ML-based Information Extraction of Documents

using the example of Safety Datasheets (SDs) according to REACH-Regulation

Gregor Breitschopf	1802357
Gerrit Merz	1843958
Adrian Pucher	1835154

Forschungszentrum für Informatik (FZI)

Research Project: Data Science & Real-Time Big Data Analytics

Examiner:	Prof. Dr. York Sure-Vetter
Superviser:	Dr. Sinan Sen
Submitted on:	26. July 2019

Abstract

We examine how document element extraction can be automated, using the example of safety datasheets (SDs), stored in the Portable Document Format (PDF) from a big retail business. For this purpose, we provide a new benchmark data set with 527 SDs, labeled with 13 different elements of interest. We experiment with Word Embeddings, Hand-Crafted Features and a newly developed feature group called Structural Features to train 46 separate Random Forest Classifiers. The results show that context incorporation leads to improvements for all feature sets, significantly for Hand-Crafted Features and Word Embeddings. Without information about the context, we already achieve strong scores within all models containing our self-developed Structural Features, whereas in this case Hand-Crafted Features are not sufficient. Overall, the best results are obtained by combining all feature groups as well as context incorporation.

Table of Contents

List of Figures

List of Tables

1. Introduction

In the digital era, information is easily produced and accessible to everyone. Finding the right piece of data can therefore be a hardship and is a crucial skill if executed efficiently. With the amount of data that is produced each day, new text data is also skyrocketing (Raconteur, 2019). Not only individuals have to cope with this overload of information, but also companies. Especially the latter ones find themselves in an environment in which they are faced with a variety of different types of documents (e.g. product sheets of new articles need to be scanned, regulatory changes need to be obeyed and contracts need to be monitored). Keeping an eye on the relevant changes, for example, to notify the affected parties can be challenging, especially if, in most cases, only a certain piece of new information is desired. To date, information extraction out of text documents is sometimes still performed manually. Given the abundance of data and the constant renewal of these, this approach seems not only outdated, but also contains various negative implications. Not only is the manual extraction time-consuming, but the tediousness of the task itself also makes it very prone to error.[1] Obviously, the information extraction out of text documents does not yet meet the requirements of the digital era. Hence, an automated data extraction approach would be a self-evident next step.

The goal of this paper is twofold. On the one hand, we replicate an already existing Machine-Learning-based (ML-based) approach to extract information out of text documents of Chalkidis et al. (2017) to our use case. On the other hand, we also extend this approach by designing new features called Structural Features (ST) and incorporate them in our models.

In detail, we examine how document element extraction can be automated, using the example of safety datasheets (SDs), stored in the Portable Document Format (PDF) from a big retail business. For this purpose, we provide a new benchmark data set with 527 SDs, labeled with 13 different elements of interest. This data set can be used to test our SD extraction algorithms and train other models. To represent the PDF data in an appropriate format, we receive the horizontal and vertical location of each symbol. Due to the lack of existing labels in our data set, we annotate every token of interest with the according label using rule-based scripts. Finally, we train 46 separate Random Forest Classifiers on different feature combinations (consisting of the existing features (Hand-Crafted Features and Word Embeddings) from Chalkidis et al. (2017) and the new ST) for various label groups in order to assess which features work best for the extraction. Additionally, we assess how our models perform when context is incorporated by using a sliding window for the features of the preceding and succeeding tokens for each label similar to the approach of Chalkidis et al. (2017).

The results of our models show three major findings:

1) Context incorporation leads to improvements for all feature sets, significantly for Hand-Crafted Features and Word Embeddings.

[1] A possible error could occur in various ways for example by extracting the wrong information, misspelling or putting the right information in the wrong textbox.

2) In the case of no window, we already achieve strong scores within all models containing our self-developed Structural Features.
3) Without context incorporation, Hand-Crafted Features are not sufficient.

Overall, the main contributions of this paper are the following:

1) This paper focuses on automatic document information extraction using the example of SDs which is of high practical value. As far as we are concerned, to date this use case has not been examined at all.
2) This paper adds to the existing literature of automated information extraction by introducing a new type of feature set.
3) This paper shows that replicating an existing ML-based approach can also work for a different use case.
4) This paper highlights the number of possible applications for information extraction out of documents and their importance to finding appropriate solutions.

The rest of the paper is structured as follows: Chapter 2 focuses on related work. Chapter 3 explains our use case, depicts main elements of SDs and highlights the importance of certain elements. Based on these findings, chapter 4 introduces our own approach. The results obtained are presented in chapter 5 and discussed in chapter 6. The conclusion in chapter 7 briefly reviews the progress of the work, the insights gained and the relevance for future work. The code and all trained models are published on the university's internal Gitlab.[2]

[2] https://git.scc.kit.edu/ucvzt/ml-based-information-extraction-safety-datasheets

2. Related Work

The theoretical background of this paper rests mainly on the work of Chalkidis et al. (2017). In their paper, “Extracting Contract Elements”, they extract contract element types like involved parties, jurisdiction, effective date and start date from various kinds of contracts. For this purpose, they use a manually labeled data set with 3500 English contracts, in which the sentences have been divided and tokenized without considering the position of the characters within the documents. For the training and testing of their models, 4 feature groups are used:

1) Hand-Crafted Features, which are features that indicate the length of the token, other binary characteristics like upper/lower case words, as well as contract element specific information which are mapped with the help of regular expressions.
2) Part-of-Speech (POS) Features categorize tokens into different semantic groups. One possible group could be that the token stands for a country, another that the token expresses a date.
3) Word Embeddings represent a string in a unique vectorized format which cover semantic relations and make them a valid input format for the performed machine learning tasks. Chalkidis et al. (2017) use a vector with 200 dimensions. The POS Features as well as the Word Embeddings are pre-trained on an unlabeled data set using 750.000 contracts.
4) They also experiment with different sizes of sliding windows for context incorporation. A sliding window includes a certain number of tokens before and after the current token. The best performing window size in their experiments is 13, which means that the information of the immediate 6 words before and after the token is also used in the models.

For the testing of different combinations of the above-mentioned feature groups Chalkidis et al. (2017) use two ML-Models: Logistic Regression (LR) and Support Vector Machine (SVM). Their results show that the more features are incorporated into the models, the higher the scores (accuracy of the extraction). The second finding is that when comparing the models, SVM performs better than LR.
The approach of Chalkidis et al. (2017) serves us as orientation for our use case. Apart from their own approach which is to the best of our knowledge the latest and most suitable work for our specific application, they introduce more related recent text analytics work.
The following paragraph should serve as a short summary of their theoretical foundation and uses literature which is at the same time the fundament of our work since we replicate their approach and extend it with our own idea of Structural Features.
In the context of legal text analytics work, Indukuri & Krishna (2010) classify contract sentences as non-clauses or clauses by employing n-gram features and SVMs. However, their database consists of only 73 sentences. Curtotti & Mccreath (2010) experiment with only 30 Australian contracts. Using 40 Hand-Crafted Features, they employ various ML-based algorithms like decision trees and SVMs. With a single

multi-class classifier (combination of Random Forest and manually written tagging rules), the best results are obtained. In their paper, “Mining Business Contracts for Service Exceptions”, Gao & et al. (2012) experiment with a bigger database of 2647 contracts, but only to locate certain exception clauses. Looking at the broader context of automated text analytics work, Stranieri & Zeleznikow (2005), Francesconi et al. (2010) and Mencia (2009) use 181 texts to employ SVM and Hand-Crafted Features to identify titles and articles by segmenting French laws. Biagioli et al. (2005) locate paragraphs within Italian law with particular types of information (e.g. sanction, obligation) using SVM with bag-of-word features. Dozier et al. (2010) focus on trial documents in the US to detect jurisdictions, firms, courts, judges as well as attorneys using various kinds of features and SVM. In a first step, Quaresma & Gonçalves (2010) perform their experiments on 2714 agreements using SVM with TF-IDF bag-of-words to categorize European international agreements into topics. Afterwards, they employ manually crafted patterns operating on parse trees to retrieve document references, organizations, dates and locations.

3. The Task

3.1. Idea and Goal

All in all, the in section 2 presented papers mostly focus on classifying clauses, phrases or entire lines, using only small data sets. On top of that, none of the previous work – Chalkidis et al. (2017) excluded – considers Word Embeddings and none – Chalkidis et al. (2017) included – Structural Features.
In this paper, we deal with the previously expressed points of criticism

1) by using a much larger database for a new application,
2) by testing a different classifier (Random Forest),
3) and by experimenting with an additional new feature group called Structural Features.

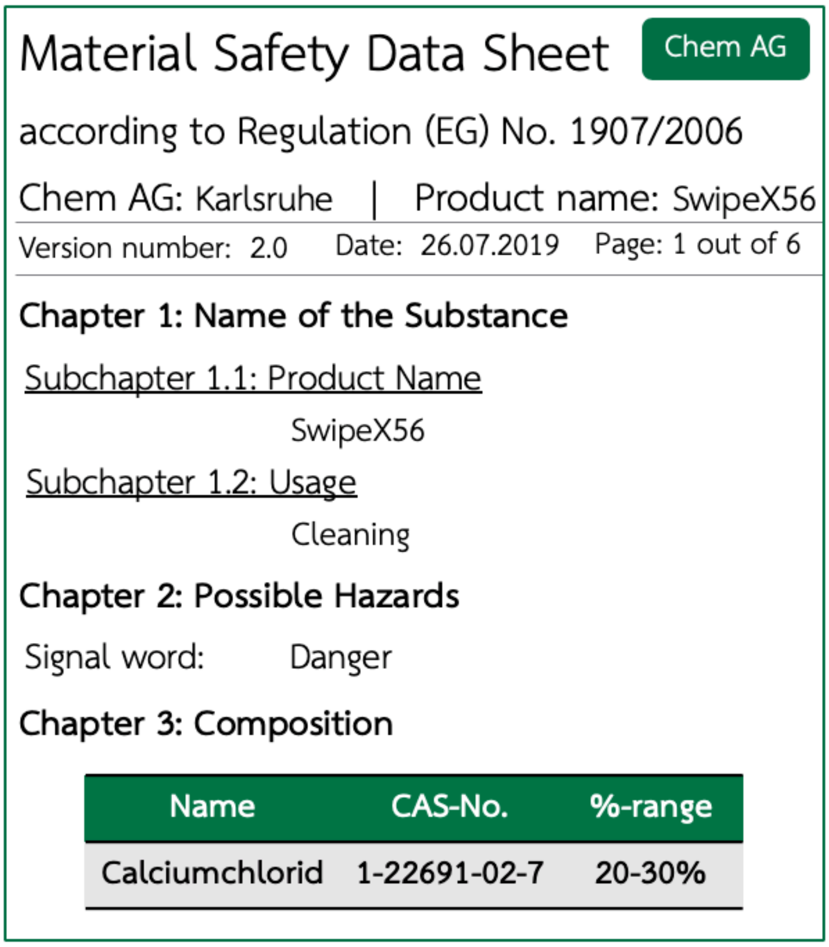
Thus, the goal of this paper is to replicate and extend the approach of Chalkidis et al. (2017) to our use case. By this means, we give our paper a clear framework as to what should be achieved.

3.2. Safety Datasheet Structure and Elements

In order to find a suitable use case, three criteria were of utmost importance to our later approach:

1) The documents needed to have a sufficient structure for the automated extraction.
2) One conclusion of the presented literature was that the database we train and test our models on ought to be sufficiently large.
3) When obtaining the documents, there should be no data protection concerns.

Therefore, we focus on safety datasheets (SDs) that are provided from a big retail business. SDs are documents that transmit safety-related information on substances and mixtures. They are needed if an Entity A sells a chemical/substance to an Entity B. SDs are intended to provide the professional user with the necessary data and handling recommendations when dealing with substances and mixtures in order to be able to take the necessary measures for the protection of health, safety at work and the environment. The legal basis for the preparation of SDs is specified in the REACH-Regulation. The regulation states that safety data sheets shall be divided into 16 sections in a specific order. In addition, certain subsections are required if there is more information about the chemical/substance available (VCI, 2008).
Figure 1 illustrates an exemplary structure of a SD, which serves as an abstract representation in the further course of this paper.



Material Safety Data Sheet Chem AG

according to Regulation (EG) No. 1907/2006

Chem AG: Karlsruhe | Product name: SwipeX56

Version number: 2.0 Date: 26.07.2019 Page: 1 out of 6

Chapter 1: Name of the Substance

Subchapter 1.1: Product Name

SwipeX56

Subchapter 1.2: Usage

Cleaning

Chapter 2: Possible Hazards

Signal word: Danger

Chapter 3: Composition

Name	CAS-No.	%-range
Calciumchlorid	1-22691-02-7	20-30%

Figure 1: Exemplary Structure of a Safety Datasheet

The number of pages a SD contains varies depending on the amount of information required for the transmission, however SDs are always structured the same way:

On top of every page, the SD contains information like the version number, the name and address of the company which provided the SD, the print date and the product. As illustrated in Figure 1, chapter 1 of a SD contains information about the substance and further subchapters like the usage of the product. Chapter 2 deals with possible hazards that emanate from the product. Chapter 3 gives information about the composition of the different kinds of chemicals, their equivalent numerical identifiers (CAS-No.) and their respective percentages within the product.

That said, SDs fulfill all our criteria for our approach. They are sufficiently structured, and the database is provided by a huge retail business, which dispels concerns about privacy breaches. On top of that, the automated information extraction of SDs is of high practical relevance, as SDs accompany the majority of products and flawed information extraction should be avoided at all events due to the uncertainty of chemical reactions.

In this paper, we focus only on the first three pages of a SD, since the chapters with the information of interest to us are located there. The following types of elements, illustrated in Table 1, should be extracted, when present:

Element	Description	Label type	Data type	Example
Chapter	Name of chapter	Labels with multiple tokens	String	Name of the Substance
Subchapter	Name of subchapter	Labels with multiple tokens	String	Product Name
Version No.	Version of SD	One-Token-Label	Numerical	2.0
Regulation	Indication of structure and content of SD	One-Token-Label	Numerical	1907/2006
Signal Word	Degree of danger arising from the product	One-Token-Label	String	Danger
Company	Specification of the sender of the SD	Labels with multiple tokens	String	Chem AG
Usage	Identification of usage of product	Labels with multiple tokens	String	Cleaning
Chemical	Triple consisting of name, Inter. Identifier and %	Triple	String/Numerical	Chloride, 1-22692-21-6, 30%
Date	Up-to-dateness of SD	One-Token-Label	Date	26.07.2019

Table 1: Specification of Elements to be Extracted

Chapter, subchapter:

Chapter and subchapter are essential parts of a SD. They are prescribed by the regulation and must exist. The labeling and the ML-extraction is useful to the sender of the SD to check if all relevant chapters and subchapters are included. Furthermore, extracting chapter and subchapter helps the receiving entity to identify where which type of information is located. The data type is string and consists of multiple tokens, e.g. *Name of the Substance*.

Version No.:

The *Version No.* reveals information about the version of publication of the document by the manufacturer. Again, in case of inquiries, referring to the same version of the SD makes coordination between the seller and the receiver of the chemical/substance a lot easier. The data type is numerical, e.g. 2.0.

Regulation:

The regulation indicates the structure and the content the SD must fulfill and contain. Hence, we are interested whether the SD is REACH-compliant. The data type can consist of numerical or/and string values, e.g. *1907/2006/EG* or *2015/830*.

Signal Word:

The signal is located in chapter 3 *Possible Hazards*. Extracting this information is another security precaution as it helps to identify which dangers are caused by the product. The data type is string and consist of one token, e.g. *Danger*.

Company:

The element *Company* specifies the sender of the document. In case of further inquiries, this information is very useful on whom to turn to. The data type is a string and can consist of multiple tokens, e.g. *Chem AG*.

Usage Pro/Con:

This element is especially relevant to the receiver of the SD. It helps to identify what the product should (and should not) be used for, making it a crucial and safety-relevant element. Additionally, this element helps to categorize different product names into the same usage category (possible substitutions) in case a certain chemical cannot be used for whatever reason. The data type is string and can consist of multiple tokens, e.g. *Inert gas for welding processes*. In the final data set this element is divided into the two sublabels *Usage Pro* and *Usage Con*.

Chemical (Name, CAS-No., %):

The element *Chemical* is a triple label which consists of the name of the chemical, its international numerical identifier and the proportion of the chemical in the product expressed as a percentage. We extract all this information as it is useful to know which substances are contained in the product and how much percentage they make up. The international identifier (CAS-No.) is extracted due to the fact that in some cases the name of the chemical is not uniform (e.g. different writing conventions are possible). The CAS-No., however, always specifies what chemical one is dealing with. Nevertheless, it is still interesting to extract the actual name of the chemical as it is not only readable but also interpretable by humans. The data types are string and numerical, e.g. *Calcium chloride, 1-226901-02-7, 20-30%.*

Date (print date, validation date, revision date and old version date):

In the case of SDs we face four different types of dates: *Print date, Validation date, Revision date and Old version date*. The data type is date and is a one-token label e.g. *26/07/2019*. The meaning of each type is self-explanatory and this element group is divided into four sublabels in the final dataset.

4. Our Approach

This chapters describes the idea of our information extraction approach using the example of SDs according to REACH-Regulation. It shows the original data set, the labeled benchmark data set as well as the feature groups we provide for the tokens of the data set. Eventually, we talk about our test-training splits as well as the models we implement.
The steps of our procedure are illustrated in Figure 2 which shall serve as an orientation throughout this chapter:

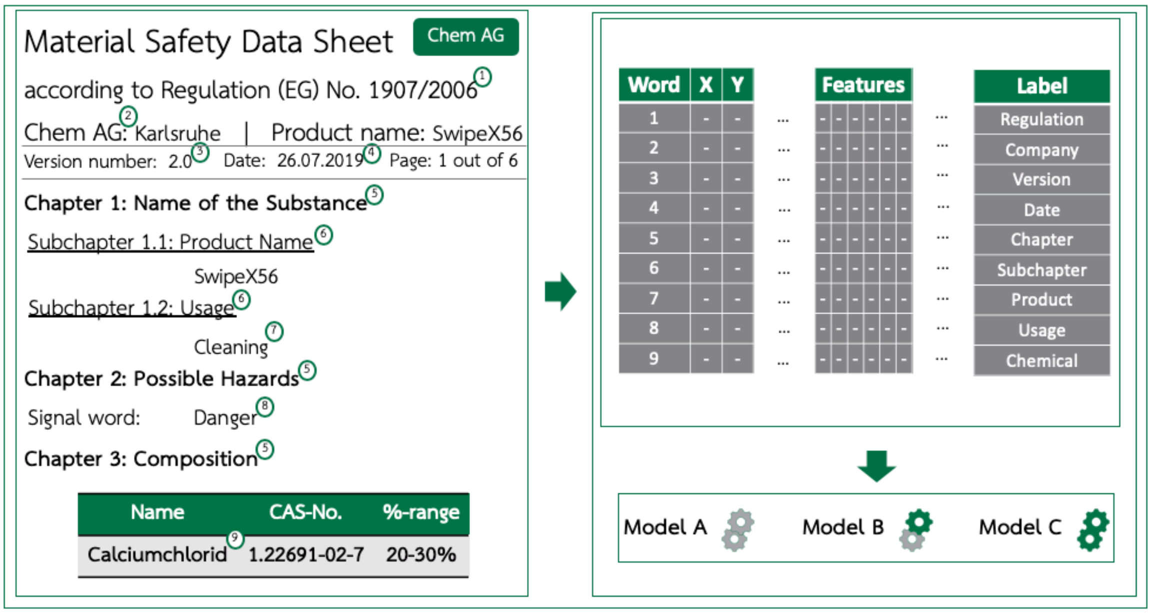

Material Safety Data Sheet
Chem AG
according to Regulation (EG) No. 1907/2006 ①
Chem AG: ② Karlsruhe | Product name: SwipeX56
Version number: 2.0 ③ Date: 26.07.2019 ④ Page: 1 out of 6
Chapter 1: Name of the Substance ⑤
Subchapter 1.1: Product Name ⑥
SwipeX56
Subchapter 1.2: Usage ⑥
Cleaning ⑦
Chapter 2: Possible Hazards ⑤
Signal word: Danger ⑧
Chapter 3: Composition ⑤
Name
CAS-No.
%-range
Calciumchlorid ⑨ 1.22691-02-7 20-30%
Word
X
Y
1 2 3 4 5 6 7 8 9
Features
Label
Regulation
Company
Version
Date
Chapter
Subchapter
Product
Usage
Chemical
Model A
Model B
Model C

Figure 2: Information Extraction Idea

4.1. Data Representation and Labeled Benchmark Data Set

In a first step, we need to preprocess our SDs from the PDF format into a machine-readable text format. The idea is to identify the x- and y-coordinates of every token for an accurate representation of the SDs in PDF format. Therefore, we use the Python library “PDFMiner”.[3] To sum it up, the script iterates through each SD, then through each page and finally inspects single characters. Then, the characters are combined to words and we are able to assign to each word the x-coordinate, y-coordinate as well as the font size and font type. Due to some irregularities in the representation, we take the average of the y-coordinates from words that are in the same line in a further step to ensure an accurate final representation of the documents in table format.

[3] https://pypi.org/project/pdfminer/

Sample size	Number of SDs
Size of initial sample	741
Duplicates	-21
Importing problems	-17
Exporting problems	-1
Formatting problems	-175
Final data set	**527**

Table 2: Data Basis

As Table 2 illustrates, the initial size of our sample contains 741 SDs. Throughout the data preprocessing, we lose the vast majority of our data basis due to formatting problems (e.g. the lack of embedded texts). Thus, our final data set consists of 527 SDs.

Having preprocessed our SDs, we then label the tokens of interest, illustrated by the numbers behind each token on the left side of Figure 2 and the corresponding table "Label" on the right side of Figure 2. As Table 1 in subchapter 3.2 already showed, we deal with different kinds of labels regarding the number of tokens, their data types as well as with one special case, *Chemica*l which is a triple. In contrast to Chalkidis et al. (2017), we do not label manually. Instead we separate our labeling task into 4 different cases, depicted in Table 3, to solve the labeling programmatically (rule-based):

Case	Elements	Token type	Idea
1	Chapters, Subchapters, Company, Usage	Labels with multiple tokens	Define Start and Stop Words
2	Regulation	One-Token-Label	Regular Expression
3	Version-No., Signal Word, Dates	One-Token-Label	Catch Words
4	Name, CAS-No., %	Triple	Regular Expression, List

Table 3: Label Cases

1) In the case of multiple tokens, we define manually for every element a list with start words and a list with stop words. Roughly speaking, the script goes through our tokens and starts labeling if it detects a certain start word or certain consecutive start words and labels until it detects the corresponding stop words. In order to achieve a high accuracy for this labeling method, several exception rules and stop conditions have been defined for each type of element within this group.
2) In case 2, we use regular expressions for the one-token-label *Regulation* as the buildup of each regulation number follows a pattern.

3) The remaining one-token-labels are labeled with the help of “catch words”. Certain elements follow subsequently after specific words, such as *Version No.* or *Print Date*. If the word (or multiple words) before the token of interest is (are) in our catch word list, we label the token.
4) The labeling of the chemicals required multiple steps. In a first step, we define regular expressions for the CAS-No., then label the CAS-No. and save it in a list. Once that is done, we create a dictionary to map the CAS-Nos. to the chemical names using an online available database[4] for chemicals. Finally, if a token is in our CAS list, our algorithm goes through the data and inspects the context around the CAS-No. and labels the information about the percentage of the chemical as well as the chemical name.[5] To do so, it uses the corresponding name for the chemical out of the previously created dictionary and looks for similar tokens or part of tokens that appear in the chemical name and labels them accordingly. Thus, different naming conventions and also differently used languages for the name of the chemical can be identified and labeled.

Table 4 shows the number of tokens per label (for the test and training part of the data set combined) in our labeled benchmark data set, containing 527 SDs. All SDs in the labeled benchmark data set are in German. Further discussed below, we also provide Hand-Crafted Features, Structural Features and Word Embeddings per token.

Label	Tokens	Label	Tokens
Chapter	23.501	Usage Con	3.680
Subchapter	24.318	Chemical	6.133
Version Nr.	1.283	Print Date	443
Regulation	1.113	Revision Date	435
Signal Word	439	Validation Date	15
Company	3.081	Old Version Date	59
Usage Pro	421	No Label	473.048
Sum	**537.969**		

Table 4: Statistics of the labeled benchmark data set

Table 4 shows that our labeled benchmark data set is very unbalanced, meaning we have a quite small number of true positives (64.921). Keeping that in mind when discussing the results of our approach is

[4] https://comptox.epa.gov/dashboard/dsstoxdb/batch_search
[5] The script descriptions are kept very simple. In reality, several exceptions had to be thrown to cover all the cases (e.g. tokens that go over several lines, tokens with different formats etc.).

crucial. Finally, for the sake of simplicity the labels are combined to three groups based on individual characteristics:[6]

1) Label Group: Chapter, Subchapter, Version-No., Regulation, Signal Word, Company, Usage
2) Chemical: Triple
3) Date: Print Date, Revision Date, Validation Date, Old Version Date

4.2. Feature Groups

Before we perform the ML-models we provide each token with different features:

1) The Hand-Crafted Features are similar to the ones Chalkidis et al. (2017) are using. In our case, the first 7 features are binary features and contain information about the length of the token (if length of token is between 1-2 characters, first feature is 1, if length of token is between 3-4 characters, second feature is 1 and so on). The next 3 binary features are about the style of character and indicate if the token contains only upper, lower or mixed (both) characters. 4 binary features suggest the type of characters: token contains a digit, is numerical, is a special character or a stop word. The last 4 binary features that belong to the Hand-Crafted Features indicate the date in question.
2) Additionally, we provide for the three groups our self-developed Structural Features. The idea of the Structural Features is that we want to exactly locate the position of each labeled token on page 1-3, but also within the page of a SD. The first 3 features give information about the x- and y-coordinate and if the token is bold (binary feature). Furthermore, we divide each page into a grid consisting of 32 cells, which is interpreted as 32 binary features indicating whether a token is located in this segment or not. This is followed by 4 binary features indicating if the token is on page 1,2 or 3 and if the token goes over more than one line. Finally, we also replicate Chalkidis et al. (2017)'s method of label specific features for all date specific elements to test if specialized features can boost extraction performance more than features which are generated for all types of labels. Therefore, we introduce 4 binary features for each date type to mark whether a string is part of the date specification or not, e.g. *Print date*. All in all, this leads to 39 (+4 date specific) features.
3) Chalkidis et al. (2017) describe Word Embeddings as "dense real-valued vectors, each representing a particular vocabulary word as a point in a high-dimensional vector space, such that vectors of words with similar morpho-syntactic and/or semantic properties will be close in the high-dimensional space." In their work they generate a new contract specific 200-dimensional word embedding based on 750.000 contracts. Due to our limited database, we were not able to train a word embedding based on safety datasheets, however we used the pretrained German

[6] The chemical labels take the order of the occurring chemicals into account, which accounts for up to 90 labels. Hence, an individual model is trained for this multi-class classification. Furthermore, it makes sense to combine all date specific information in one group. The remaining labels are assigned to the Label Group.

word embedding library fastText[7], developed by the AI Research Lab from Facebook, to represent each token in a 300-dimensional word representation[8].

4) Eventually, we use a sliding window of size 13 in case of the Hand-Crafted and Structural Features and a sliding window of size 3 in case of Word Embeddings. This means that for each token that is classified we equipped the 6 (1) word(s) before and the 6 (1) word(s) after the token with the above-mentioned features as well. Chalkidis et al. (2017) come to the conclusion that a window size of 13 works best for the context of their dataset. As we want to focus on the interaction of the introduced feature groups in our paper, we adopt this window size. In the case of the Word Embeddings we reduce the size of our window to 3, thus keeping the dimensions for our feature-set on a computationally efficient level.

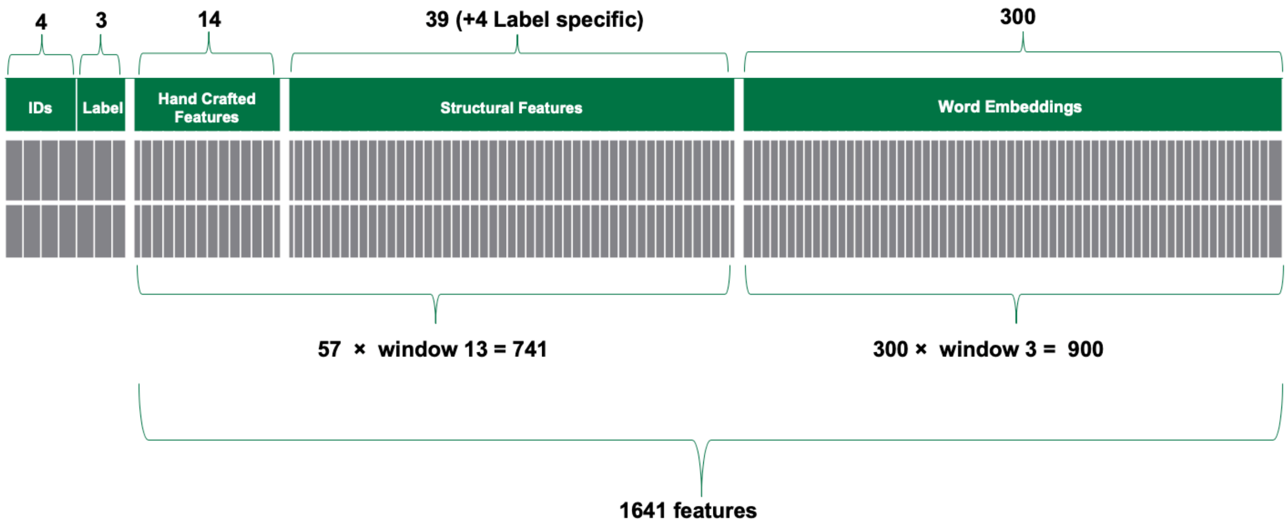


Figure 3: Dimensions of Data Set

To conclude, our training data set consists of 1648 columns for each token. As figure 5 depicts, the first 4 columns serve as orientation and the next 3 represent the grouped labels as discussed in subchapter 4.1. 14 columns consist of Hand-Crafted Features, 43 (counting the label specific features for date as well) columns consists of Structural Features and 300 columns represent our Word Embeddings. The number of features increases to 1641 by including the sliding windows (13 for Hand-Crafted Features and Structural Features, 3 for Word Embeddings)

4.3. Element Extraction Methods

After representing the SDs in a suitable data format, labeling that data by using manually written rules, and generating features that serve the purpose of Natural Language Processing (NLP), the dataset is ready to get fed into machine learning-based classifiers. The extraction of information is solely based on ML-models. Manually written rules are only used for the labeling of the data.

[7] https://fasttext.cc/docs/en/crawl-vectors.html

[8] Unknown terms are represented with a null vector.

Since the goal of the model is to classify natural language tokens into predefined entities, such as the *Version no*. or the *Usage of the product*, classifier ML models can be used. Including the context of the tokens into the feature set by using a sliding window, classifiers are suitable for natural language processing problems, specifically Named-Entity-Recognition (NER) problems. Based on the results of Chalkidis et al. (2017), a Support Vector Machine (SVM), a Logistic Regression (LR) model and a Random Forest (RF) model have been tested and the RF model has proven to be the most suitable one for the purpose of this paper and the available computing power.

RF is an approach, which combines several randomized decision trees and aggregates their predictions by averaging. Besides being very accurate, the main reason why RF serves our purpose is that it handles efficiently datasets with high dimensions (Biau & Scornet (2016), p.198). That is crucial, giving the size of the feature set (1641 features). Furthermore, it can handle unbalanced data by adjusting the error rate for different classes depending on the size of the class (Biau & Scornet (2016), p.223). In our case, 88% (473048) of all tokens (537969) belong to the class with no label, leading to an unbalanced dataset.

In order to train the RF model a regular 0.25 train test split has been conducted, separating 25% of the SDs into the test group and 75% of the SDs into the training group of the dataset. It is important to mention that the split has been conducted on SD level, ensuring that either all tokens of a SD are in the test or all the tokens are in the training dataset. The search for the right number of trees revealed 25 to be the most suitable number. Figure 4 shows that more than 25 trees would not affect the accuracy of the model significantly but lead to substantially higher computational effort:

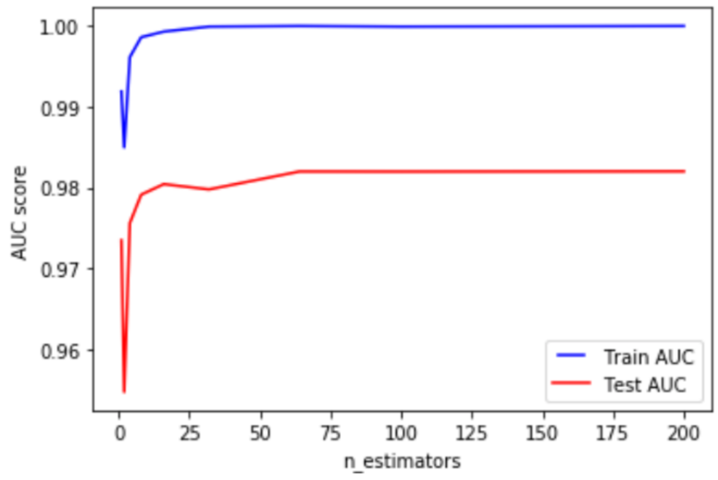


Figure 4: Identifying the Right Amount of Trees

The usage of the three different feature groups (HC: Hand-Crafted Features, ST: Structural Features and WE: Word Embeddings) are tested in all possible combinations, leading to 7 different feature set combinations (HC, ST, WE, HC+ST, HC+WE, ST+WE, HC+ST+WE), Additionally, the impact of including the sliding window of 13 (3 for WE) on the results is examined. The most common performance metrics for NLP classification problems are used on the different labels:

1. Precision: How many of the tokens that are classified as a certain element were correctly classified?

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

2. Recall: How many of all the tokens that belong to the certain element were identified?

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

3. F1-Score: Harmonic mean of Precision and Recall.

$$Precision = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall}$$

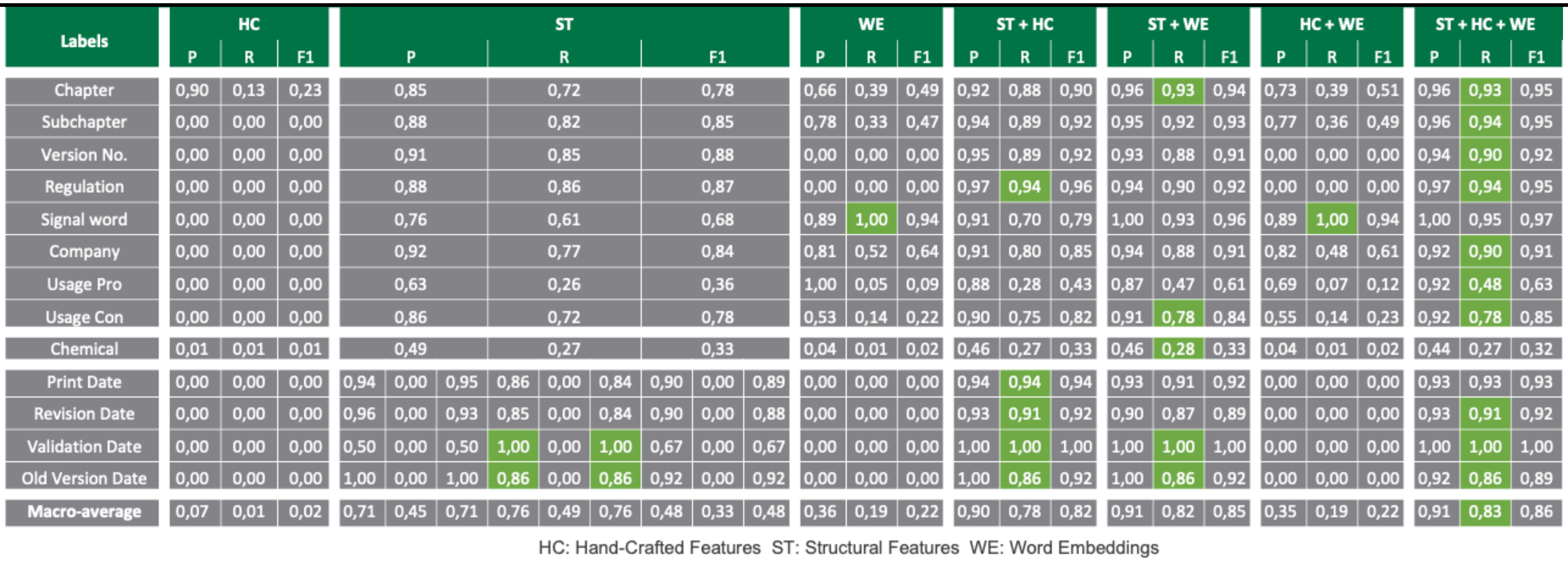
5. Results

Table 5 presents the results without context incorporation and table 6 with. The tables are structured as follows: In the left column, we see the different labels grouped into the *Label Group*, the *Chemical triple* and the *Date label*. The green highlighted headings show the results of Recall, Precision and F1-Score for the respective labels when the different feature groups are given into our model. Due to the high imbalance in our data set, the high Precision scores could be misleading. Therefore, we focus on the Recall and highlight the highest score(s) within each row. Finally, we compute the macro-average, which treats all classes equally and takes the average.

Starting with the results of the feature combinations without sliding window, it is obvious that in the case of no context incorporation, HC are not sufficient. A possible interpretation could be that as HC gives only word-specific meta-information like the string length, style and type of character. Due to that, it is impossible for our model to conclude anything solely from this information. The same goes for the *Chemical triple* as well as the *Dates*.
Our self-developed ST show a very strong performance overall. Only the Recall score of *Usage pro* (0.26) is an outlier which is a label with high variance in its length and positioning. The *Chemical triple* has a slightly better score (0.27). This is no surprise due to the triple combination which is hard to detect. Regarding the ST, as described in chapter 4.2 the date labels are enriched with specific features. Therefore, the order of the subcolums is the following: Normal Structural Features, Date Specific Features and the combination of both presented in the table from left to right, respectively. The addition of specific features seems to have no significant influence on the results in this case, which is clearly shown by the Date Specific Features only column. However, this result is not surprising, because these features are just true for the preceding context words which are unconsidered in the case of the no window scenario. Overall, the macro-average of 0.76 for the scenario with all features for *Date* is strong.
The third feature group, Word Embeddings, shows solid results for vocabulary words. To recall, word embeddings represent words as numeric vector. Therefore, data that is not included in the vocabulary (*Version No., Regulation, Dates, Chemicals*) have scores of 0, whereas unique keywords are found perfectly (Recall of 1) by our model. Labels with a high variance (*Chapter, Subchapter etc.*) reach only average scores. Subsequently, the macro-average when using Word Embeddings does not perform well.
If we combine the ST with the HC, the scores improve significantly. It seems that our ST help to give the information derived from the HC a purpose. In the case of detecting the *Chemicals*, the score does not change at all (0.27) compared when only the ST are applied.
Looking at the combination of our self-developed ST and WE, the scores perform significantly better than in case of solely WE. This applies especially to the words with high variance in the WE. With this combination we achieve the highest score (0.28) with the label *Chemical*. Overall, the macro-average of our Recall reaches 0.82, representing the second-best result.

The combined feature groups of HC and WE do not have a real impact on the scores. This is no surprise as we have already seen that both feature groups perform badly on their own.
Finally, the combination of all feature groups, ST, HC and WE, leads to the best results (macro-average score: 0.83) in the case of no context incorporation.

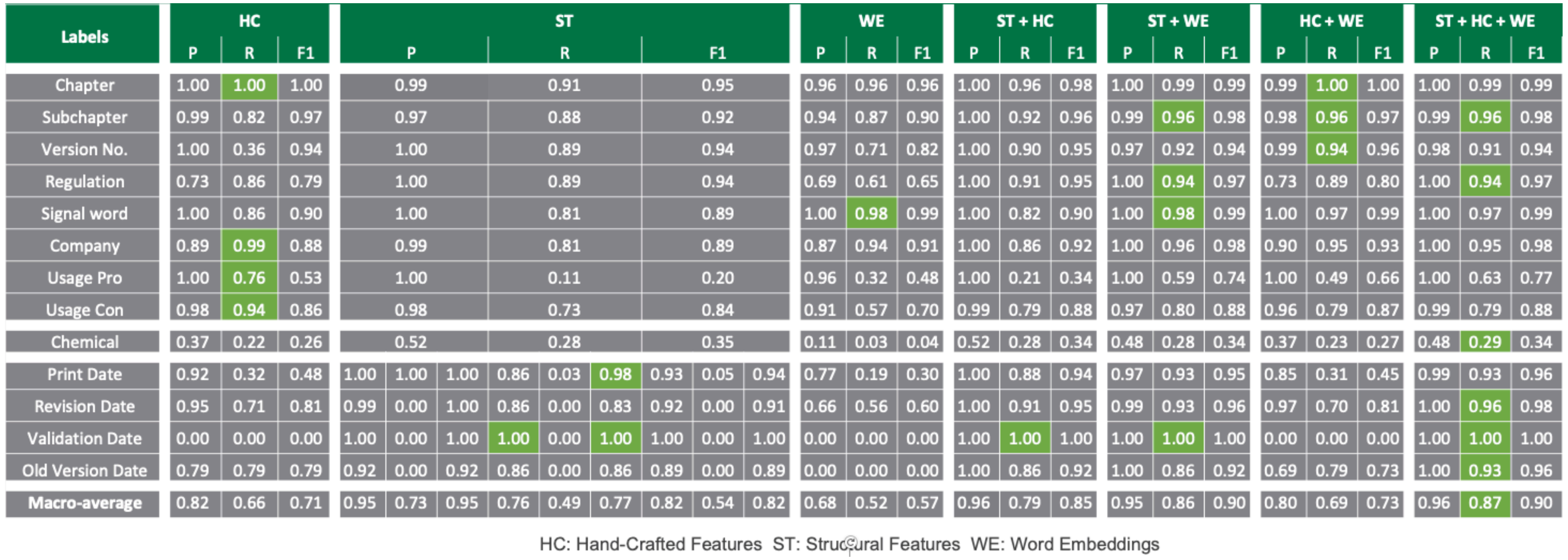
In the case of no sliding window, HC are not useful at all. However, if we take context into account, we achieve a huge improvement. This might be due to the fact that context incorporation allows to use the information of the words before and after our labels which improves the predictive power. This finding is highly interesting, because it seems so that these actually very general features can produce distinct patterns, which help to identify the requested information. The highly underrepresented *Validation date* is not detected at all.
The context incorporation regarding our self-developed ST achieves better scores than before. This seems reasonable as we extract relatively short text information which are often located in a similar position (same line, below each other etc.) resulting in the same grid. Therefore, there will not be a lot of repeating patterns for the predecessors and successors, which can be used to identify the tokens, but in general this also resulted in better scores.
WE now perform significantly better. Especially the ones that are labels with high variance (*Chapter, Subchapter etc.*) show significant improvements, but also the labels that are not in the vocabulary of the Word Embedding (*Version No., Regulation, Dates, Chemicals*) reach better scores. A possible explanation for this observation could be that "catch words" are now taken into account.
The addition of ST to HC (WE) does not really lead to a score improvement. The macro-averages change from 0.78 to 0.79 (0.82 to 0.86).
Looking at the combination of HC and WE, we experience huge improvements, especially in the case of *Version No.* (from 0 to 0.96) and *Regulation* (from 0 to 0.89). The context incorporation has already boosted HC and WE on its own. The combination, however, seems to benefit from the sliding window even more. The improvements are also reflected by the macro-average which goes up by 0.50 points.
If all the feature groups and sliding window are taken into account, we reach the strongest results. However, the improvements are marginal as there was not much space left to a perfect score. Nevertheless, context incorporation leads to the best overall results with an outperformance of 0.04 (from 0.83 to 0.87).
In summary, if context is taken into account, the best results are obtained within all feature groups, significant improvements are achieved for HC and WE. However, without sliding window, we already obtain very strong results with our self-developed ST and with WE in the case the words are in the vocabulary. Broken down into individual feature groups, the label group performs better, the more features we give our model whereas dates and the chemical triple are hardly detected. As mentioned at the beginning, a single interpretation of the Precision scores could be misleading. However, it is worth to note that in the case of high Recall scores we also achieve relatively high Precision scores overall which means that most of the detected elements have been detected correctly.



Labels	HC			ST									WE			ST + HC			ST + WE			HC + WE			ST + HC + WE		
	P	R	F1	P			R			F1			P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Chapter	0,90	0,13	0,23	0,85			0,72			0,78			0,66	0,39	0,49	0,92	0,88	0,90	0,96	0,93	0,94	0,73	0,39	0,51	0,96	0,93	0,95
Subchapter	0,00	0,00	0,00	0,88			0,82			0,85			0,78	0,33	0,47	0,94	0,89	0,92	0,95	0,92	0,93	0,77	0,36	0,49	0,96	0,94	0,95
Version No.	0,00	0,00	0,00	0,91			0,85			0,88			0,00	0,00	0,00	0,95	0,89	0,92	0,93	0,88	0,91	0,00	0,00	0,00	0,94	0,90	0,92
Regulation	0,00	0,00	0,00	0,88			0,86			0,87			0,00	0,00	0,00	0,97	0,94	0,96	0,94	0,90	0,92	0,00	0,00	0,00	0,97	0,94	0,95
Signal word	0,00	0,00	0,00	0,76			0,61			0,68			0,89	1,00	0,94	0,91	0,70	0,79	1,00	0,93	0,96	0,89	1,00	0,94	1,00	0,95	0,97
Company	0,00	0,00	0,00	0,92			0,77			0,84			0,81	0,52	0,64	0,91	0,80	0,85	0,94	0,88	0,91	0,82	0,48	0,61	0,92	0,90	0,91
Usage Pro	0,00	0,00	0,00	0,63			0,26			0,36			1,00	0,05	0,09	0,88	0,28	0,43	0,87	0,47	0,61	0,69	0,07	0,12	0,92	0,48	0,63
Usage Con	0,00	0,00	0,00	0,86			0,72			0,78			0,53	0,14	0,22	0,90	0,75	0,82	0,91	0,78	0,84	0,55	0,14	0,23	0,92	0,78	0,85
Chemical	0,01	0,01	0,01	0,49			0,27			0,33			0,04	0,01	0,02	0,46	0,27	0,33	0,46	0,28	0,33	0,04	0,01	0,02	0,44	0,27	0,32
Print Date	0,00	0,00	0,00	0,94	0,00	0,95	0,86	0,00	0,84	0,90	0,00	0,89	0,00	0,00	0,00	0,94	0,94	0,94	0,93	0,91	0,92	0,00	0,00	0,00	0,93	0,93	0,93
Revision Date	0,00	0,00	0,00	0,96	0,00	0,93	0,85	0,00	0,84	0,90	0,00	0,88	0,00	0,00	0,00	0,93	0,91	0,92	0,90	0,87	0,89	0,00	0,00	0,00	0,93	0,91	0,92
Validation Date	0,00	0,00	0,00	0,50	0,00	0,50	1,00	0,00	1,00	0,67	0,00	0,67	0,00	0,00	0,00	1,00	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,00	1,00	1,00	1,00
Old Version Date	0,00	0,00	0,00	1,00	0,00	1,00	0,86	0,00	0,86	0,92	0,00	0,92	0,00	0,00	0,00	1,00	0,86	0,92	1,00	0,86	0,92	0,00	0,00	0,00	0,92	0,86	0,89
Macro-average	0,07	0,01	0,02	0,71	0,45	0,71	0,76	0,49	0,76	0,48	0,33	0,48	0,36	0,19	0,22	0,90	0,78	0,82	0,91	0,82	0,85	0,35	0,19	0,22	0,91	0,83	0,86

HC: Hand-Crafted Features ST: Structural Features WE: Word Embeddings

Table 5: Results – Without Sliding Window



Labels	HC			ST									WE			ST + HC			ST + WE			HC + WE			ST + HC + WE		
	P	R	F1	P			R			F1			P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Chapter	1.00	1.00	1.00	0.99			0.91			0.95			0.96	0.96	0.96	1.00	0.96	0.98	1.00	0.99	0.99	0.99	1.00	1.00	1.00	0.99	0.99
Subchapter	0.99	0.82	0.97	0.97			0.88			0.92			0.94	0.87	0.90	1.00	0.92	0.96	0.99	0.96	0.98	0.98	0.96	0.97	0.99	0.96	0.98
Version No.	1.00	0.36	0.94	1.00			0.89			0.94			0.97	0.71	0.82	1.00	0.90	0.95	0.97	0.92	0.94	0.99	0.94	0.96	0.98	0.91	0.94
Regulation	0.73	0.86	0.79	1.00			0.89			0.94			0.69	0.61	0.65	1.00	0.91	0.95	1.00	0.94	0.97	0.73	0.89	0.80	1.00	0.94	0.97
Signal word	1.00	0.86	0.90	1.00			0.81			0.89			1.00	0.98	0.99	1.00	0.82	0.90	1.00	0.98	0.99	1.00	0.97	0.99	1.00	0.97	0.99
Company	0.89	0.99	0.88	0.99			0.81			0.89			0.87	0.94	0.91	1.00	0.86	0.92	1.00	0.96	0.98	0.90	0.95	0.93	1.00	0.95	0.98
Usage Pro	1.00	0.76	0.53	1.00			0.11			0.20			0.96	0.32	0.48	1.00	0.21	0.34	1.00	0.59	0.74	1.00	0.49	0.66	1.00	0.63	0.77
Usage Con	0.98	0.94	0.86	0.98			0.73			0.84			0.91	0.57	0.70	0.99	0.79	0.88	0.97	0.80	0.88	0.96	0.79	0.87	0.99	0.79	0.88
Chemical	0.37	0.22	0.26	0.52			0.28			0.35			0.11	0.03	0.04	0.52	0.28	0.34	0.48	0.28	0.34	0.37	0.23	0.27	0.48	0.29	0.34
Print Date	0.92	0.32	0.48	1.00	1.00	1.00	0.86	0.03	0.98	0.93	0.05	0.94	0.77	0.19	0.30	1.00	0.88	0.94	0.97	0.93	0.95	0.85	0.31	0.45	0.99	0.93	0.96
Revision Date	0.95	0.71	0.81	0.99	0.00	1.00	0.86	0.00	0.83	0.92	0.00	0.91	0.66	0.56	0.60	1.00	0.91	0.95	0.99	0.93	0.96	0.97	0.70	0.81	1.00	0.96	0.98
Validation Date	0.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00	1.00
Old Version Date	0.79	0.79	0.79	0.92	0.00	0.92	0.86	0.00	0.86	0.89	0.00	0.89	0.00	0.00	0.00	1.00	0.86	0.92	1.00	0.86	0.92	0.69	0.79	0.73	1.00	0.93	0.96
Macro-average	0.82	0.66	0.71	0.95	0.73	0.95	0.76	0.49	0.77	0.82	0.54	0.82	0.68	0.52	0.57	0.96	0.79	0.85	0.95	0.86	0.90	0.80	0.69	0.73	0.96	0.87	0.90

HC: Hand-Crafted Features ST: Structural Features WE: Word Embeddings

Table 6: Results – Sliding Window

6. Discussion

The overall performance of the models in predicting the type of elements is very high. The results of the models without using a sliding window for the context of words show that the location of a token indicated by the Structural Features is already a good identifier for the token's class. Information about the token itself, for instance whether it is a digit, how many characters it contains, or its representation through Word Embeddings become only a good identifier once the context is incorporated into the models using the sliding window Features.

The results for the label group *Chemical triple* show that identifying not only the occurring chemicals together with their CAS No. and Percentage No., but also their order is difficult when using the described feature sets. This is suspected, since the order of the occurring chemicals leads to more than 90 labels for certain SDs.

While looking at those findings, multiple points need to be taken into account:

Firstly, all SDs have very similar content determined by the REACH-Regulation. For instance, the wording of the chapters and subchapters – if present - is predefined. Therefore, the occurrence of certain words in a specific order could have been memorized by the models. Additionally, information such as the version number or the printing date is often mentioned at the same location. In our use case of extracting information from SDs, these are all element-specific indicators that shall be leveraged within our models, since information from new SDs with a similar structure shall be extracted. However, extracting information with these models out of other types of documents might lead to significantly lower performance scores, given the different structure of the document.

Secondly, in contrast to Chalkidis et. al (2017), no context-specific Word Embeddings have been trained, given the amount of data available. We use the library fastText provided by Facebook for the Word Embeddings. The settings of fastText represent a word as a zero vector if the word is not in the vocabulary of the library.

Thirdly, our approach is strongly paper-oriented. Using computationally expensive models like SVM, LR as well as a sliding window and applying them to a different use case, Chalkidis et al. (2017), achieve an average Recall score out of all tested features and model combinations of 0.80, whereas we reach 0.71 with context incorporation (0.49 without). This seems right, taking into account the various outliers (score 0) in case of the *Date label* and *Chemical triple*.

7. Conclusion and Future Work

In this paper, we examine how document element extraction can be automated, using the example of safety datasheets (SDs), stored in the Portable Document Format (PDF) from a big retail business. For this purpose, we provide a new benchmark data set with 527 SDs, labeled with 13 different elements of interest. Due to the lack of existing labels in our data set, we annotate every token of interest with the according label using rule-based scripts. Then, we provide Hand-Crafted Features, Word Embeddings and self-developed Structural Features. We use these features to perform 46 separate Random Forest Classifier on different feature combinations in order to assess which features work best for the extraction. Additionally, we assess how our models perform when context is incorporated by using a sliding window of the features of the preceding and succeeding tokens for each label.

The results show that context incorporation leads to improvements for all feature sets, significantly for Hand-Crafted Features and Word Embeddings. In the case of no window, we already achieve strong scores within all models containing our self-developed Structural Features. Without context incorporation we see that Hand-Crafted Features are not sufficient.

This paper contributes to the existing literature by focusing on a highly practical and yet not considered information extraction approach. Secondly, it adds to the literature of automated information extraction by introducing a new type of feature set. Thirdly, this paper is strongly oriented to the approach of Chalkidis et al. (2017) and thereby shows that an existing ML-based approach can also work for a different use case. Finally, the seminar paper also shows the multitude of possible applications for automated information extraction and the importance of finding efficient solutions.

All in all, we regard the results of this paper as a useful orientation for future work to experiment with more balanced data sets using our models or with more complex classifiers (SVM, LR or recurrent/convolutional neural networks) using the data that we provide.

List of References

Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., & Soria, C. (2005). *Automatic Semantics Extraction in Law Documents.* Bologna, Italy: In Proceedings of the 10th International Conference on Artificial Intelligence and Law.

Biau, G., & Scornet, E. (2016). *A random forest guided tour.* Sociedad de Estadística e Investigación Operativa.

Chalkidis, I., Androutsopoulos, I., & Michos, A. (2017). *Extracting Contract Elements.* London, UK: The 16th International Conference on Artificial Intelligence and Law.

Curtotti, M., & Mccreath, E. (2010). *Corpus Based Classification of Text in Australian Contracts.* Melbourne, Australia: In Proceedings of the Australasian Language Technology Association Workshop.

Dozier, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., & Wudali, R. (2010). *Named Entity Recognition and Resolution in Legal Text.* Springer.

Francesconi, E., Montemagni, S., Peters, W., & Tiscornia, D. (2010). *Semantic Processing of Legal Texts.* Springer.

Gao, X., Singh, M., & Mehra, P. (2012). *Mining Business Contracts for Service Exceptions.* IEEE Transactions on Services Computing 5.

Indukuri, K., & Krishna, P. (2010). *Mining e-Contract Documents to Classify Clauses.* Bangalore, India,: In Proceedings of the 3rd Annual ACM Bangalore Conference.

Mencia, E. (2009). *Segmentation of legal documents.* Barcelona, Spain: In Proceedings of the 12th International Conference on Artificial Intelligence and Law.

Quaresma, P., & Gonçalves, T. (2010). *Using Linguistic Information and Machine Learning Techniques to Identify Entities from Juridical Documents.* Springer.

Raconteur. (2019, Juli 25). *www.raconteur.net.* Retrieved from http://res.cloudinary.com/yumyoshojin/image/upload/v1/pdf/future-data-2019.pdf

Stranieri, A., & Zeleznikow, J. (2005). *Knowledge Discovery from Legal Databases.* Springer.

United States Environmental Protection Agency. (2019, 07 25). Retrieved from https://comptox.epa.gov/dashboard/dsstoxdb/batch_search

VCI. (2008). *Leitfaden Sicherheitsdatenblatt mit Hinweisen zur Einstufung und Kennzeichnung mit aktuellen Ergänzungen gem. REACH-Verordnung.* Verband der chemischen Industrie e.V.