen. This means the words woman and queen are more similar thank woman and king.

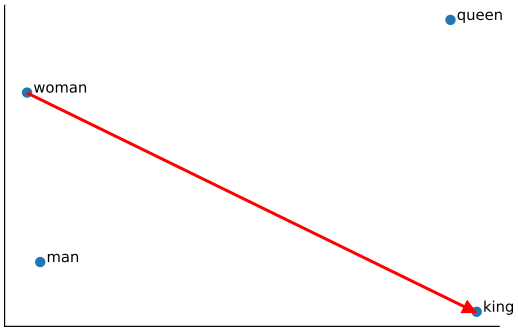

Figure 8.2.: Euclidean Distance as Measure

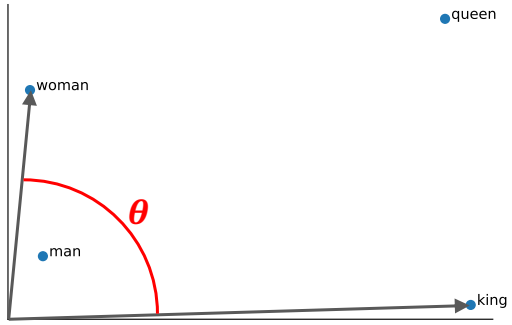

Figure 8.3.: Cosine Similarity as Measure

$$\text{cosine similarity} := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The difference between cosine distance and euclidean distance is clear: cosine measures the angle, and euclidean the vector length. But which one performs better in high dimensions? Cosine distance is perceived as more suitable for high dimensional data [2, p. c.2.1.2.1], and often suggested for NLP tasks.

But there is one fact left out in this suggestion: On normalized data (all vectors are unit vectors), cosine distance and euclidean distance are linearly connected.

$$\cos(\theta_{AB}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

$$\text{for normalized vectors with } \|\mathbf{A}\| = 1 \text{ and } \|\mathbf{B}\| = 1:$$
$$\cos(\theta_{AB}) = \frac{\mathbf{A} \cdot \mathbf{B}}{1 * 1} = \mathbf{A} \cdot \mathbf{B}$$

$$
\begin{aligned}
d(A,B) &:= \sum_{i=1}^{n} (A_i - B_i)^2 \\
&= \sum_{i=1}^{n} A_i^2 - 2A_iB_i + B_i^2 \\
&= \sum_{i=1}^{n} A_i^2 + \sum_{i=1}^{n} -2A_iB_i + \sum_{i=1}^{n} B_i^2 \\
&= \|\mathbf{A}\| + \sum_{i=1}^{n} -2A_iB_i + \|\mathbf{B}\| \\
&= \|\mathbf{A}\| - 2\sum_{i=1}^{n} A_iB_i + \|\mathbf{B}\| \\
&= \|\mathbf{A}\| - 2(\mathbf{A}\cdot\mathbf{B}) + \|\mathbf{B}\|
\end{aligned}
$$

for normalized vectors with $\|\mathbf{A}\| = 1$ and $\|\mathbf{B}\| = 1$ and $\mathbf{A}\cdot\mathbf{B} = \cos(\theta_{AB})$:

$$
\begin{aligned}
d(A,B) &= 1 - 2\cos(\theta_{AB}) + 1 = 2 - 2(\cos(\theta_{AB})) \\
&= 2 - 2(\cos(\theta_{AB})) \sim \cos(\theta_{AB})
\end{aligned}
$$

Now, with the newly gained insight into the distance relationships of normalized vectors of $\cos(\theta_{AB}) = \mathbf{A}\cdot\mathbf{B}$ and $d(A,B) = 2 - 2(\cos(\theta_{AB}))$
, it can be stated that the cosine distance and euclidean distance of two normalized vectors are linearly related.

Cosine and euclidean distance being connected linearly for normalized data has not made the decision easier. But there is an additional factor:

One additional constraint Cosine distance returns impractical results Cosine distance only looks at dimensions where both values are non-zero. All other dimensions turn 0 in the dot product. Cosine is better for sparse data, euclidean is better for dense data, since The distance approaches a uniform value for euclidean but actually But for cosine its almost the same this is because they are proportional (insert proof) Additionally, k-Means cannot use cosine because it violates triangle equality and needs averages

Explain how we should not use t-SNE before clustering

8.1.2. Clustering Algorithms

K-Means The k-Means algorithm generates k groups of data by iteratively adapting clusters and their centers. The algorithm is mainly focused on performance, hence the design is kept lean by omitting logic for determining the number of clusters [2, pp. c. 6.2]. Following pseudo code illustrates the workings of the algorithm.

```
1 randomly choose k_centers
2 while (iterations < max_iterations):
3     assign each point to nearest center in k_centers
4     mean of each cluster's elements are k_centers_new
5     if k_centers == k_centers_new:
6         break
```

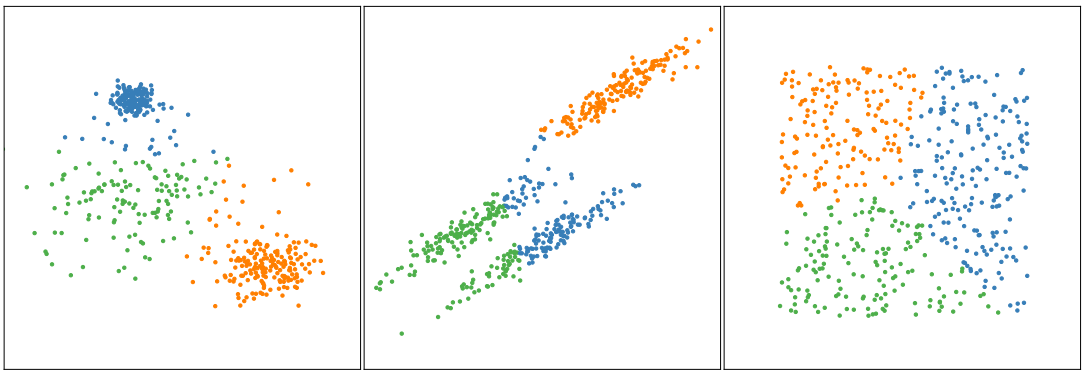

Figure 8.4.: How MiniBatchKmeans recognizes Clusters [28]

The k-Means algorithm performs reasonably well in identifying clusters in data such as in figure 8.4. The left example seems unspectacular, but in the center example, it becomes obvious that k-Means is prone to overfit on noisy data. This can be partially blamed on the focus on centers, and not density. The focus on centers is also obvious in the rightmost example. While the data is quite evenly distributed over the area, the algorithm still groups the data.

Implementations such as scikit-learn (sklearn)'s `MiniBatchKMeans` [28] speed up the computation time by processing subsets of the data in a parallel way. While only achieving almost perfect results [43], the batched k-Means algorithm is of special appeal when handling large data sets.

This algorithm has the advantage of high performance, and performs well on large data sets. Unfortunately, the quality of the results depend highly on choosing the correct parameters. Also, k-Means is not robust against outliers, and results are not reproducible since the first cluster centers are chosen randomly [2, p. c.6.2].

Density-based Spatial Clustering of Applications with Noise The DBSCAN clustering algorithm was developed to solve the problem of needing domain knowledge while tuning cluster models [15]. The algorithm uses an intuitive approach for identifying which points belong into a cluster, and which are outliers.

For two points to be considered neighboring (density-reachable, [15]), they have to be within a certain distance to each other: `eps`. Neighboring points are categorized into one cluster, expanding the reach of this cluster over all values within the distance `eps`. But, there is a limitation to the points being able to expand a cluster. If the cluster were to expand without limitations apart from a maximum distance, the clusters would be able to "spread" from one cluster to another if connected by one data point. To prevent this, only center points can expand the cluster. A center point has a minimum number of neighboring points (`min_samples`). This helps to create dense clusters and prevents the undesirable spreading.

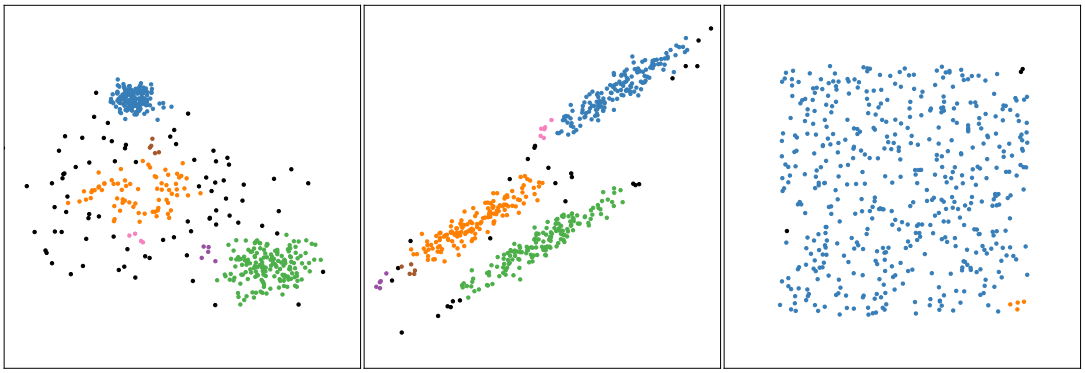

Figure 8.5.: How DBSCAN recognized Clusters and Outliers [28]

Figure 8.5 illustrates how DBSCAN does not fall prey to outliers, as k-Means does. A broad distribution of data points in the left example is very intuitively grouped into clusters and outliers. Especially the center picture showcases the ability of DBSCAN to recognize clusters by their density. The even spread of data in the third example is also

correctly identfied as one large cluster, even though a small part of the data was not within `eps` range.

But areas of high density do not say anything about being topically related.

8.1.3. Topic Model

After documents are clustered, each cluster needs a meaningful label to provide actual value. This task is called topic modeling. Several methods exist for generating or extracting labels from clustered data.

Firstly, the word frequency in one cluster can determine the topic. Each word is evaluated with respect to its frequency over the whole cluster. Either the most frequent one or top n words can be the topic. This method is quite prone to skewing by one document containing a word many times.

The second method uses the document frequency. This measure is also part of the TF-IDF measure (7.3.1). The TF-IDF vectorization method had to basic assumptions:

1. Words with a high frequency in a document describe it well.
2. Words with a high frequency in a corpus do not describe one specific document well.

But the second assumption does not hold true any more when applied to a cluster. Clusters should already be very similar, and contain similar words. Therefore, words occurring often over many different documents describe the cluster very well. The metric for this is the document frequency:

$$DF(t_i) = \frac{\#(d_{t_i})}{|C|} = \frac{number\ of\ documents\ containing\ word\ t_i}{number\ of\ documents\ in\ cluster\ C}$$

The document frequency of a word is the relative share of documents in the cluster it occurs in.

8.1.4. Dimensionality Reduction

The embeddings generated in the data preparation section are 512-dimensional. While the clustering algorithms work well with processing this data, 512 dimensions are not imaginable with the human mind. To visualize clusters, these dimensions have to be projected onto a two-dimensional plane. This task is called Dimensionality Reduction (DR), and is unsupervised.

Selecting a DR method, there are important aspects to consider:

- How well can the algorithm preserve the distances in the clusters (local structure)?
- How well can the algorithm preserve the distances between the clusters (global structure)?

Data scientists have many methods for DR at hand, some of the most popular ones being t-SNE and PCA.

The PCA identifies principal components in the dataset with an unsupervised ML algorithm.[2] A principal component is an axis in the multidimensional space along which the variance is maximized [44]. The number of learned principal components depends on the number of target dimensions, in the case of visualizations either two or three dimensions and principal components. The principal components are orthogonal to each other and span the hyperplane on which the values are projected [44]. It is important to note that PCA approximates the values, trading in accuracy for performance [2].

t-SNE is focused on retaining as much structure, both globally and locally. It does so by computing pairwise similarities in the dataset. The algorithm uses these similarities to accurately represent the same distances in a lower number of dimensions. This design of t-SNE helps to represent existing clusters very well.

The two methods are contrasted on the task of representing three-dimensional clusters (Figure **??**) in two dimensions. This task is quite similar to the later application of the DR algorithm. In figure **??** the PCA algorithm proves that it is able to represent the global structure in some way, but not quite satisfactory. It is heavily outperformed by t-SNE, in both preserving the global distances and the local structures. Apart from two clusters merging, the algorithm forms very distinct and clean clusters.

Overall, design and the practical application of the t-SNE algorithm make the case for it.

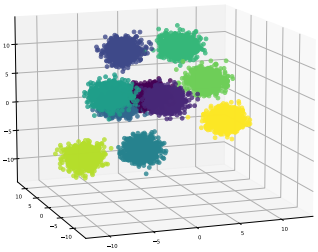

Figure 8.6.: Three-dimensional Clusters

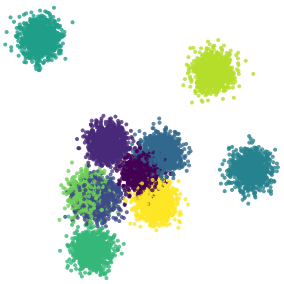

Figure 8.7.: PCA

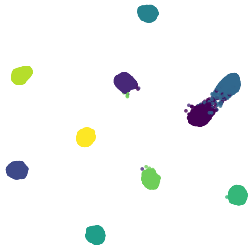

Figure 8.8.: t-SNE

Task	**Alternatives**	**Planned Implementation**
Distance Measure	Dot Product Euclidean Distance Cosine Distance	Cosine Distance
Clustering	k-Means DBSCAN	DBSCAN for outlier detection k-Means for clustering
Topic Modeling	Word Frequency Document Frequency	Document Frequency
Dimensionality Reduction	PCA t-SNE	t-SNE

8.2. Theoretical Implementation

The table **??** sums up all available alternatives from the prior section and the chosen alternative. The implementation of the modeling part starts with an analysis of the cosine distances in the dataset. For this, the KNeighbors algorithm will show the distance of each datapoint to its next neighbor. Next the outliers are sorted out with the DBSCAN clustering method. Before training the k-Means model, the ideal number of clusters has to be found with the elbow method. The k-Means model identifies the clusters. Topics per cluster are generated and the clusters are visualized with t-SNE.

8.3. Practical Implementation

8.3.1. Determining composition of dataset with KNeighbors Algorithm

Both density-based and distance-based clustering algorithms work with distances between objects. Before any clustering is applied, an overview over the distances should be established [23]. This allows to make meaningful decisions while hyperparameter-tuning.

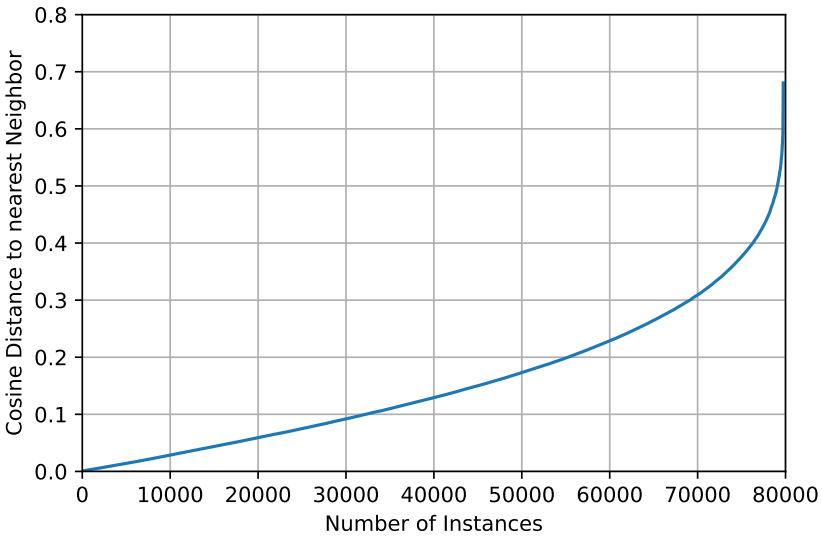

Figure 8.9.: Each Instance's Distance to the nearest neighboring Point

This can be done with the algorithm `sklearn.neighbors.NearestNeighbors`, which takes the input of a dataset with arbitrary dimensions. Using a specified distance metric, the algorithm outputs the distance of each data point to its nearest neighbor in the data set. The distance metric used is the cosine distance. The distance search shows that almost all points are within a maximum distance of 0.4 to the next neighbor. All other points are outliers and will be filtered out with DBSCAN.

8.3.2. Preselect documents with DBSCAN and cosine distance

The DBSCAN algorithm outputs, apart from clusters, outlier data points which do not belong into any cluster. With this algorithm, outliers in the data are detected to prevent the main clustering algorithm (KMeans) from overfitting. The clustering algorithm has the parameters `eps` and `metric` which tune the model.

The (distance) `metric` is of course cosine. The `eps` stands for the maximal distance between two points to be considered a neighbor [45]. As established before, the outliers are those points which are more than 0.4 units separated from the next data point. Therefore, `eps` is set to 0.4.

The DBSCAN model identified the outlier data (A.7), amounting to about 3% of the data points. These data points are not included into clustering by k-Means.

8.3.3. Determine the number of clusters with the Elbow Method

The KMeans algorithm is parameterized with the number of clusters (`n_clusters`). Since the algorithm itself is not able to determine the cluster count, this number has to be approximated. [21] suggest several applicable methods, among them are rule of thumb, and the elbow method with a silhouette score. A good point for starting with choosing `k` is the rule of thumb $k \approx \sqrt{\frac{n}{2}}$ with n being the number of instances in the dataset. Other literature also names the approximation of $k \approx log(n)$, but [22] showcase how k should be chosen in the order of n.

heuristic for best k	k
$k \approx \sqrt{\frac{n}{2}}$	199
$k \approx log(n)$	12
$k \approx n$	e.g. 10 000, 20 000

Table 8.1.: Three possible estimates for the optimal k

With the elbow method, the silhouette score of clustering results for different values of `k` are visualized.

The silhouette score is a measure of how similar intra-cluster points are and at the same time how dissimilar inter-cluster points are. A silhouette score of 1, being the highest value, is the most desirable.

The silhouette score s of each data point i can be calculated with the following formula [21]:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

The measure a denotes the mean distance to other points in the same cluster as i. This is interpreted as the inter-cluster dissimilarity, the higher $a(i)$, the more i doesn't belong into the cluster.

The measure b stands for the mean distance between i and the points in the nearest other cluster. To identify the nearest other cluster to the point i, all mean distances to points of each cluster are calculated. The cluster with the smallest mean distance between all its points and i is the neighboring cluster. The measure b is interpreted as the similarity of i to other clusters.

Being calculated for every point, this measure always falls in the interval $[-1; 1]$. For $S = 1$, the relationship of $a(i)$ and $b(i)$ has to be $a(i) \ll b(i)$ so that the denominator and divisor each tend to $b(i)$ resulting in $S = 1$. This would mean that the point i would have to be very far away from points in other clusters and very near to its own cluster. The value $S = 1$ is therefore the most desirable.

For $S = 1$, the relationship of $a(i)$ and $b(i)$ has to be $a(i) \gg b(i)$ so that the denominator and divisor each tend to $-a(i)$ respective $a(i)$ resulting in $S = -1$. In distance terms this means that the point i is very far away from its own cluster points and very close to points in the next nearest cluster.

Therefore, the task of the elbow method using the silhouette coefficient is to maximize S while preventing overfitting.

The above mentioned estimates (8.1) are a starting point for choosing values for k during the calculation of silhouette coefficients.

8.3.4. Train a KMeans Model

8.3.5. Visualize cluster formation

8.3.6. Generate Topics per Cluster

one hot encoding/binary

9. Deployment

Tableau Dashboard

10. Evaluation of the result

10.0.1. Machine Learning

Already Alan Turing understood that for laymen a learning machine can be perceived as a paradox. How can a machine learn, if a human has to define its behavior beforehand? There are three major subfields in the discipline of artificial intelligence that fundamentally explain how a computer can learn how to behave despite predefined behavior.

10.0.2. Supervised Learning

A supervised learning algorithm learns its decision with the help of a data set (input) that also contains the correct decision (output) as information. It is trained with only a part of the entire data set, so that the model can be tested in a later step with the help of unknown data. This way, a statement can be made about the accuracy of the model.

10.0.3. Unupervised Learning

Unsupervised learning is complementary to supervised learning. All algorithms that fall into the category of unsupervised learning are trained with data that does not contain the correct output (label) as information. Here, the categorization is not constrained by the given data, but decided on by the algorithm.

10.0.4. Reinforcement Learning

The third way in which an algorithm can make better decisions as it gains experience is called reinforcement learning. Reinforcement learning is about letting algorithms solve very complex tasks. The special feature is that there is no defined solution path, but the algorithm is rewarded for goal-oriented behavior and punished for wrong decisions. The definition of goal-oriented behavior has to be put into place by the engineers setting up

the training of the model. Real-world tasks are extremely complex, so not all possible solution paths can be calculated and compared to find the optimal path. Parking a car is a routine task for a human after a few hours of driving, but a computer sees only an infinite set of possibilities for turning angles. This problem can be solved by reinforcement learning. The algorithm is rewarded for each parking attempt where the car ends up seeing in the parking space. For the remaining attempts, the algorithm is penalized. Over many thousands of attempts, the reinforcement learning model is trained in this way.

The three major ways of learning even with previously defined behavior can now be implemented by specific models. For example, there are several ways to create and train a model using Unsupervised Learning.

Tf-Idf

Word2Vec

BERT

10.1. Visualization

10.2. Measures

11. Conclusion and Outlook

11.1. Conclusion

11.2. Outlook

Other process model could have been used. One document could belong into many clusters (sparse clusters) Not universal sentence encoder but word encoder weighted

References

[1] ____. *Why BERT Has 3 Embedding Layers and Their Implementation Details*. Medium. 2021-01-03. URL: https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a (visited on 2022-05-20).

[2] Ahmad, I. *40 Algorithms Every Programmer Should Know: Hone Your Problem-Solving Skills by Learning Different Algorithms and Their Implementation in Python*. Packt Publishing, 2020. URL: https://books.google.de/books?id=4IzrDwAAQBAJ.

[3] Alammar, J. *The Illustrated BERT, ELMo, and Co. (How NLP Cracked Transfer Learning)*. URL: https://jalammar.github.io/illustrated-bert/ (visited on 2022-05-20).

[4] Alammar, J. *The Illustrated Transformer*. URL: https://jalammar.github.io/illustrated-transformer/ (visited on 2022-05-19).

[5] Azeroual, O./ Saake, G./ Wastl, J. "Data Measurement in Research Information Systems: Metrics for the Evaluation of Data Quality". In: *Scientometrics* 115.3 (2018-06), p. 1279. URL: http://link.springer.com/10.1007/s11192-018-2735-5 (visited on 2022-05-07).

[6] Ba, J. L./ Kiros, J. R./ Hinton, G. E. "Layer Normalization". 2016-07-21. arXiv: 1607.06450 [cs, stat]. URL: http://arxiv.org/abs/1607.06450 (visited on 2022-05-19).

[7] Ball, D. *It's Costing How Much? The Truth about Manual Invoice Processing*. SME Magazine. 2018-05-02. URL: https://www.smeweb.com/2018/05/02/costing-much-truth-manual-invoice-processing/ (visited on 2022-03-28).

[8] Beyer, K. et al. "When Is "Nearest Neighbor" Meaningful?" In: *LNCS*. International Conference on Database Theory. Vol. 1540. Springer-Verlag Berlin Heidelberg, 1998.

[9] Brown, S. *Why External Data Should Be Part of Your Data Strategy*. MIT Sloan. URL: https://mitsloan.mit.edu/ideas-made-to-matter/why-external-data-should-be-part-your-data-strategy (visited on 2022-04-06).

[10] Brownlee, J. *7 Ways to Handle Large Data Files for Machine Learning*. Machine Learning Mastery. 2020-12-10. URL: https://machinelearningmastery.com/large-data-files-machine-learning/ (visited on 2022-04-28).

[11] Chapman, P. et al. *CRISP-DM 1.0: Step-by-step Data Mining Guide*. 2000.

[12] Chowdhury, G. G. "Natural Language Processing". In: *Annual Review of Information Science and Technology* 37.1 (2003-01-31). URL: https://strathprints.strath.ac.uk/2611/1/strathprints002611.pdf.

[13] Chun, W. *Core Python Programming*. Vol. 1. Prentice Hall Professional, 2006-09.

[14] Devlin, J. et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". 2019-05-24. arXiv: 1810.04805 [cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 2022-05-20).

[15] Ester, M./ Kriegel, H.-P./ Xu, X. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: (), p. 6.

[16] Fayyad, U./ Piatetsky-Shapiro, G./ Smyth, P. "From Data Mining to Knowledge Discovery in Databases". In: *American Association for Artificial Intelligence* 17.3 (1996).

[17] Gant, M. *Scrum and the Solo Dev*. Medium.com. 2019-04-15. URL: https://medium.com/@jmgant.cleareyeconsulting/scrum-and-the-solo-dev-fb8e810ed42b (visited on 2022-03-29).

[18] Hayes, A. *Understanding Invoices*. Investopedia. URL: https://www.investopedia.com/terms/i/invoice.asp (visited on 2022-03-28).

[19] Jupyter Team. *Architecture*. URL: https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html (visited on 2022-04-30).

[20] Koch, B. *The E-Invoicing Journey 2019-2025*. Billentis. URL: https://www.billentis.com/The_einvoicing_journey_2019-2025.pdf.

[21] Kodinariya, T./ Makwana, P. "Review on Determining of Cluster in K-means Clustering". In: *International Journal of Advance Research in Computer Science and Management Studies* 1 (2013-01-01), pp. 90–95.

[22] Maier, M./ Hein, M./ von Luxburg, U. "Optimal Construction of K-Nearest Neighbor Graphs for Identifying Noisy Clusters". In: *Theoretical Computer Science* 410.19 (2009-04), pp. 1749–1764. arXiv: 0912.3408 [math, stat]. URL: http://arxiv.org/abs/0912.3408 (visited on 2022-05-18).

[23] Maklin, C. *DBSCAN Python Example: The Optimal Value For Epsilon (EPS)*. Medium. 2022-05-09. URL: https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc (visited on 2022-05-18).

[24] Martinez-Plumed, F. et al. "CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories". In: *IEEE Transactions on Knowledge and Data Engineering* 33.8 (2021-08-01).

[25] Metwalli, S. A. *What to Do When Your Data Is Too Big for Your Memory?* URL: https://towardsdatascience.com/what-to-do-when-your-data-is-too-big-for-your-memory-65c84c600585 (visited on 2022-04-13).

[26] Mikolov, T. et al. "Efficient Estimation of Word Representations in Vector Space". 2013-09-06. arXiv: 1301.3781 [cs]. URL: http://arxiv.org/abs/1301.3781 (visited on 2022-03-24).

[27] Muller, B. *BERT 101 - State Of The Art NLP Model Explained*. 2022-03-02. URL: https://huggingface.co/blog/bert-101 (visited on 2022-05-20).

[28] Pedregosa, F. et al. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[29] *Python - Official Image | DockerHub*. URL: https://hub.docker.com/_/python (visited on 2022-04-20).

[30] *Python vs. R: What's the Difference?* URL: https://www.ibm.com/cloud/blog/python-vs-r.

[31] *R-Base - Official Image*. URL: https://hub.docker.com/_/r-base.

[32] Reimers, N./ Gurevych, I. "Making Monolingual Sentence Embeddings Multilingual Using Knowledge Distillation". 2020-10-05. arXiv: 2004.09813 [cs]. URL: http://arxiv.org/abs/2004.09813 (visited on 2022-05-22).

[33] Reimers, N./ Gurevych, I. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks”. 2019-08-27. arXiv: 1908.10084 [cs]. URL: http://arxiv.org/abs/1908.10084 (visited on 2022-05-22).

[34] Rundle, D. *3 Keys to Managing Data and Maximizing Business Intelligence*. URL: https://worthwhile.com/insights/2017/02/20/data-business-intelligence/ (visited on 2022-04-06).

[35] Rutschmann, D. *The Journey from SAP Leonardo Machine Learning Foundation to SAP AI Core and SAP AI Launchpad*. SAP Community. 2021-10-11. URL: https://blogs.sap.com/2021/10/11/the-journey-from-sap-leonardo-machine-learning-foundation-to-sap-ai-core-and-sap-ai-launchpad/ (visited on 2022-03-16).

[36] Saltz, J. *CRISP-DM Is Still the Most Popular Framework for Executing Data Science Projects*. Data Science Process Alliance. 2020-11-30. URL: https://www.datascience-pm.com/crisp-dm-still-most-popular/ (visited on 2022-03-29).

[37] SAP SE. *AI Research*. URL: https://www.sap.com/products/artificial-intelligence/research.html (visited on 2022-04-27).

[38] SAP SE. *Geschichte Der SAP - Die Anfangsjahre 1972-1980*. URL: https://www.sap.com/germany/about/company/history/1972-1980.html (visited on 2022-03-15).

[39] SAP SE. *Juergen Mueller Biography*. URL: https://www.sap.com/about/company/leadership/juergen-mueller.html (visited on 2022-03-15).

[40] Scale AI. *Scale Document AI: Template-Free, High-Quality Extraction*. Scale Document AI: Template-Free, High-Quality Extraction. URL: https://scale.com/document-ai (visited on 2022-05-09).

[41] Schmitz, A. *Was ist SAP Leonardo?* 2017-07-11. URL: https://news.sap.com/germany/2017/07/was-ist-sap-leonardo-iot/ (visited on 2022-03-16).

[42] Schopf, T. *Parallel Inference of HuggingFace Transformers on CPUs*. Medium. 2022-02-22. URL: https://towardsdatascience.com/parallel-inference-of-huggingface-transformers-on-cpus-4487c28abe23 (visited on 2022-05-22).

[43] Sculley, D. “Web-Scale k-Means Clustering”. In: *Proceedings of the 19th International Conference on World Wide Web - WWW ’10*. The 19th International Conference. Raleigh, North Carolina, USA: ACM Press, 2010, p. 1177. URL: http://portal.acm.org/citation.cfm?doid=1772690.1772862 (visited on 2022-05-18).

[44] Sivarajah, S. *Dimensionality Reduction for Data Visualization: PCA vs TSNE vs UMAP vs LDA*. Medium. 2020-12-31. URL: https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29 (visited on 2022-05-21).

[45] *Sklearn.Cluster.DBSCAN*. scikit-learn. URL: https://scikit-learn/stable/modules/generated/sklearn.cluster.DBSCAN.html (visited on 2022-05-18).

[46] Srivastava, A. *Difference Between Data Science and Data Mining*. The Startup. 2020-07-25. URL: https://medium.com/swlh/difference-between-data-science-and-data-mining-37104b1c6a61 (visited on 2022-04-06).

[47] Taylor, C. *Structured vs Unstructured Data*. Datamation. 2021-05-21. URL: https://www.datamation.com/big-data/structured-vs-unstructured-data/ (visited on 2022-03-28).

[48] Tok, J. *Memory Optimisation – Python DataFrames vs Lists and Dictionaries (JSON-like)*. 2021-06-07. URL: https://www.joeltok.com/blog/2021-6/memory-optimisation-python-dataframes-vs-json-like (visited on 2022-05-13).

[49] *What Is the Purpose of an Invoice? | Invoicing Tips for Small Business*. FreshBooks. URL: https://www.freshbooks.com/hub/invoicing/how-do-invoices-work (visited on 2022-03-28).

A. Appendix

A.1. Inventory of Resources

Table A.1.: Inventory of Resources

Type of resources	**Kind of resources**	**Quantification**
Personnel	business experts	1 person
	data experts	1 person
	technical support	1 person
	data mining experts	1 person
Data	fixed extracts	-
	live data	-
	warehoused data	-
	operational data	1 dataset
Computing resources	hardware platforms	1 machine access to SAP AI Core
Software	data mining tools	anaconda, jupyter
	other relevant software	Excel, Visual Studio Code, Git

*) An asterisk indicates theoretical availability of more resources, if needed.

A.2. Invoice Header Data

accountNumber.key
accountNumber.value
barCode.value
buyerAddress.cityTownVillage.value
buyerAddress.country.value
buyerAddress.district.value
buyerAddress.extraName.value
buyerAddress.full.key
buyerAddress.full.value
buyerAddress.houseNumber.value
buyerAddress.stateProvince.value
buyerAddress.street.value
buyerAddress.zip.value
buyerName.key
buyerName.value
comments.key
comments.value
country.value
currency.key
currency.value
deliveryDate.key
deliveryDate.value
deliveryNoteNo.key
deliveryNoteNo.value
discount.key
discount.value
dueDate.key
dueDate.value
employeeName.key
employeeName.value
exchRate.key
exchRate.value
exchRateSrcCurr.key
exchRateSrcCurr.value
exchRateTarCurr.key
exchRateTarCurr.value
filename
index
invoiceAmount.key
invoiceAmount.value
invoiceDate.key
invoiceDate.value
invoiceNo.key
invoiceNo.value
invoiceType.value
language
paymentTerms.key
paymentTerms.value
pii.address.key
pii.address.value
pii.email.key
pii.email.value
pii.name.key
pii.name.value
pii.other.key
pii.other.value
pii.phone.key
pii.phone.value
purchaseOrderNo.key
purchaseOrderNo.value
shipToAddress.cityTownVillage.value
shipToAddress.country.value
shipToAddress.district.value
shipToAddress.extraName.value

shipToAddress.full.key
shipToAddress.full.value
shipToAddress.houseNumber.value
shipToAddress.stateProvince.value
shipToAddress.street.value
shipToAddress.zip.value
shippingAmount.key
shippingAmount.value
subtotalAmount.key
subtotalAmount.value
tableHeader.batchNumber.value
tableHeader.description.value
tableHeader.discount.value
tableHeader.materialNumber.value
tableHeader.purchaseOrderNumber.value
tableHeader.quantity.value
tableHeader.serialNumber.value
tableHeader.tableHeaderBox
tableHeader.tax10Amount.value
tableHeader.tax10Rate.value
tableHeader.tax1Amount.value
tableHeader.tax1Rate.value
tableHeader.tax2Amount.value
tableHeader.tax2Rate.value
tableHeader.tax3Amount.value
tableHeader.tax3Rate.value
tableHeader.tax4Amount.value
tableHeader.tax4Rate.value
tableHeader.tax5Amount.value
tableHeader.tax5Rate.value
tableHeader.tax6Amount.value
tableHeader.tax6Rate.value
tableHeader.tax7Rate.value
tableHeader.totalAmount.value
tableHeader.unitOfMeasure.value
tableHeader.unitPrice.value
tax10Amount.key
tax10Amount.value
tax10Description.key
tax10Description.value
tax10Rate.key
tax10Rate.value
tax1Amount.key
tax1Amount.value
tax1Description.key
tax1Description.value
tax1Rate.key
tax1Rate.value
tax2Amount.key
tax2Amount.value
tax2Description.key
tax2Description.value
tax2Rate.key
tax2Rate.value
tax3Amount.key
tax3Amount.value
tax3Description.key
tax3Description.value
tax3Rate.key
tax3Rate.value
tax4Amount.key
tax4Amount.value
tax4Description.key
tax4Description.value
tax4Rate.key
tax4Rate.value
tax5Amount.key
tax5Amount.value

tax5Description.value
tax5Rate.key
tax5Rate.value
tax6Amount.key
tax6Amount.value
tax6Description.key
tax6Description.value
tax6Rate.key
tax6Rate.value
tax7Amount.key
tax7Amount.value
tax7Description.value
tax7Rate.key
tax7Rate.value
tax8Amount.key
tax8Amount.value
tax8Description.value
tax8Rate.key
tax8Rate.value
tax9Amount.key
tax9Amount.value
tax9Description.value
tax9Rate.key
tax9Rate.value
totalAmount.key
totalAmount.value
vendorAddress.cityTownVillage.value
vendorAddress.country.value
vendorAddress.district.value
vendorAddress.extraName.value
vendorAddress.full.key
vendorAddress.full.value
vendorAddress.houseNumber.value
vendorAddress.stateProvince.value
vendorAddress.street.value
vendorAddress.zip.value
vendorBankAccountNo.key
vendorBankAccountNo.value
vendorName.key
vendorName.value
vendorTaxID.key
vendorTaxID.value

A.3. Invoice Line Item Data

lineItem.batchNumber.value
lineItem.description.value
lineItem.discount.value
lineItem.lineItemBox
lineItem.materialNumber.value
lineItem.purchaseOrderNumber.value
lineItem.quantity.value
lineItem.serialNumber.value
lineItem.tax10Amount.value
lineItem.tax10Rate.value
lineItem.tax1Amount.value
lineItem.tax1Rate.value
lineItem.tax2Amount.value
lineItem.tax2Rate.value
lineItem.tax3Amount.value
lineItem.tax3Rate.value
lineItem.tax4Amount.value
lineItem.tax4Rate.value
lineItem.tax5Amount.value
lineItem.tax5Rate.value
lineItem.tax6Amount.value
lineItem.tax6Rate.value
lineItem.tax7Amount.value
lineItem.tax7Rate.value
lineItem.tax8Amount.value
lineItem.tax8Rate.value
lineItem.tax9Amount.value
lineItem.tax9Rate.value
lineItem.totalAmount.value
lineItem.unitOfMeasure.value
lineItem.unitPrice.value

A.4. Benchmarking DataFrames and Dictionaries

```
import numpy, pandas
df = pandas.DataFrame(numpy.zeros(shape=[10, 10]))
df
```

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
%timeit value = df.loc[5, 5]   # label-based indexing
%timeit value = df.at[5, 5]    # label-based scalar lookup
%timeit value = df.iloc[5, 5]  # integer-based indexing
%timeit value = df.iat[5, 5]   # integer-based scalar lookup
```

```
6.32 µs ± 140 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
3.34 µs ± 343 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
14.8 µs ± 571 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
11.8 µs ± 1.38 µs per loop (mean ± std. dev. of 7 runs, 100000 loops eac
h)
```

```
dictionary = df.to_dict()
%timeit value = dictionary[5][5]
```

```
52.1 ns ± 3.74 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops ea
ch)
```

Listing A.1: Benchmark of Indexing with Python Data Structures

The `.loc` indexing is 130.24 times slower than dictionary access.
The `.at` indexing is 62.06 times slower than dictionary access.
The `.iloc` indexing is 310.28 times slower than dictionary access.
The `.iat` indexing is 213.44 times slower than dictionary access.

A.5. File Processing Script

```
def read_invoice(filename):

    path = '../data/'+filename
    with open(path) as data_file:
        data = json.load(data_file)
        annotations = data.get('response').get('annotations')

    labels = ['filename']
    texts = [filename]

    for ann in annotations:
        label = ann.get('label')
        if 'lineItem' in label or label in labels:
            pass
        else:
            labels.append(ann.get('label'))
            texts.append(ann.get('text'))

    return pd.DataFrame([texts], columns=labels)
```

Listing A.2: Python Script for extracting JSON Data into a DataFrame

A.6. TF-IDF

	aa	aaa	aaaparisto	aac	aachen	aag	aagaard	aaj	aak	aakilde	...	화환	활용	황혁준	회선	회선료	회수	회의	회차	후렌치	휴대용
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
79736	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79737	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79738	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79739	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79740	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure A.1.: Features extracted with TF-IDF

```
1 df_tfidf.sample(n=10, axis='columns').sum(axis=0)
```

```
beer                        8.955057
ascotia                     0.427341
fólia                       1.949317
firmowa                     5.466856
cordão                      0.387344
teu                         0.485548
lista                       6.209191
perea                       2.896118
steuerschuldnerschaft       0.157522
almega                      3.012966
dtype: float64
```

Figure A.2.: Ten words from the Vocabulary and their summed Weight

A.7. Architectures for Learning Word Embeddings

Continuous Bag of Words With the CBOW architecture the word2vec model is trained to predict a word using the context as input. A sliding window of a predefined size is moved along the text.

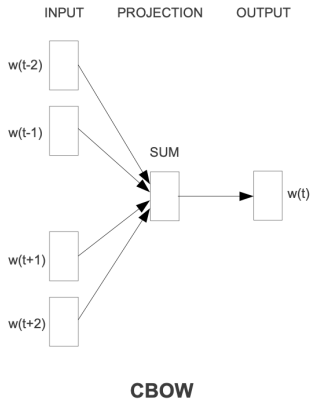

Figure A.3.: Continuous Bag of Words architecture
with sliding window of size $C = 5$

Skipgram With the skipgram architecture the word2vec model is trained to predict the context of the input word.

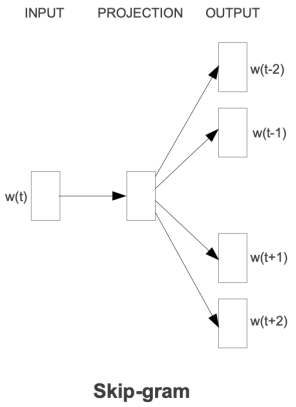

Figure A.4.: Skipgram architecture
with sliding window of size $C = 5$

input	target
secretary	my
secretary	did
did	secreatary
did	the
the	did
the	ticket
ticket	the
ticket	reservation

Figure A.5.: Observations for training with skipgram architecture

Negative Sampling Word2Vec uses either SKIPGram or CBOW.

Word2vec can utilize two models for selecting observations: either skipgram or CBOW.

A.8. Transformer Architecture

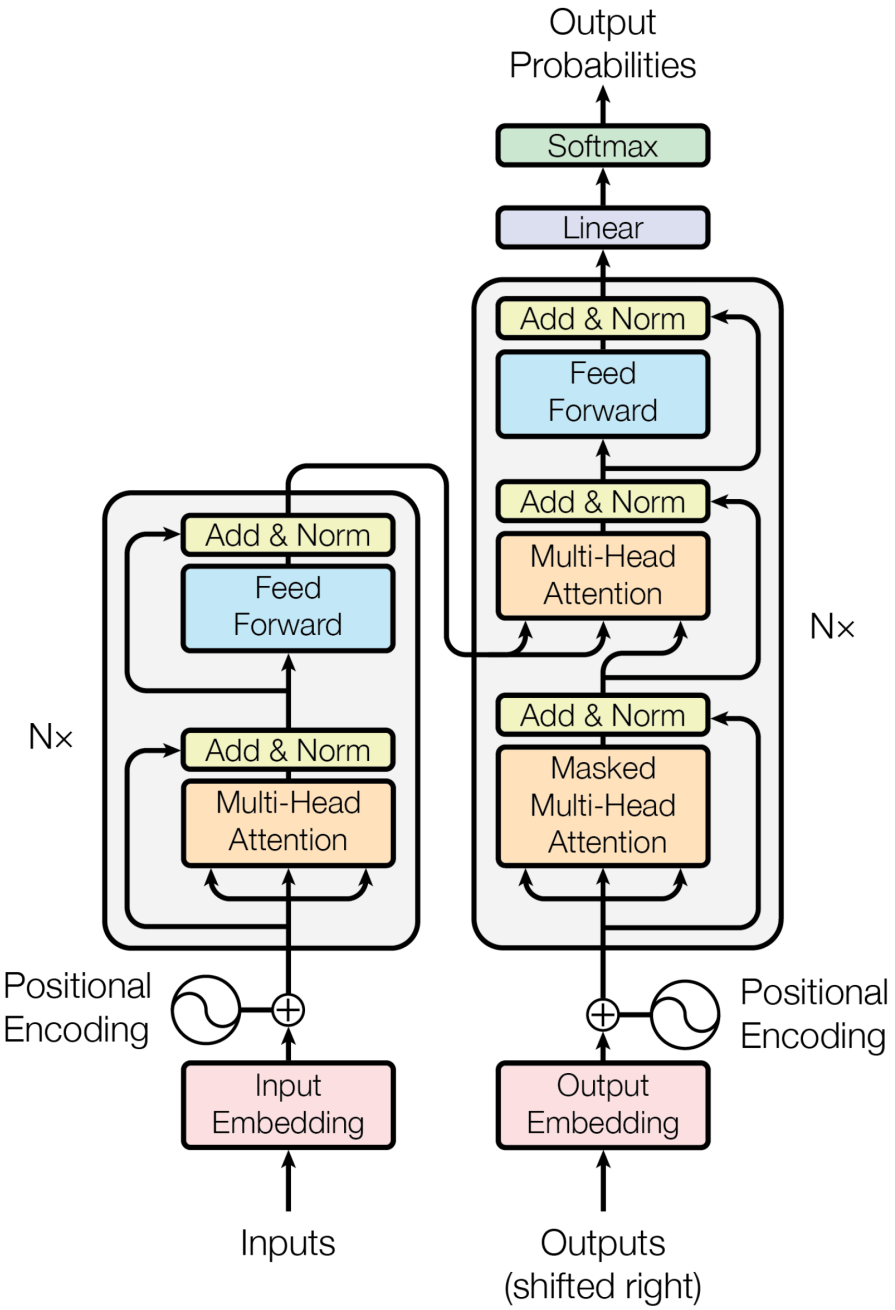

Figure A.6.: Observations for training with skipgram architecture

A.9. Outliers

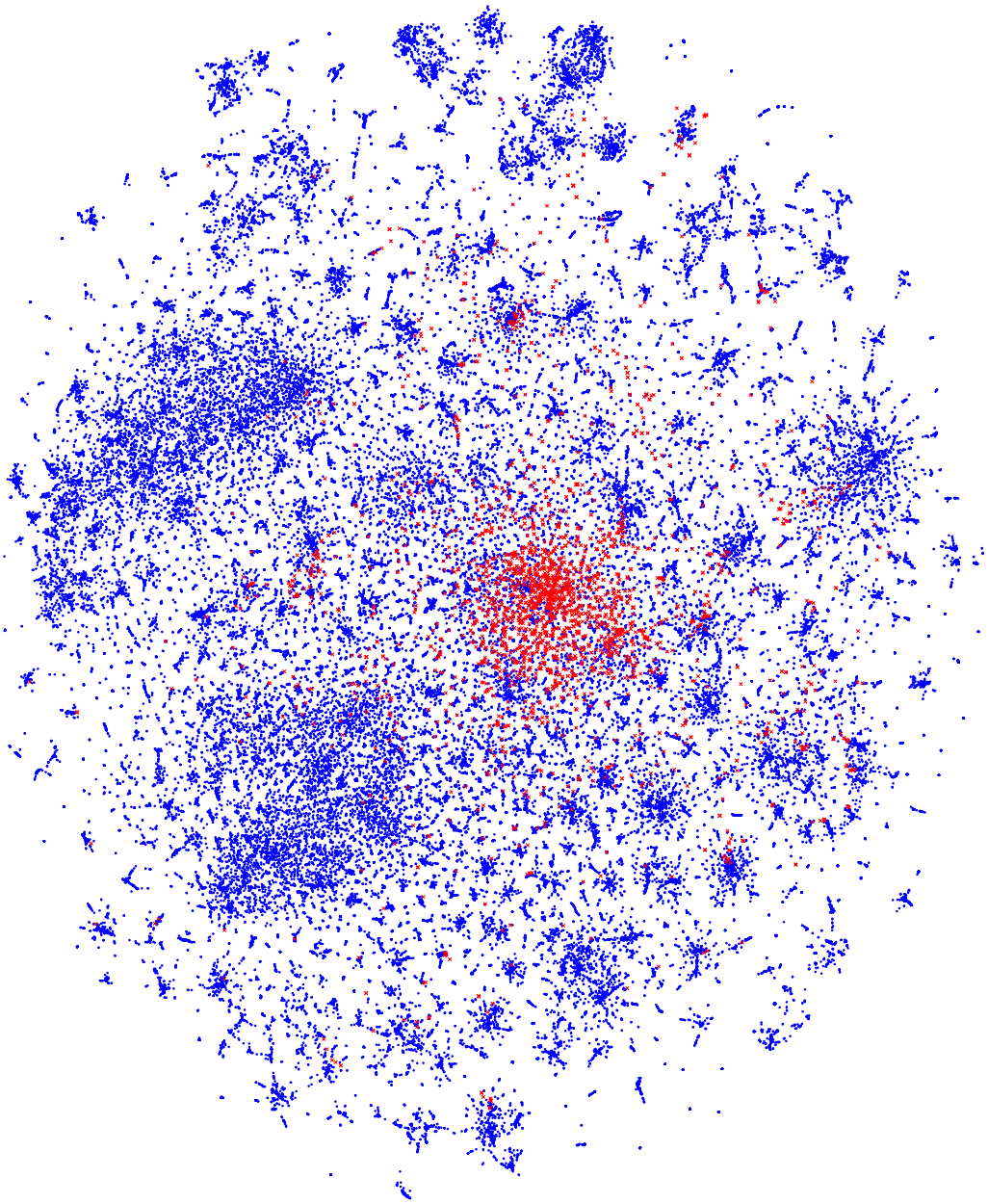

Figure A.7.: Projection of the Data with Outliers marked Red