



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: 01. Februar 2018

Company Supervisor: Dr. Karthik Muthuswamy

Academic Supervisor: Prof. Dr. Joachim Melcher

Contents

Abkürzungsverzeichnis	III
List of Figures	IV
List of Tables	V
Listings	VI
1 Introduction	1
1.1 Motivation	1
1.2 Current situation	1
1.3 Research Questions	2
1.4 Outline	2
2 Objectives and Criteria	3
2.1 Detailed Task Description	3
2.2 Criteria set by SAP SE	3
2.3 Research Model	3
3 Fundamentals	6
3.1 Glossary of Terms	6
3.2 Corporate Environment	7
3.3 Machine Learning	8
3.4 SAP AI Core	9
4 Dataset selection	10
4.1 Alternatives	10
4.2 Practical Implementation	11
5 Business Understanding	12
6 Data Understanding	13
7 Data Preparation	14
7.1 Data preparation	14
7.2 Data preparation	19
8 Evaluation of the result	28
8.1 Visualization	28
8.2 Measures	28

9 Conclusion and Outlook	29
9.1 Conclusion	29
9.2 Outlook	29
Literaturverzeichnis	I

Abkürzungsverzeichnis

AI	Artificial Intelligence
AI BUS	SAP AI Business Services
NLP	Natural Language Processing
CRISP-DM	Cross Industry Standard Process for Data Mining
CoE	Center of Excellence
BoW	Bag of Words
TF-IDF	term frequency - inverse document frequency
CBOW	Continuous Bag of Words
JSON	JavaScript Object Notation
KDD	Knowledge Discovery in Databases
SEMMA	Sample Explore Modify Model Assess
GIL	Global Interpreter Lock
NLTK	Natural Language Toolkit

List of Figures

2.1	Adjusted CRISP-DM Model	4
7.1	Exemplary depiction for processed invoices in a DataFrame	15
7.2	Exemplary depiction for processed invoice items in a DataFrame	16
7.3	Exemplary depiction for processed invoice items in a DataFrame	16
7.4	Entity Relationship Diagram for Invoice Documents	20
7.5	Continuous Bag of Words architecture with sliding window of size $C = 5$. .	26
7.6	Skipgram architecture with sliding window of size $C = 5$	26
7.7	Observations for training with skipgram architecture	27

List of Tables

7.1	Comparison of available Storage Options	21
-----	---	----

Listings

7.1	JSON of one invoice	14
7.2	Shortened JSON Schema of one invoice document representation	17

1 Introduction

1.1 Motivation

An invoice is a document recording the main information regarding a sales transaction. Usually, an invoice contains the unit cost, a timestamp and the payment terms. Other information, such as shipping terms, shipping adress or payment conditions may also be included.

Apart from their use for tax records, tracking the inventory and legal protection, invoices are essential for a company's external and internal financial reporting (controlling). They are the main source of information for controlling [16], as they record the complete financial history of cash flow [17]. In general, interal financial reporting aims to provide information about the health of a company and supply means for improvement for a company's financials. In turn, this requires an in-depth analysis of spending.

According to [14], more than 4 out of 5 financial departments are "overwhelmed by the high numbers of invoices they are expected to process". Already overwhelmed 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. With analysis results, financial advisors have a factual base for recommendations.

1.2 Current situation

An essential part of economic counselling is the assessment of allocated spending for different segments of a company. Spending of a firm usually is written down in invoice documents, which have to be grouped for segments to analyze their costs. While the global market is estimated to comprise 550 billion invoices annually, 90% are exchanged paper-based [12]. 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 [15]. This data is

not leverageable with traditional data analysis tools, but its value must be harvested for companies to utilize their full potential. A large share of unstructured data found in companies is textual data.

1.3 Research Questions

Which methods exist for clustering large-scale multilingual corpora? Which combinations of feature-selection methods and clustering methods return the most valuable information?

1.4 Outline

This should be written at the end since it's very prone to changes.

2 Objectives and Criteria

2.1 Detailed Task Description

The goal of the thesis is to add value to real business documents by aggregating expenses into clusters of similar expenses. The supplied document dataset consists of 150.000 invoices. The invoices contain different information, for example the vendor, billing amount or a description of the goods. Valuable information for companies would be insight into the different categories of expenses and the corresponding cost. With traditional data analysis methods, the company's controlling departments cannot identify which expenses are similar in nature (for example logistics costs).

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 should be an added value to the dataset in the form of aggregated expenses.

2.2 Criteria set by SAP SE

The analysis should be performed utilizing only available resources, which are the student's company laptop and already available instances for SAP internal services.

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) [4]. 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 [13] conducted amongst users of a webpage about data science project management, found that almost half of all respondents most commonly 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 than CRISP-DM, Scrum and Kanban.

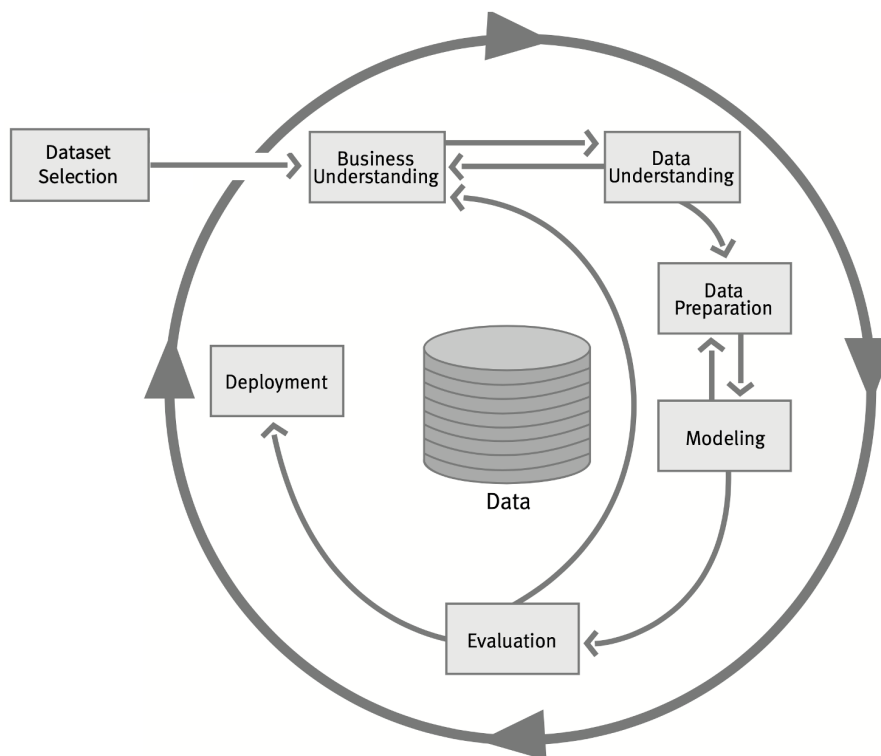


Figure 2.1: Adjusted CRISP-DM Model

Being the most popular model, CRISP-DM not necessarily is 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 modelling needs to occur before the evaluation, the preparation needs to occur before the modelling, and an adequate understanding of business and data aspects has to be developed in the beginning of the process. All those elemental 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 underly the premise of 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 [7]. Thirdly, the model allows for a tailored trail through its phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The free choice of path is more favorable compared to KDD, which allows for loops, but has a fixed order [7]. 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 [8]. Fifth, the CRISP-DM model was publicized in the context of a 70-page guide with generic task descriptions and outputs for each phase [4]. The detailed guide is especially valuable for teams with little experience, as it prevents tasks being forgotten.

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.

3 Fundamentals

3.1 Glossary of Terms

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

Natural Language Processing (NLP) is often attributed to computer science, but after closer examination, NLP is a discipline comprised of linguistics, computer science, artificial intelligence and mathematics [10].

Data Cleaning

Data Wrangling

Feature Extraction

- SAP SE
- SAP Leonardo MLF
- SAP AI Core
- SAP AI Launchpad
- AI Foundation

3.2 Corporate Environment

3.2.1 Historical

SAP was founded in 1972 by five ex-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" [9]. 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 AI business with the SAP Leonardo Machine Learning Foundation [6], and challenged the market for hyped products in machine learning, blockchain, big data and design thinking. The name Leonardo refers to Leonardo Da Vinci, who is renowned for his interdisciplinary innovations [2]. 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 [6]. With the general availability of AI Foundation in late 2021, SAP Leonardo is officially sunsetted.

- What comes in the future?
- how much does SAP earn with AI?

3.2.2 Organizational

SAP SE has an executive board consisting of seven members, each attributed to one area. The Artificial Intelligence (AI) division falls into the responsibility of Jürgen Müller, Chief Technology Officer and leader of the board area for technology and innovation [11]. 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 [1]. The research teams regularly publish articles on their advancements.

3.2.3 Technological

<https://www.sap.com/products/artificial-intelligence.html>

3.3 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.

3.3.1 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.

3.3.2 Unsupervised 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.

3.3.3 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.

3.3.4 Clustering Algorithms

multinomial, one bad example for a clustering would be the closest 5 docs to each one (this is multilabel)

3.4 SAP AI Core

3.4.1 Docker

4 Dataset selection

Which datasets are available?

4.1 Alternatives

The dataset is the fundament of a data-science project. The quality, size and closeness to reality decide the degree to which the findings can be helpful for solving real problems. In this section, a classification for data-science projects is introduced.

	Problem-First	Data-First
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

The problem-first project type is characterized by a predefined problem statement or research goal. The underlying question is, how a specific or a set of problems can be solved. 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 judgements can be sufficient. In a problem-first project, different datasets can and should be considered.

The complementary project type is the data-first project, which describes most data-mining projects. This type is characterized by a more explorative approach, and the goal to find problems and patterns. Here, the dataset is at the core of the project and the fixed aspect of a project. In turn, this means swapping the dataset is the start of a new project.

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.

	Internal	External
Source	Internal or customer data, bought or generated	Publicly available, generated or supplied by companies
Relevance	Business-relevant	Anonymized, processed
Value	Relevant to one business	Of general relevance
Availability	Authorization processes in place, no central registries	Ubiquitously available

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. Data is either directly or very closely related to the company's business. Depending on the solution, it can be of use to customers or it can be utilized inside the company. 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. Data sources include real-life data and data generated for educational purposes. Because of its publication, the data is stripped from all parts which could expose confidential information such as corporate secrets. Additionally, data is anonymized and processed to limit the usefulness to potential competitors.

It can be stated, that both sources are suited for different goals.

4.2 Practical Implementation

T

5 Business Understanding

In [4], different methods for business understanding are mentioned. Firstly, a

- why did i decide for clustering?

6 Data Understanding

- different data exploration tools can be used for the data understanding.

7 Data Preparation

7.1 Data preparation

7.1.1 Data processing and data wrangling

The data for an invoice and its contained items is stored in an array of objects. This is also shown in code example (7.2) of a shortened JSON Schema for one invoice. The array "annotations" contains objects. One object represents information such as the invoice date or the unit price. Information about invoice items have labels containing the prefix "lineItem". One invoice containing several items is represented by duplicate values of one label (in the example 7.1 the label "lineItem.description.value"). Here, the order of the labels has to be retained during data wrangling to ensure not mixing up information about different invoice items.

```
1  {
2      "annotations": [
3          {
4              "label": "vendorName.value",
5              "text": "Example IT Vendor"
6          },
7          {
8              "label": "invoiceDate.value",
9              "text": "2020-02-32"
10         },
11         {
12             "label": "lineItem.description.value",
13             "text": "Mouse"
14         },
15         {
16             "label": "lineItem.description.value",
17             "text": "HDMI Adapter"
18         },
```

```

19         {
20             "label": "lineItem.description.value",
21             "text": "Shipping Cost"
22         }
23     ]

```

Listing 7.1: JSON of one invoice

The files are processed in python, using the library "json". Each file is opened, and decoded with python's build in json decoder. The hierarchy is traversed until the array "annotations" is reached. Now, the tuples of label and text are saved. This process is split up into invoice and item labels.

The information on invoices is stored in a tabular data structure, a python 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. The filename is the unique identifier for one invoice.

	vendorName.value	invoiceDate.value	totalAmount.value
filename			
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 7.1: 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 invoice item has an unique id and can be linked to the respective invoice through the filename.

The resulting two DataFrames are serialized with the python standard library "pickle". The data can be efficiently loaded into memory and reserialized 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. Even further improvements to storage will be dicussed later on.

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

	filename	linelItem.totalAmount.value	linelItem.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 7.2: Exemplary depiction for processed invoice items in a DataFrame

processor was consequently at the maximum, only one process is executed, leaving the total CPU usage at only 20%.

```

I518232 — top — 91x13
Processes: 674 total, 5 running, 669 sleeping, 4145 threads
Load Avg: 5.28, 4.06, 3.96 CPU usage: 19.86% user, 9.27% sys, 70.86% idle
SharedLibs: 349M resident, 62M data, 18M linkedit.
MemRegions: 533222 total, 5046M resident, 146M private, 2401M shared.
PhysMem: 15G used (4050M wired), 578M unused.
VM: 26T vsize, 3100M framework vsize, 1069012590(126) swapins, 1116797165(0) swapouts.
Networks: packets: 30163699/31G in, 13333797/4979M out.
Disks: 67718340/4990G read, 30826238/4765G written.

PID    COMMAND    %CPU    TIME    #TH    #WQ    #PORTS    MEM    PURG    CMPRS    PGRP    PPID
88340  efilogin-hel 181.4   00:05.23 14/8   12/8   121-     345M+  22M-    0B      88340  1
49106  python3.9    97.1   06:55.47 15/1    1      32      1374M  0B      1217M  49106  42969
189    WindowServer 12.7   09:57:23 16      6      8271    1982M  448K+  479M   189    1

```

Figure 7.3: Exemplary depiction for processed invoice items in a DataFrame

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 [5]. This of course is not favorable, as valuable CPU capacity is unused. Fortunately, the GIL behaves in a special way regarding C code, and I/O operations in Python utilize C code: the lock is released before executing a C routine [5]. This allows to bypass the (in this case) inconvenient locking mechanism.

Different python libraries exploit this speciality 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  {
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 7.2: Shortened JSON Schema of one invoice document representation

7.1.2 Data cleaning

The task of data cleaning was also completed using python scripts. The library Natural Language Toolkit (NLTK)


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

7.1.3 Considerations of Space and Time Complexity

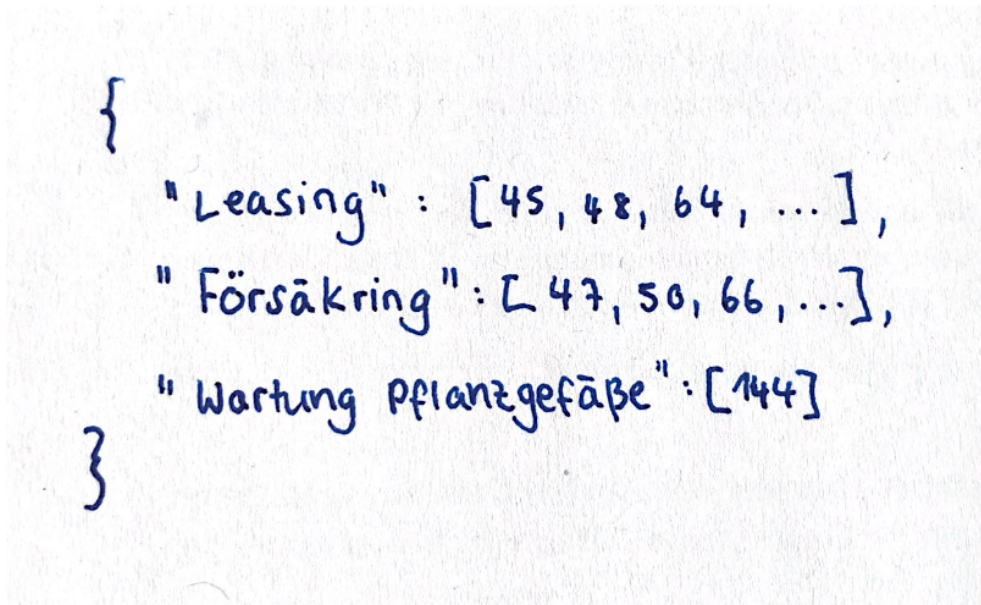
The dataset consists of over 150.000 invoices, and in those invoices, over 350.000 items are listed. With hardware-limitation in place, an optimized approach for storing and processing the data is required. Several considerations for speeding up processing time and reducing storage space can be made. In the following, the observations are explained and approaches for improvement are given.

Duplicates and space complexity

Investigation 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.

Reconstructing Relationships and time complexity of searching

After the deletion of duplicate descriptions (documents in the corpus), the



```
{  
  "Leasing": [45, 48, 64, ...],  
  "Försäkring": [47, 50, 66, ...],  
  "Vårdning pflanzgefäße": [144]  
}
```

7.1.4 Feature Extraction and Feature Engineering

7.1.5 Data processing and data wrangling

The data is available as a local folder of size 5.12 GB, it contains 152,592 JavaScript Object Notation (JSON) files. Each JSON represents one invoice document.

For the data wrangling the invoices have to be read into memory and then processed into a reusable structure. Each document will be read into memory, then the necessary information will be extracted. All invoices will be combined into one data structure, which then is persisted for later use.

Explain the structure of one invoice. Explain which data structures I want to use.

Explain how I want to store the data. Explain the successive artifacts.

7.2 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

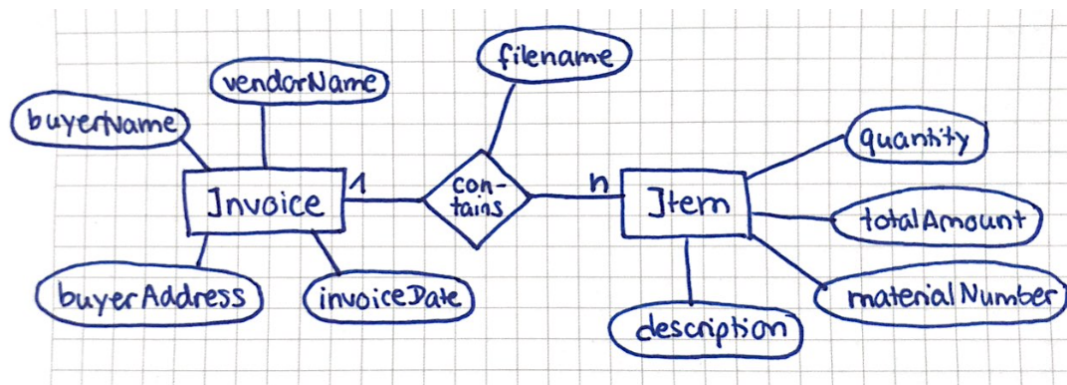


Figure 7.4: Entity Relationship Diagram for Invoice Documents

learning results are likely satisfactory. Impurities may be missing values, shifted columns, encoding errors or different data formats. 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.

7.2.1 Data processing and data wrangling

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

Storage 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 filesystem. 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 filesharing 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

Transformation The data is still available only in the form of JSON documents.

- transforming json documents into dataframe rows

7.2.2 Data cleaning

- removing stopwords from several languages
- removing numbers and interpunction
- tokenization

7.2.3 Feature Extraction and Feature Engineering

The majority of popular ML algorithms require the input of scalar, vector or matrix data. A form of ML models, which work with textual input will be discussed later in this section. But since many algorithms were not designed to work with textual data, a transformation is required before already existing algorithms can be applied. Several methods for representing text as mathematical object will be discussed in the following.

One-hot encoding

Bag of Words Model

Following three documents will be considered to explain the workings of the Bag of Words (BoW) model:

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

A corpus is transformed into the BoW representation in two steps.

Firstly, the vocabulary is determined. The vocabulary 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 represented as a matrix. The resulting matrix 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, the occurrences are counted. The times, how often the specific word is contained in the document is noted in the matrix at location ij .

	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

The result of vectorization are three vectors:

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

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

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

This vectorization method can be implemented with ease and in a computationally efficient manner. The BoW model allows for a very intuitive understanding of documents, since texts consisting of the same words are considered topically related.

One of the drawbacks 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 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 automobiles. 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 offence (d_3).

To conclude, the BoW model is a straightforward text representation method. Still, it fails to capture several aspects of the natural language.

Tf-Idf

The term frequency - inverse document frequency (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 Term Frequency A word t_i which occurs very frequently in one document is considered to describe a text very well. 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 would be considered equally important to each document. Of course, the word should be considered more important to d_3 , since it accounts for a larger share of the text. The measure resulting from both ideas is the term frequency:

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

2 Inverse Document Frequency A word t_i which occurs in a large number of documents does not describe one document well. Words occurring in many documents often are articles or pronouns (stopwords) 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.

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

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:

$$d_1 = [0 \ 0.707107 \ 0 \ 0 \ 0 \ 0.707107 \ 0 \ 0] \quad (7.1)$$

$$d_2 = [0.39798 \ 0 \ 0.39798 \ 0 \ 0.39798 \ 0.39798 \ 0.605349] \quad (7.2)$$

$$d_1 = \begin{bmatrix} 0 & 0.707107 & 0 & 0 & 0 & 0.707107 & 0 & 0 \end{bmatrix} d_2 = \begin{bmatrix} 0.39798 & 0 & 0.39798 & 0 & 0.397980 & 0.397980 & 0.605349 \end{bmatrix} d_3 = \begin{bmatrix} 0 & 0 & 0 & 0.7 \end{bmatrix}$$

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

Just as the BoW representation, TF-IDF suffers from high-dimensionality. The vectors contain one element for each word in the vocabulary, resulting in vectors which are inefficient to handle. Also, TF-IDF fails to represent the topical relationship between d_1 and d_3 .

Word Embeddings

Again, 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. Popular embeddings include word2vec, fasttext and doc2vec. Word2Vec is the original model and will be discussed in the following.

The word2vec model assumes that words appearing in a similar context are also similar to each other. There are two options for the input and output: the word w_i and its context c_i .

document	
1	The car repair was expensive.
2	My secretary did the ticket reservation.
3	I got a parking ticket yesterday.

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.

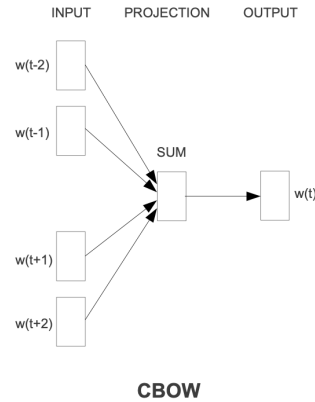


Figure 7.5: Continuous Bag of Words architecture with sliding window of size $C = 5$

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

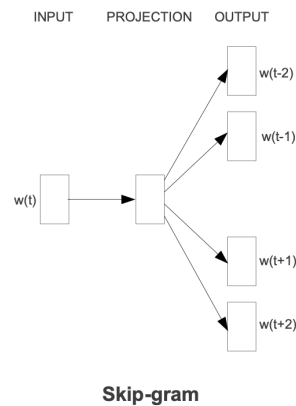


Figure 7.6: Skipgram architecture with sliding window of size $C = 5$

Negative Sampling Word2Vec uses either SKIPGram or CBOW.

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

Figure 7.7: Observations for training with skipgram architecture

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

8 Modelling

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 definition of similarity as well as different clustering algorithms are presented and contrasted in the following sections.

8.0.1 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.

Euclidean Distance The most intuitive and popular distance measure is the euclidean distance. This measure is applicable for vector spaces F^n with $n \in \mathbb{N}_0$. 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 [3] it is proven that with increasing dimensions, the distance between data points approaches an uniform value for all datapoints. 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, this distance measure is not suitable.

Dot Product

Cosine Distance

8.0.2 K-Means with Euclidean Distance

9 Evaluation of the result

9.1 Visualization

9.2 Measures

10 Conclusion and Outlook

10.1 Conclusion

10.2 Outlook

Literaturverzeichnis

- [1] *AI Overview Research*. <https://www.sap.com/products/artificial-intelligence/research.html>.
- [2] Andreas Schmitz. *Was ist SAP Leonardo?* 07/2017.
- [3] Beyer, K. et al. “When Is “Nearest Neighbor” Meaningful?” In: (), p. 19.
- [4] Chapman, P. et al. “CRISP-DM 1.0: Step-by-step Data Mining Guide”. In: 2000.
- [5] Chun, W. *Core Python Programming*. Vol. 1. Prentice Hall Professional, 2001.
- [6] Daniel Rutschmann. *The Journey from SAP Leonardo Machine Learning Foundation to SAP AI Core and SAP AI Launchpad*. <https://blogs.sap.com/2021/10/11/the-journey-from-sap-leonardo-machine-learning-foundation-to-sap-ai-core-and-sap-ai-launchpad/>. 10/2021.
- [7] Fayyad, U. “From Data Mining to Knowledge Discovery in Databases”. In: (), p. 18.
- [8] Gant, M. *Scrum and the Solo Dev*. 06/2019.
- [9] *Geschichte Der SAP*. <https://www.sap.com/germany/about/company/history/1972-1980.html>.
- [10] Gobinda G. Chowdhury. “Natural Language Processing”. In: *Annual Review of Information Science and Technology* 37.1 (01/2003), pp. 51–89.
- [11] *Juergen Mueller Biography*. <https://www.sap.com/about/company/leadership/juergen-mueller.html>. Company Website.
- [12] Koch, B. “The E-Invoicing Journey 2019-2025”. In: (), p. 7.
- [13] Saltz, J. *CRISP-DM Is Still the Most Popular Framework for Executing Data Science Projects*. 11/2020.
- [14] SME. *It’s Costing How Much? The Truth about Manual Invoice Processing*. 05/2018.
- [15] *Structured vs Unstructured Data 101: Top Guide*. <https://www.datamation.com/big-data/structured-vs-unstructured-data/>. 05/2021.
- [16] *Understanding Invoices*. <https://www.investopedia.com/terms/i/invoice.asp>.

- [17] *What Is the Purpose of an Invoice? / Invoicing Tips for Small Business*. <https://www.freshbooks.com/do-invoices-work>.