DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Computer-aided Analysis of Privacy Policies: Extraction of Data Subject Rights According to GDPR

Sabrina Heinrich

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Computer-aided Analysis of Privacy Policies: Extraction of Data Subject Rights According to GDPR

Computergestützte Analyse von Datenschutzerklärungen: Extraktion von Rechten der betroffenen Person nach DSGVO

Author:	Sabrina Heinrich
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ingo Glaser
Submission Date:	August 15, 2019

I confirm that this master's thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, August 15, 2019 Sabrina Heinrich

Acknowledgments

Many thanks goes to the Software Engineering for Business Information Systems chair of the TUM, especially to Prof. Dr. Florian Matthes and my advisor Ingo Glaser, for giving me the opportunity to write this thesis and work on this particular topic. I want to thank Ingo Glaser specifically for guiding and supporting me throughout the whole process and providing me with valuable feedback.

Furthermore, I want to thank my family and all my friends who supported me continuously. Additionally, I would like to offer my special thanks to Jan-Philipp for his enduring and unwavering support.

Abstract

Nowadays, the internet plays an important role in everyday life. The provided services create and collect a huge amount of data. In 2016, the General Data Protection Regulation (GDPR) has been implemented, regulating the collection and processing of personal data within the European Union. The GDPR specifies rights of the data subject, defining claims everyone has on his or her personal data. These rights need to be granted by service providers and specified in their corresponding privacy policies which are the most important source of information for the internet user regarding the usage of their personal data. Such policies often lack readability and can cause an information overload affecting the user. During this thesis, we built a tool to extract the data subject rights according to GDPR. The base of this tool are two different supervised Machine Learning (ML) approaches, both trained on a manually labeled dataset of privacy policies, and a rule based sentence classifier, helping the user to capture the rights included in a policy more easily. The ML approaches were traditional ML algorithms, such as Support Vector Machine (SVM) and Decision Tree, and sentence classification using sequence labeling with a Conditional Random Field. The rule based classifier is based on a dictionary of keywords, automatically created based on the labeled corpus. The ML methods using the SVM classifier outperformed the other two approaches on the utilized dataset, whereas the CRF reached better results than the rule based classifier. The insights gained in this thesis demonstrate that ML methods are able to extract data subject rights to a certain extent and could be used to provide the internet user with support regarding the recognition of these rights. The results could possibly be improved by extending the dataset and adjust imbalances between the different classes or by combining different ML and rule based approaches.

Zusammenfassung

Heutzutage spielt das Internet eine bedeutende Rolle in unserem täglichen Leben, wodurch eine riesige Datenmenge generiert und gesammelt wird, darunter auch viele personenbezogene Daten der Web-Nutzer. Im Jahr 2016 trat die Datenschutzgrundverordnung (DSGVO) in Kraft, welche die Sammlung und Verarbeitung von personenbezogenen Daten in der Europäischen Union regelt. Sie enthält ein Kapitel, das die Rechte der betroffenen Person (Rechte jedes Nutzers an den sie/ihn betreffenden, personenbezogenen Daten) definiert, die von allen Anbietern im Netz gewährt und in deren Datenschutzerklärungen festgehalten werden müssen, welche die Hauptinformationsquelle für den Umgang mit persönlichen Daten darstellen. Da diese häufig schwer verständlich, lang und überwältigend für den Anwender sind, beschäftigte sich diese Arbeit mit der Entwicklung eines Tools zur automatischen Analyse von Datenschutzerklärungen und insbesondere der Extraktion der Rechte der betroffenen Person, um dem Nutzer das Erfassen der Inhalte zu erleichtern. Implementiert und getestet wurden zwei verschiedene Ansätze von überwachtem maschinellem Lernen (ML) und eine regelbasierte Methode zur Klassifikation der einzelnen Sätze der Datenschutzerklärungen, die die Basis dieses Tools bilden. Die ML Methoden sind zum einen klassische ML Algorithmen wie Entscheidungsbäume und zum anderen ein Sequence Labeling Algorithmus, das sogenannte Conditional Random Field (CRF), der als Klassifikator eingesetzt wurde. Der regelbasierte Ansatz arbeitet mit einem Wörterbuch aus Schlüsselwörtern der jeweiligen Klasse. Der traditionellen ML Ansatz erzielte auf dem verwendeten Datensatz die besten Ergebnisse und das CRF konnte eine höhere Genauigkeit erreichen als der regelbasierte Ansatz. Die in dieser Arbeit gewonnenen Erkenntnisse deuten darauf hin, dass überwachtes Lernen geeignet ist um die Rechte der betroffenen Personen aus Datenschutzerklärungen zu extrahieren. Die erreichten Ergebnisse können möglicherweise noch verbessert werden, indem der Datensatz vergrößert und die Ungleichverteilung der Beispiele pro Klasse aufgehoben wird oder verschiedene ML und regelbasierte Methoden kombiniert werden.

Contents

Acknowledgments **v**

Abstract **vii**

Zusammenfassung **ix**

Abbreviations **xv**

1 Introduction **1**
1.1 General Data Protection Regulation 1
1.2 Motivation 3
1.3 Aims and Contribution 4

2 Fundamentals of Computer-aided Text Analysis **5**
2.1 Pre-processing 5
2.2 Feature Extraction and Selection 6
2.2.1 Vector Representation 7
2.2.2 Linguistic Pre-processing 8
2.3 Classification 8
2.4 Sequence Labeling 8

3 Related Work **11**
3.1 Clear Representation of Policy Contents 11
3.1.1 Layered Privacy Notices 11
3.1.2 Platform for Privacy Preferences & Expandable Grid Representation 12
3.1.3 Privacy Nutrition Labels 13
3.2 Automated Understanding of Privacy Policies 14
3.2.1 Machine Learning Based Approaches 15
3.2.2 Rule Based and Combined Approaches 16

4 Requirements and Design **19**
4.1 Requirements . . . 19
4.1.1 Functional Requirements . . . 19
4.1.2 Non-Functional Requirements . . . 20
4.2 Design . . . 22
4.2.1 User Interface . . . 22
4.2.2 Classification Back-end . . . 23

5 Implementation **25**
5.1 Dataset . . . 25
5.2 Classification Methods . . . 27
5.2.1 Sentence Classification . . . 27
5.2.2 Sequence Labeling . . . 36
5.2.3 Rule Based . . . 38
5.3 User Interface . . . 40
5.4 Classification Back-end . . . 42

6 Evaluation **45**
6.1 Evaluation Metrics . . . 45
6.2 Evaluation of Relevant Classes for Tool . . . 46
6.2.1 Sentence Classification . . . 47
6.2.2 Sequence Labeling . . . 56
6.2.3 Rule Based Model . . . 60
6.3 Comparison of Different Methods . . . 65
6.4 Evaluation of Initial Classes . . . 68

7 Discussion and Future Work **71**
7.1 Discussion . . . 71
7.2 Future Work . . . 73

8 Conclusion **75**

List of Figures **77**

List of Tables **81**

Bibliography **83**

Abbreviations

BOW Bag Of Words

CATA Computer-aided Text Analysis

CRF Conditional Random Field

CUPS CyLab Usable Privacy and Security Lab

DT Decision Tree

GDPR General Data Protection Regulation

GUI Graphical User Interface

HMM Hidden Markov Model

IR Information Retrieval

kNN k-Nearest-Neighbor

ML Machine Learning

MLkNN Multilabel-k-Nearest-Neighbor

MLTSVM Multi-Label Twin Support Vector Machine

NB Naive Bayes

NLP Natural Language Processing

NLTK Natural Languaga Toolkit

POS Part of Speech

P3P Platform for Privacy Preferences

RF Random Forest

RAKEL **Ra**ndom **k**-lab**el**set

RQ Research Question

SVM Support Vector Machine

TF Term Frequency

TF-IDF Term Frequency - Inverse Document Frequency

UI User Interface

1 Introduction

Nowadays, the internet, with all the services it provides, plays an important role in all our lives. During the use of these services a huge amount of data is created and collected by the corresponding service providers. Legally binding regulations, like the General Data Protection Regulation (GDPR), have been implemented in order to protect the users, as well as their rights on and control over their personal data. It is crucial for service providers to be compliant to the specifications of the GDPR, since it specifies severe penalties up to twenty million euros or 4% of the total worldwide annual turnover in the financial year for the gravest violations (Art. 83 GDPR). To inform about their practices of information collection and processing and to show their compliance to the mentioned regulations, service providers utilize privacy policies which therefore are the most important source of information for the users with respect to the handling of their personal data [1].

1.1 General Data Protection Regulation

The GDPR has been implemented in 2016 in response to the increasing digitalization. It regulates the collection and processing of personal data within the European Union. It is applied from May, 25 2018 (Art99) and replaces the Data Protection Directive of the European Parliament and the Council of Europe from 1995 concerning the protection of individuals with regard to the processing of personal data and on the free movement of such data (Art94).
Chapter three of the GDPR is the core of this thesis. It specifies the rights of the data subject[1] and contains the following eight fundamental rights:

- **Right to be informed:** The data subject has the right to be informed about the collection and usage of his/her personal data.

[1] identifiable natural person whose personal data is collected or processed

- **Right to access:** The data subject has the right to learn whether or not his/her personal data is processed by the collector. If this is the case, the subject has the right to access the according data as well as more detailed information such as the purpose of the processing or the existence of automated decision making.
- **Right to rectification:** This right describes the claim of the data subject on requesting the correction of erroneous personal data, as well as the completion of personal data regarding the purpose of processing.
- **Right to erasure:** This right is also referred to as "right to be forgotten" and describes the obligation of the controller to erasure the personal data concerning the subject on his/her demand on in the GDPR specified reasons (e.g. the data is no longer necessary for the specified purpose of collection).
- **Right to restriction of processing:** The data subject has the right to restrict the data processing by the controller in specified cases (e.g. the processing is illegitimate).
- **Right to data portability:** This right describes the claim of the data subject on obtaining the personal data he/she provided to a controller in a structured form and in some cases, have the right to pass them on to another controller.
- **Right to object:** The data subject has the right to object the processing of their personal data.
- **Right not to be subject to automated decisions:** This right describes the claim of the data subject to not be subject to decision-making based in automated processing (this includes profiling).

These are the fundamental rights of every person on their personal data and need to be granted by every service provider. They need to specify these rights in their privacy policies to show their compliance to the GDPR. There are issues with these policies, such as hard readability and information overload which hinder the user from accessing essential information on their personal data collection and their rights, specified in the GDPR. These issues motivate the aim of this thesis to automatically extract the data subject rights and are explained in more detail in the next section.

1.2 Motivation

Privacy policies are the most important source of information for the internet user regarding the usage of their personal data and thus it is important for everyone to understand the content. However, studies show that these policies often lack readability and are hard to understand for non-experts [1, 2], due to their length and detail, causing an information overload. Obar et al. [2] have conducted a study with 543 participants with the result that 80% of them have only spent one minute or less reading the presented privacy policy. In addition, the time one would spend on reading every privacy policy from all the internet services we use is enormous. A study conducted in 2008 by McDonald et al. [3] estimated the annual reading time to 244 hours, if users read every policy from the utilized online services.

The described issues of lack of readability, information overload, and massive amount of reading time, are a big problem, since this way it is impossible for internet users to tell, if all their rights (and in our case the data subject rights specified in the GDPR) are covered by the according privacy policy and thus legally granted to be applied by the service providers. Furthermore, it is impossible to make informed decisions about privacy and data usage while using internet services.

However, a study conducted in terms of a research project supported by the Federal Ministry of Education and Research [4] with 1,000 participants shows that users are interested in protecting their data. 76% of the participants have already taken action in form of using pseudonyms instead of their real name, uninstalling applications or have even exercised one of their rights (access, object, etc.).

To tackle the mentioned problems and help users protect their data and make informed decisions, there have been different approaches to increase the readability of privacy policies [1, 5], such as privacy nutrition labels as a simple and compact presentation of the contents, which will be presented in Section 3. These attempts show that there is a need to make privacy policies more usable and easier for the user to understand. However, none of these approaches has been widely adopted yet, or lays its focus solely on the data subject rights.

1.3 Aims and Contribution

The aim of this thesis is to create a tool which makes it easier for users to quickly recognize whether all necessary data subject rights are covered in a regarded privacy policy, using an iterative research approach. Along the way the following Research Questions (RQs) are considered:

- **RQ1:** Which approaches exist to automatically analyze legal texts and privacy policies?
- **RQ2:** Are supervised machine learning methods suitable to analyze privacy policies with regard to the coverage of the data subject rights?
- **RQ3:** What are the strengths and weaknesses of sentence classification and sequence labeling based approaches?
- **RQ4:** Which supervised machine learning approach performs best at extracting data subject rights from privacy policies?
- **RQ5:** Which performance can be achieved on the test data and can the automated extraction add value for private consumers regarding the understanding of privacy policies?

To answer **RQ1** we conduct an extensive literature review looking for existing approaches to increase the readability of privacy policies, as well as Natural Language Processing (NLP) techniques in general and especially with regard to legal texts. The results of this review are presented in Chapter 3.

RQ2, RQ3, RQ4 and **RQ5** are examined by implementing and testing various supervised Machine Learning (ML) algorithms for sentence classification, as well as sequence labeling approaches, which is described in Section 5. Subsequently, these implementations are assessed by the means of different metrics and compared to each other. The gained results are presented in Chapter 6.

2 Fundamentals of Computer-aided Text Analysis

Computer-aided Text Analysis (CATA), or text mining, describes an interdisciplinary field which deals with machine supported analysis of texts. It uses techniques from research areas such as Information Retrieval (IR), NLP and ML in order to gain structured information from unstructured text data and to perform tasks like the clustering or classification of text (e.g. for spam filtering) [6]. CATA's relevance has increased drastically with the growing amount of web applications resulting in the emergence of machine readable text documents [6, 7]. As mentioned above, one type of increasingly occurring documents are privacy policies presented by internet service providers. In this thesis we want to examine whether text mining approaches are suitable to analyze these policies with regard to the coverage of the data subject rights presented in the GDPR. In the following paragraphs, we will look into the most important techniques which we have used during the text analysis process consisting of data pre-processing, feature extraction for representing the text, and knowledge discovery which in our case are classification and sequence labeling tasks [7].

2.1 Pre-processing

ML algorithms and computers in general are based on numbers and not capable of processing text. Thus, it is important to prepare the textual data, also referred to as text corpus, in order to translate them into a format that computers and algorithms can handle [6]. This step applies NLP techniques to prepare the data for the ML algorithms which then can be used for classification or other IR tasks. There exist well established pre-processing steps, like stopword removal or stemming, which can be used to reduce the size of the dictionary of all contained words and are briefly explained in the following sections.

Tokenization Tokenization describes the process of segmenting the coherent text into a dictionary of words. This step is necessary in order to obtain all words that are used in a text. This can be achieved by eliminating all punctuation and replacing all white spaces and non-textual letters by a single white space. [6]

Filtering Filtering describes the process of removing words from the text corpus in order to reduce its size and prevent frequently occurring words with little importance from outweighing the more meaningful tokens. A commonly used filtering method is the stopword removal. This means to remove words with little or no content information like articles or prepositions (e.g. "the", "a", "and") by using predefined lists containing stopwords with general and meaningless words. [6, 7]

Stemming Just like the filtering process, stemming aims to reduce the dimensionality of the dictionary by decreasing the number of different words. Stemming algorithms reduce the number of grammatical word forms such as "delete", "deleted", "deleting", "deletion" to the corresponding root form or word *stem* "delete". For this purpose, different algorithms have been developed which apply a set of rules, such as removing prefix or suffix, in order to achieve their goal. [6, 8]

Lemmatization Lemmatization works similar to stemming. Instead of using a set of rules to obtain the word stem this method tries to reduce the word forms to their roots or *lemma* by understanding the word form, meaning the part of speech (e.g. noun or verb), and the context of the word. [6, 8]

2.2 Feature Extraction and Selection

The strength of ML methods lies in working on numbers and they are not capable of processing textual data. Hence, for further processing with ML methods it is necessary to store the corpus in a data structure which is more suitable than plain text. One possibility to do so, is by describing the text document solely based on the set of occurring words which is the so called Bag Of Words (BOW) or "Vector Space Model" [6, 7]. Alternative representations of the text are so called word embeddings which are obtained by training ML models and represent the similarity between words. In addition to transforming the text data into a vector representation, we can add

additional features, such as Part of Speech (POS) tagging or prefix and suffix of a word, in order to provide our ML model with further information. In the end, different combinations of the extracted features are tested and the best subset of the pool of potential features is selected for the final model.

2.2.1 Vector Representation

As mentioned above, there are two different ways to represent the text corpus in a structured form: BOW approaches, resulting in a sparse vector representing the occurring words and word embeddings, which represent each word with an own vector obtained from ML algorithms.

Bag of Words When using the most elementary BOW model, every word existing in the corpus is represented as a single variable storing its number of occurrences, the so called Term Frequency (TF). It is possible to add weights to these numbers in order to grant higher importance to meaningful words. In order to do so, Term Frequency - Inverse Document Frequency (TF-IDF) can be used which, in addition to the TF, also takes the document frequency (number of documents containing the respective word), as well as the number of documents into account. [7]

Word Embeddings The word embedding techniques transform a high dimensional sparse vector, in a form they can be obtained from BOW models, where every word takes one dimension, to a dense vector space. This way, they create a vector for every word and, other than BOW models, can take distances between individual words into account [9]. It is possible to use pre-trained word vectors obtained by algorithms, such as Google's word2vec[1] [10, 11] or GloVe[2] (global vectors for word representation) [12] which have been trained on billions of tokens obtained from Wikipedia, Twitter, Google News data and the like and contain hundreds of vocabularies. Furthermore, it is possible to train word embeddings directly on the data the CATA task is performed on. Thus one can make sure that every word is contained in the vocabularies which is especially interesting for texts that differ from the corpus used for the pretrained embeddings.

[1] https://code.google.com/archive/p/word2vec/
[2] https://nlp.stanford.edu/projects/glove/

2.2.2 Linguistic Pre-processing

To increase available information about words, sometimes linguistic pre-processing is used to create additional features. One often applied approach is the POS which associates a part of speech tag such as noun, verb, or adjective to each word. [6]

2.3 Classification

Classification is mostly a supervised ML task, meaning a set of training data is required, where each record is labeled with a class value. In case of text or sentence classification, these training data consist of a text corpus and labels for each document or sentence. These data are used to construct a classification model which then tries to make predictions on unseen data. This method is used in applications such as news or spam filtering or, document organization.

Frequently used classification models are Naive Bayes (NB), Support Vector Machine (SVM), Logistic Regression, Decision Tree (DT) and Random Forest (RF), as well as different types of Neural Networks. NB works with the class-conditional distribution, SVM separates two classes using a hyperplane with maximum margin, Logistic Regression makes use of the logistic function in order to gain output probabilities, and DT and RF classifiers recursively divide the input data resulting in regions corresponding to defined classes [6, 7, 13].

In order to find out which sentence describes one or multiple data subject rights as defined in the GDPR an approach of ours is performing sentence classification as described above. In this case, the defined classes correspond to the different data subject rights, meaning a sentence is assigned to a class if it contains the corresponding right.

2.4 Sequence Labeling

Sequence labeling is another, different kind of supervised learning model which can be used to extract information about contained data subject rights in a text corpus. Sequence labeling describes the process of labeling each unit in a sequential data stream which, in this case, is the input text. Examples for these methods are POS, where each word is labeled with the corresponding part of speech, and named entity recognition.

the latter means the tagging of each word with one of several pre-defined categories, such as organizations, locations or names. The difference to standard classification methods, like NB or SVM, is that sequence labeling approaches take surrounding words and context into account [6].

Commonly applied methods are Hidden Markov Model (HMM) and Conditional Random Field (CRF). HMMs use a given sequence of tokens in order to compute a probability distribution over possible label sequences. These models consult the so called Markov assumption which states that for the prediction of the future only the present state is relevant, not the past [14]. CRFs as well as the HMMs are a sequence modeling framework and have been presented by Lafferty et al. [15]. They have a model for the probability of the entire label sequence in comparison to the independence assumption of Markov Models.

We use a text corpus with accordingly labeled tokens to apply sequence modeling methods to the problem of extracting and recognizing the in the rights specified in the GDPR. It is used to train one of the mentioned models in order to generate output labels corresponding to the classes of data subject rights.

3 Related Work

In Chapter 2, we gave a short overview over CATA and text mining techniques and their general use-cases like information extraction or text summarization. The analysis of privacy policies is a more specialized application of text mining techniques. The following sections present existing CATA approaches as well as other methods for increasing the readability of privacy policies.

3.1 Clear Representation of Policy Contents

There exist different approaches for a simple and clear way of presenting the content of privacy policies, thus increasing the readability. These methods aim to inform the user about the most relevant aspects without the need to read the entire text, while strongly relying on cooperation of website owners and service providers or manual creation of the representations.

3.1.1 Layered Privacy Notices

The concept of layered privacy policies came up in 2003, presenting the content in a short and easy to understand format. A standardized form was presented by the *Article 29 Data Protection Working Party in the Opinion on More Harmonised Information Provisions*[1]. According to this definition, a layered representation of privacy notices can consist of up to three layers which differ in detail.

Layer 1 - The short notice contains the core information of the policy defined in Article 10 of the Data Protection Directive, such as the data controller and the purpose of processing.

Layer 2 - The condensed notice depicts the basic information in more detail, in particular all information required under the Data Protection Directive, such as the

[1] http://www.statewatch.org/news/2004/dec/wp100.pdf

purpose of processing and choices available to the individual. Furthermore, it has to contain contact information of the controller.

Layer 3 - The full notice contains the full text of the privacy policy.

Studies show that the layered representation enables faster decision making than the full-text policy. However, this representation does not increase the understanding of the text. The layered approach is a first, to some extent, standardized representation. Nevertheless, it still provides companies with a certain flexibility regarding the structure of the different layers, making it ineffective to use the top layer because of varying structure choices. [16, 17, 18]

3.1.2 Platform for Privacy Preferences & Expandable Grid Representation

The Platform for Privacy Preferences (P3P) project offers a standardized, machine-readable language. The P3P policies in this XML-like format are provided by the website-owners and can be interpreted by user-agents, like a web-browser plug-in. When visiting a website, these agents search for P3P policies, analyze them and display prompts indicating the sites privacy practices. P3P also provides the user with the possibility to specify a set of privacy preferences which can be automatically checked by the user-agent. [19]
Since P3P does not specify a specific presentation format, the CyLab Usable Privacy and Security Lab (CUPS) [20] presented the P3P Expandable Grid. This representation shows the policy content in an interactive matrix consisting of thematic blocks corresponding to the P3P Policy Statements. Figure 3.1 shows an extract of a P3P Expandable Grid depicting the P3P statement "Ordering Books". Every block lists all the information this company collects and uses ten different symbols to illustrate how the data is used and shared for each row. [20]
Studies show that the P3P representation did not help the users to capture and understand the privacy notices content [20, 21]. As possible reasons, CUPS identified among others confusing icons and missing focal points [20]. This is where the approach presented in the next section builds on.

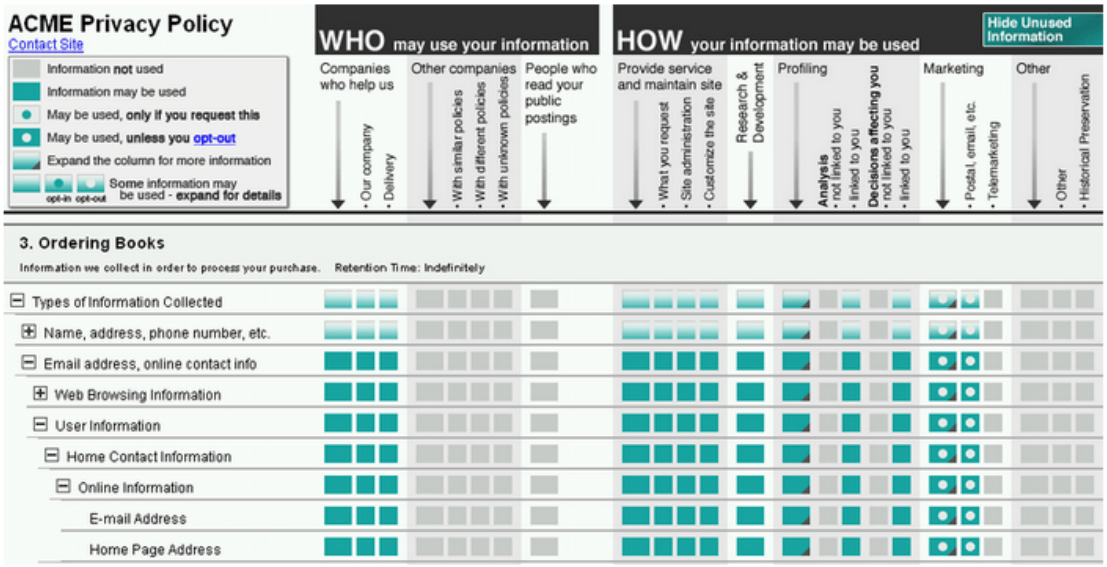


Figure 3.1: Extract of a P3P Expandable Grid

Source: [20]

3.1.3 Privacy Nutrition Labels

Privacy Nutrition Labels, as presented by Kelley et al. [21], are another one of the existing approaches which try to present contents of privacy policies in a standardized form. These labels are inspired by nutrition warning and energy labeling known from the obligatory nutrition facts panel from the United States. Kelley et al. used an iterative and user-centered design process to create their representation, whereby insights in already existing labels, such as the P3P Expandable Grid (see Section 3.1.2) and their issues were considered. This way, they created an intuitive representation of policy content making it easy to find information and to understand differences between policies. [21]
An extract of their final state of the Privacy Nutrition Label is depicted in Figure 3.2. It consists of a table containing ten rows representing one data category each. These categories describe different types of information that may be collected and processed by the service provider, such as contact or demographic information and cookies. The columns describe the way the information is used, as well aswith whom it is shared. Four different symbols are used to depict distinct levels of data collection and usage.

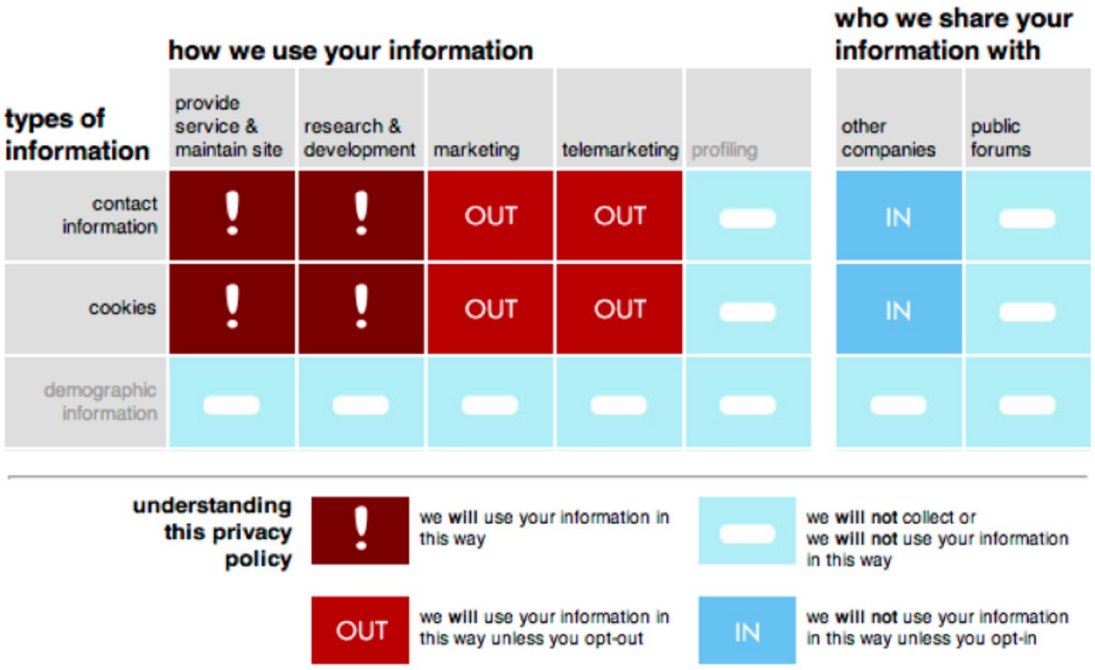


Figure 3.2: Extract of Privacy Nutrition Labels by Kelley et al.

Source: [21]

Kelley et al. distinguish between the four levels where information is always used, used until you opt-out, not used until you opt-in, and never used. [21]
These labels could be created manually but they have been implemented in a way that enables automatic generation by the P3P technology [16]. They show promising results in online studies, outperforming the layered policy approach and full-text versions both in comprehensibility and time spent scanning [18]. However, they have not been widely adopted yet [1].

3.2 Automated Understanding of Privacy Policies

Researchers proposed different approaches for an automatic analysis of privacy policies. They have tried to use ML, NLP and rule based methods, as well as combinations of these.

3.2.1 Machine Learning Based Approaches

One ML and NLP based solution has been presented by Costante et al. with the objective to check privacy policies for completeness [22]. This is achieved by calculating a completeness level under consideration of different privacy categories and weights for these categories which are specified by the user. The results are depicted in a User Interface (UI) showing the completeness grade, as well as the individual categories and whether they are included or not. In order to assess which categories are covered, they used supervised ML techniques. For this purpose they created a corpus by manually labeling privacy policies with six core categories, such as data collection and purpose, and eleven additional categories like cookies and data storage location. The pre-processing consists of text normalization and stopword removal, along with a BOW transformation using TF-IDF. To classify paragraphs of the privacy notices they tested different classifiers, namely NB, SVM, Ridge Regression, k-nearest neighbors and DT. All classifiers but the NB reached a maximum F1-score (definition see Section 6.1) of 92%. [22]

Another ML based approach has been presented by Ammar et al. [23]. They aim to automatically categorize the policy text in order to answer categorical questions about them. Their data corpus made use of crowdsourcing annotations from the "Terms of Service; Didn't Read" project [24],and of additional policies without annotations. The pre-processing consisted of lowercasing as well as punctuation and stopword removal. The problem they want to solve can be described as a binary classification, representing the presence and absence of a concept like the *transparency on law enforcement requests*. As classification model they used logistic regression together with supervised learning and self-training methods. On the task of recognizing the *transparency on law enforcement requests*, they reached an F1-score up to 76%. However, their algorithm performed worse on the second concept they tested, namely the *user's right to terminate their account*. The authors gave the number of available positive instances of this concept and a machine translation during the corpus generation as possible explanations. [23]

Tesfay et al. [25] proposed their approach *PrivacyGuide* which also works with ML and NLP based methods. Their *PrivacyGuide* is a summarization tool for privacy policies focused on the GDPR. They also created a graphical UI consisting of a dynamic dashboard showing high-level information using colored icons. They defined classification categories based on the privacy aspects of the GDPR, resulting in eleven

categories, such as data collection, third-party sharing or account deletion. Additionally, for each category three risk levels were determined. These classes were used to create a manually labeled corpus of privacy policies. As pre-processing steps, Tesfay et al. identified a keyword catalog using the labeled corpus, as well as stopword removal, tokenization and stemming. For the text classification, Tesfay et al. used four different classifiers, namely NB, SVM, DT and Random Forest. The proposed *PrivacyGuide* reached a weighted average accuracy of 74%. [25]

3.2.2 Rule Based and Combined Approaches

Assuming that privacy policies contain fixed patterns, Costante et al. proposed to extract information about the data being collected by the website owner using a set of extraction rules and pattern matching [26]. The procedure for generating these rules contains text pre-processing steps, such as tokenization, POS and Named Entity Recognition. The authors defined their own entities by analyzing their corpus of privacy policies resulting in entities like Data Provider, Data Collector or Provider Action. The annotation pattern engine Jape [27] uses the resulting annotations in order to extract regular expressions describing the document and hence creating extraction rules. They consist of two parts, namely the pattern to be matched and the according action to take once the pattern was matched. Classification using the created extraction rules reached an F1-score of around 80%. Costante et al. name the POS as the weak point in their pipeline, causing most of the erroneous predictions. [26]

The Privee architecture presents a combination of rule and ML based classification, additionally using crowd-sourcing results [28]. This module has been implemented as browser extension showing the classification result in a standardized label format. To generate these labels, the crowd-sourcing results from the "Terms of Service; Didn't Read" project [24] are checked for the corresponding policy and directly transformed into labels. In case there is no result for the policy, the rule and ML based classifiers come into play. The applied pre-processing and feature generation includes lowercasing and TF-IDF on bigrams [2]. The classification into six different categories (collection, encryption, ad tracking, limited retention, profiling and as disclosure) is performed by pattern matching using regular expressions and in case the rule based method fails, ML algorithms, such as Random Forests and Naive Bayes, are applied. Using this

[2] two consecutive words or a word and a punctuation

combination, Privee reaches an overall F1-score of 90%. [28]

Another research project making use of ML and rule based methods to extract information from German privacy policies is "PGuard" which aims to help the end-user to evaluate the trustworthiness of applications regarding data privacy [4]. For this purpose, they analyze applications technically, for example by observing data flows of the application, and semantically, meaning parsing the corresponding privacy policy and extracting relevant information about the collection of personal data. They created a dataset by crawling online services and domain experts annotated them manually resulting in 175 labeled policies containing 17.000 sentences. For information extraction, they developed different methods depending on the searched content. To find contact information, they used a rule based system based on regular expressions and lists to recognize of German addresses (cities, countries, e-mail addresses, etc.). Additionally, they used the annotated dataset to train a ML classifier and find information like "personalized merchandise" or "processing in a third country". The used classifiers are SVM, RF and NB and they are trained on the corpus after removing numbers, special characters and stopwords, as well as lowercasing and other pre-processing steps. To encode the sentence, TF and TF-IDF were used and compared. Depending on the class, the best F1-score reached lies between 79.0% and 95.86%.

None of the described tools has been widely adopted and they rely on the collaboration of service providers to manually create the privacy notices in the desired form of layered notices, as nutrition labels, or in the P3P language. The research on automated analysis of privacy policies shows that it is possible to extract and classify their content. The described approaches focus on different information they want to extract from the policies, but none of them targets the data subject rights defined in the GDPR. This is why this thesis aims to evaluate a method of automatically extracting these rights and presenting them to the user in a practical form.

4 Requirements and Design

This thesis aims to create a tool to display the data subject rights contained in a policy, consisting of a Graphical User Interface (GUI) and a text classifier in the background. The following sections describe the requirements for this tool, as well as the planned design of the different parts.

4.1 Requirements

The requirements for our policy analysis system are divided into functional and non-functional requirements and are categorized according to their significance. The essential requirements are marked with •, whereas the optional features are tagged with ∗.

4.1.1 Functional Requirements

The functional requirements mostly concern the GUI, describing what the user can do when coming across a privacy policy and wanting to analyze it according to the data subject rights. The specifications are inspired by the tools described in Chapter 3 which take one privacy policy, analyze it according to the specified metrics and present the results to the user. Additionally, we considered possible use-cases to better understand the functional requirements of our tool. Figure 4.1 depicts the use-case diagram for the GUI which interacts with the User and the text Classification Algorithm in the back-end. In general, our system needs to provide three use-cases, namely "Specify Privacy Policy Text", "Analyze Privacy Policy Text", and make it possible for the user to "Get Overview Of Included Rights". The last two cases are connected to the Classification Algorithm which performs the analysis and provides the classification results. The depicted child use-cases specialize the classifier "Analyze Privacy Policy Text" and represent the User's ability to choose between the different classification methods implemented in this thesis.

Since our tool is not integrated in the browser, as for example P3P, we need to enable the user to specify the according privacy policy by providing the URL, like in [25] or in any other way. The results need to be presented to the user in an easy and understandable manner and since we want to implemented different approaches for the text analysis, we aim to provide the user with the possibility to choose one of them. These observations and considerations resulted in the following functional requirements.

- User can input a privacy policy to analyze in the form of raw text
- User can start the analysis of the entered policy
- GUI displays the results of the analysis of the policy according to data subject rights: Show included rights and mark the according parts in the text

  * User can input a privacy policy to analyze in the form of PDF or URL to the according website
  * User can choose the preferred classifier

During our implementation, we focused on the first three described requirements (input policy text, start the analysis and display of results) since they describe the basic functionality of the analysis system and cover all three defined use-cases.
The fourth point describes a convenience for the user, making manually copying the policy text unnecessary. The last functionality is interesting in the case that different classifiers provide very different classification results. Especially regarding the two cases of a high recall and low precision (many detected positives, containing many false-positive labels) and high precision (few detected positives, many of them true positives). The first case has the advantage, that the probability of missing any contained rights is small. However, a high recall and low precision involve the risk of depicting rights as included which are actually missing. Here, the user can check the selected sentences manually. In the second case, it is just the other way around.

4.1.2 Non-Functional Requirements

The non-functional requirements relate to both the UI and the performance of the ML classifier, describing the necessary capabilities of the tool to provide added value for the

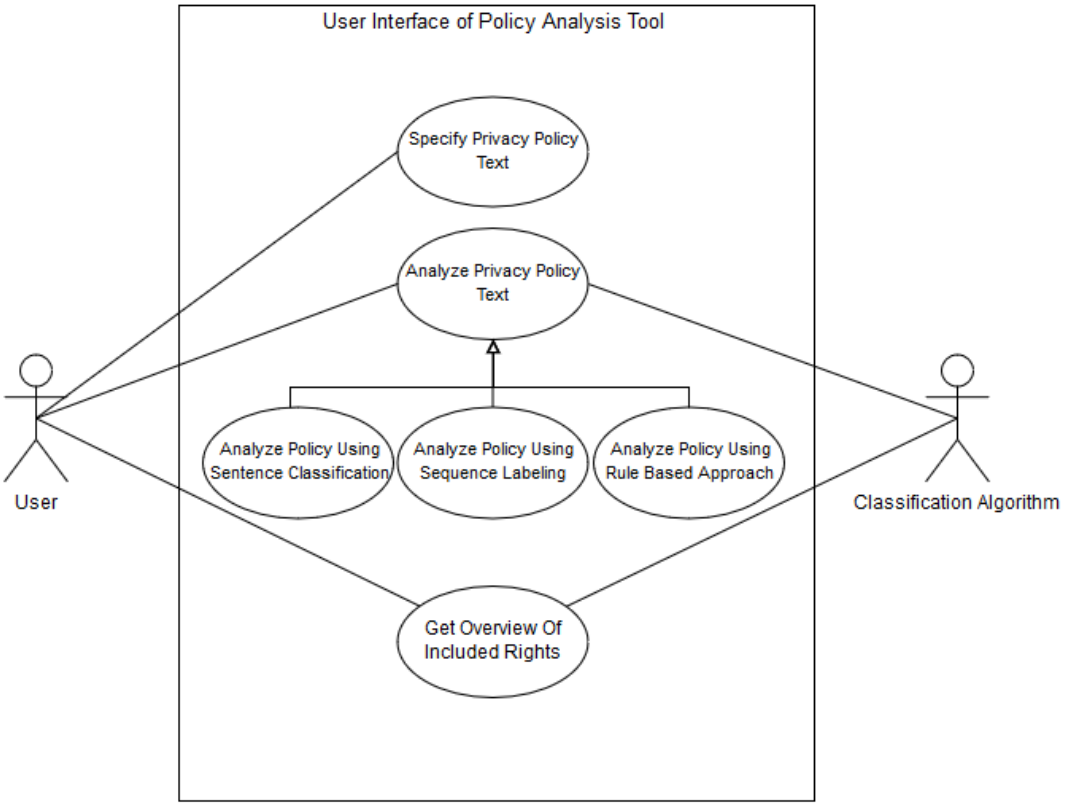


Figure 4.1: UML use-case diagram for analysis tool GUI

Source: Own illustration

internet users. We worked out the following non-functional requirements, by looking at other tools for depicting content of privacy policies:

- UI is intuitive and easy to use
- UI is connected to the text classifier
- UI provides a faster overview over contained rights, than manual reading/scanning of the policy
- Accuracy of system equals or outperforms the understanding of a non-expert user, under the assumption of average policy reading time

* Accuracy of system equals or outperforms the understanding of a non-expert user and expert user, even with precise scanning of the policy

The first four of these non-functional requirements are essential for providing the user with added value. The optional feature presents the optimal state of such an analysis system.

4.2 Design

In order to meet the in Section 4.1 carved out requirements, we designed a software system consisting of two parts, the UI enabling the user input and presentation of the analysis results and the classification back-end providing these results. Figure 4.2 shows an overview of these two components and technologies that can be used within and for communication in between them. Both of these systems are described in more detail in the following sections.

4.2.1 User Interface

For building our UI, we used the javascript library React[1] which has been developed by Facebook. This library makes it easy to build interactive UIs since it provides reusable components managing their own state and encapsulating the corresponding logic. To meet the essential functional requirements, our UI consists of the following components:

- Text input field for entering privacy policy text and highlighting the resulting sentences
- Button to start the analysis of the privacy policy text
- Indicators for the data subject rights showing their state

The connection to our classification back-end is realized using an HTTP POST request, sending the textual data and receiving a response. This response contains the classification results affecting the state of the indicator components and text highlights in the text input field. The UI is a web application which can be accessed in a browser.

[1] https://reactjs.org/

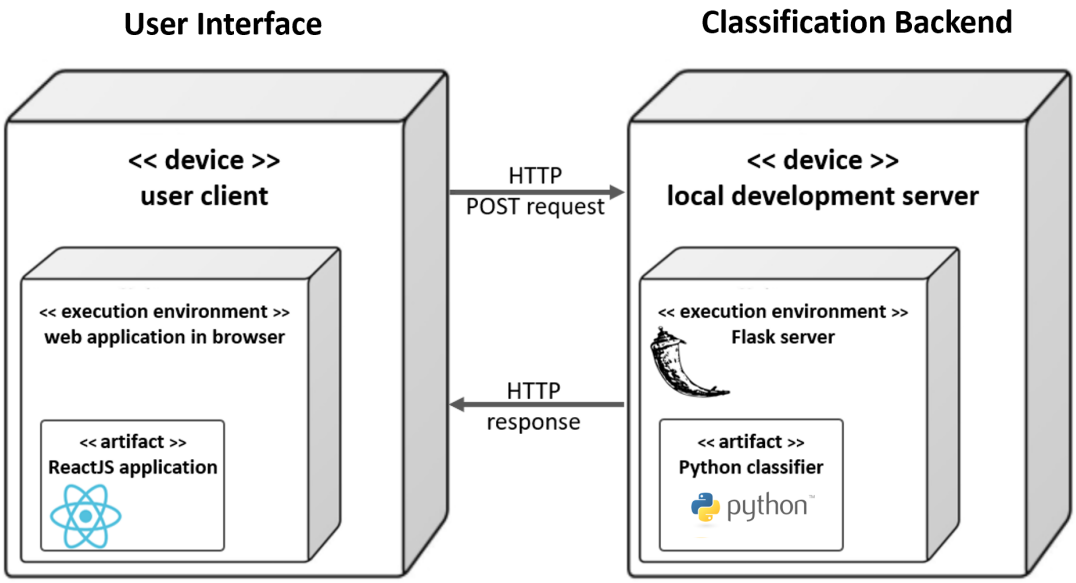


Figure 4.2: UML deployment diagram for the analysis system

Source: Own illustration using logo prints for React, Flask and Python
https://reactjs.org/
https://www.fullstackpython.com/flask.html
https://www.python.org/community/logos/

The intuitive usage is the remaining necessary requirement relating the UI. Based on results from related work presented in Chapter 3, it is important to use simple icons and clear focal points. We focus the attention of the user on relevant texts by highlights and color coded indicators. Figure 4.3 shows a first sketch of a possible UI design for an intuitive and easy usage. It contains all components described above and is kept as simple as possible. We consider adding a drop-down menu in order to fulfill the optional functional requirement of choosing a classification algorithm.

4.2.2 Classification Back-end

For the classification back-end we created different classifiers using ML and rule based methods in a Python program and compared their performance against each other (see Section 6). In the finalized tool, the strongest classifier (or the preferred one, in the case of letting the user choose classification method) is provided by the back-end

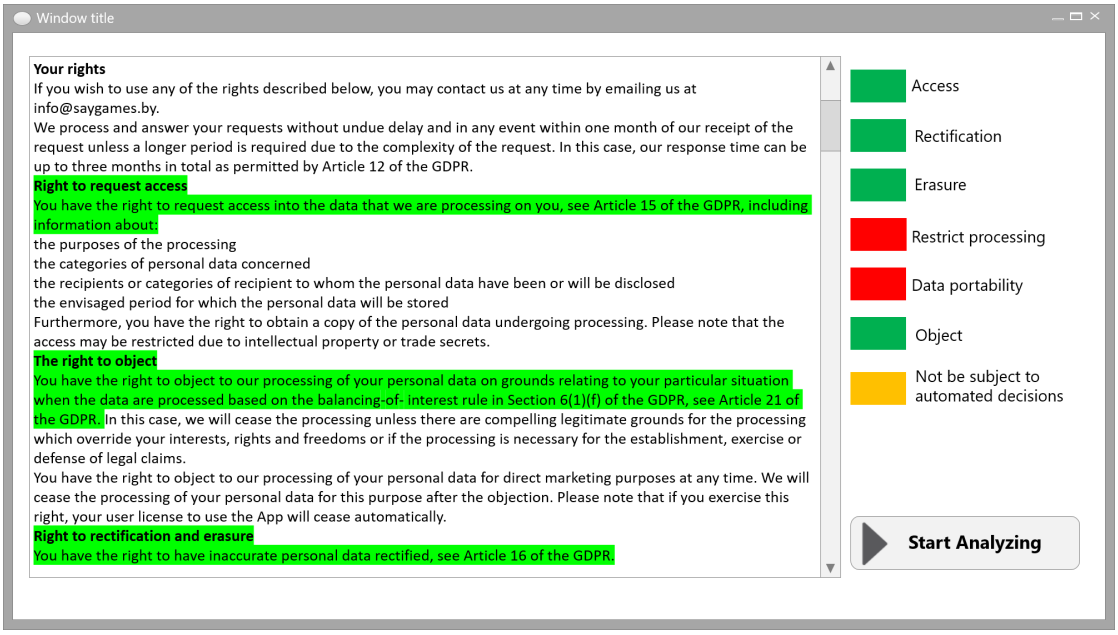


Figure 4.3: First prototypical draft of the UI

Source: Own illustration

consisting of a Flask server. Flask[2] has been developed by Armin Ronacher and is a web framework for Python. It provides a locally running web-server which we chose to use for our back-end implementation, since it is very flexible, easy to use and provides us with all the functionality we need for our prototypical UI implementation. It is responsible for performing the whole CATA pipeline from pre-processing to classifying the text sent by the front-end via a POST-request and returning the obtained results. The server provides the classifiers with the pre-processing and feature combination which performed best during our implementation and evaluation phase. This is important to meet the fourth non-functional requirement concerning the accuracy of the tool.

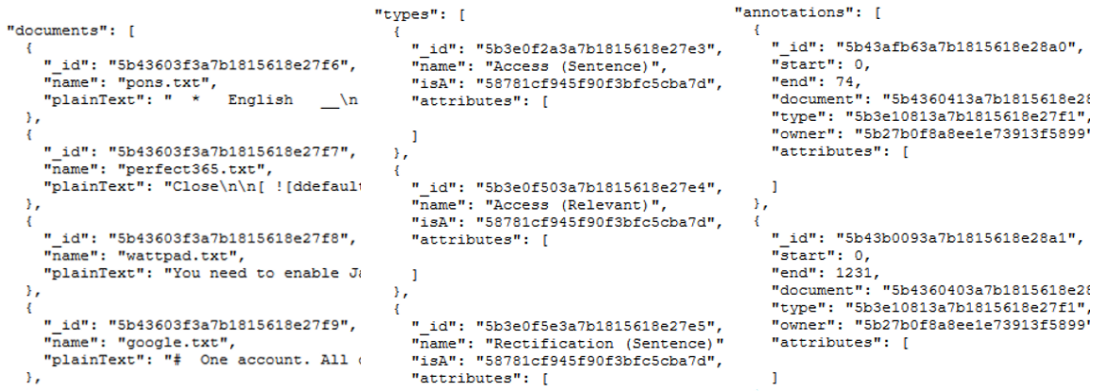
[2]https://palletsprojects.com/p/flask/

5 Implementation

The in Section 4 characterized tool consists of two components. The Flask server, containing the Python back-end and providing the text classification, and the React UI, providing the user with all necessary information and interaction possibilities. This chapter describes the implementation of these components, as well as the creation of the different classification methods and the dataset used for training and testing.

5.1 Dataset

To begin with, the chair of Software Engineering for Business Information Systems provided a dataset. It was created using a web crawler browsing through the Google PlayStore, extracting all privacy policies of the first 40 apps in every category in the app-charts. The policies were extracted as HTML-files and processed using a Python parser. After extracting and storing the text of the policies, three domain expert labeled them manually by dividing the policies amongst them, labeling 97, 29 and 10 respectively. The experts used two different annotation approaches, on a sentence and a token level. The dataset totals 17 labeling categories, resulting from seven different data subject rights (access, rectification, erasure, restrict processing, data portability, object, not to be subject to automated decisions) labeled on two levels and three other classes (not part of privacy policy, data subject rights paragraph, data subject rights title), labeled on sentence level. The class "not part of privacy policy" contains parts like the HTML-header. This information is not relevant for our use-case of copying the policy text and is not considered in this thesis. This results in the initial dataset, consisting of 136 annotated privacy policies, containing 1794 relevant annotations (see Table 5.1).
They are stored in a JSON-file, containing three objects, as shown in Figure 5.1. The first object contains a list of documents (Figure 5.1a), representing the single policies, including an identifier, the name of the application, and the full-text. The second object is



(a) (b) (c)

Figure 5.1: Structure of dataset in JSON-format

Source: Own screenshot of dataset

called "types" (Figure 5.1b) and contains all possible classes, such as "Access (Sentence)", representing the right to access on sentence level, with a corresponding identifier. The third class describes the actual annotations (Figure 5.1c), consisting of the ID of the policy and the annotation type, as well as the start and end point of the annotation. Different processing and formatting steps are required for the algorithms to be able to work with this corpus which are described in more detail in the corresponding sections in Chapter 5.

The results achieved on this dataset did not reach high accuracy scores, as determined during the first evaluation phase (see Section 6). To evaluate the classification performance on a bigger dataset, we used the crawler to obtain more policies and annotated them manually by ourselves. Together with the initial dataset, we obtained 262 labeled policies and 3472 annotations, which almost doubles the previous amount of data. The newly created annotations were stored in the same format as described above. Table 5.1 gives an overview of the amount of labeled sentences and tokens as well as the class distributions of the initial and during this thesis extended dataset. The additional annotations were not created by a domain expert, which needs to be kept in mind when analyzing the results on the extended dataset.

As depicted in Table 5.1, the class distribution in the obtained corpus is unbalanced. The *Right to Access* is represented by the most examples (126 in the smaller set and 267 in the larger one), whereas the corpus contains the fewest examples for the class *Right*

not to be subject to automated decisions (21 in the smaller set and 60 in the larger one). This can be observed for the token based labels as well and is caused by annotated policies, not containing all of these rights (e.g. since they do not concern users in the EU), or mentioning some of the rights in more than one sentence. This unbalance can lead to difficulties during the training of text classification algorithms and needs to be considered during implementation and evaluation.
The two classes "data subject rights paragraph" and "data subject rights title" are not directly relevant for the UI described in Chapter 4.2. It only shows if the data subject rights are included or not and highlights the corresponding parts. This is why they were almost neglected when creating the extended dataset during this thesis. Nevertheless, they can be interesting for future work on the topic of extracting data subject rights or other parts of privacy policies by pre-structuring the text into relevant paragraphs. In contrast, a relevant class for the designed tool is a null-class for the sentences which do not contain a data subject right, providing the algorithm with negative samples during the training, to teach it to deal with the whole text of privacy policies and not only the relevant sentences. We annotated 100 negative sentences for the initial and 200 for the extended corpus to approximately match the distribution of the other classes.

5.2 Classification Methods

The background process of our tool consists of a text classification algorithm, trying to extract the relevant parts of the privacy policy. We implemented different concepts for text classification, namely sentence classification techniques using simple ML classifiers, a sequence labeling based approach, and a rule-based method using a dictionary of keywords. The implementation followed the in Section 1 described process of CATA, containing the steps of data pre-processing, feature generation, and knowledge discovery which are described in the following sections.

5.2.1 Sentence Classification

The task of extracting different contents of a text can be seen as classification task. Other authors [4, 22, 23] used this perspective to give an overview of the content of privacy policies. We aim to categorize each sentence of the regarded privacy notice and highlight the sentences associated to any data subject rights class. This categorization

	Sentence-Based		Token-based	
	initial	extended	initial	extended
Right to access	126	267	148	329
Right to rectification	114	258	158	331
Right to erasure	120	259	158	319
Right to restrict processing	75	159	113	211
Right to data portability	66	150	98	249
Right to object	108	204	136	244
Right not to be subject to automated decisions	21	60	26	81
TOTAL	630	1357	837	1764

	initial	extended
Data subject rights paragraph	165	167
Data subject rights title	162	184
No right included	100	200

Table 5.1: Labeled instances per data subject rights classes on two levels and three additional classes on sentence level for the initial and extended dataset

is a multi-label classification problem, since every sentence can contain zero, one, or multiple data subject rights leading to either one or multiple labels.

Formatting of Dataset

Because of the multi-label classification problem, we needed to transform the dataset. It contained an individual entry for every annotation, meaning for every sentence with multiple label, there are multiple entries. Since the classification algorithm cannot detect duplicate sentences, we transformed the representation into a matrix with one column per class and stored the newly generated corpus in a CSV-file. Every sentence is represented by one row, where the columns are representing the class labels for each $class_i$ with $i \in \{1, 2, ..., 7\}$ in all data subject right classes are encoded as follows:

$$class_i = \begin{cases} 1, & \text{sentence in class}_i \\ 0, & \text{else} \end{cases} \tag{5.1}$$

Some sentences appeared more than once in the resulting CSV-file. This was caused by multiple annotation entries for sentences belonging to more than one class. In some cases, these annotations differed in the start or end position by a small amount, for example including a blank at the beginning or a period at the end. This resulted from the manual annotation process and we filtered these duplicates by applying a threshold of ± 1 to start and end position.

Data Pre-processing

We implemented and compared the impact of the following well-established steps, preparing the textual data for the classification algorithm and to reducing the size of the dictionary,

Noise Removal and Lowercasing The text in the corpus contained many special characters like square brackets and parentheses, since it was extracted from HTML-files. We deleted all characters, except the letters from A to Z, as well as numbers, and replaced them with a single white space. Furthermore, we removed possible spaces at the start and end of every sentence and converted the whole text into only lowercase letters.

Stopword Removal As described in Section 2, filtering the text corpus by removing words with little importance is a common way to reduce its size and make sure that meaningful tokens are not outweighed. For this purpose, we used a predefined list of English stopwords from the Natural Languaga Toolkit (NLTK) package in Python [29] which contains around 180 frequently used words, like "a", "to", or "which".

Stemming and Lemmatization We implemented stemming and lemmatization, two very similar methods, in order to reduce the number of different words in the corpus. For the stemming process, we used the algorithm for removing the tokens' suffixes introduced by Porter et al. [30]. The lemmatization we used is based on the Wordnet database [31], where English nouns, verbs, and adjectives are combined to sets of synonyms. Implementations for both of these algorithms are provided by the NLTK.

Feature Extraction

After the text pre-processing we generated a vector of features to represent the text in a suitable form for the ML algorithms. We used different vector representations for the sentences to be classified, as well as additional information like the POS.

Bag of Words We implemented a sparse vector representation for the sentence using both, simple word count vectors (TF), as well as vectors, giving more weight to meaningful tokens, using $TF - IDF$. The weighted BOW model takes into account how many documents contain the regarded word (df) [7]:

$$TF - IDF(word) = TF * log\frac{Number\ of\ documents}{df(word)} \tag{5.2}$$

Word Embeddings we used different approaches, for representing the sentences using word embeddings. First, we created our own word vectors by training Google's Word2Vec algorithm [10, 11] on our corpus of privacy policies. The open-source NLP package Gensim[1] provides a Python implementation of Google's deep learning algorithm.
Pre-trained sets of word-vectors can be used, to improve the performance of a model with the lack of a large labeled training dataset. Our implementation contains pre-trained vectors obtained by standfords GloVe algorithm trained on six billion tokens from Wikipedia entries and English Gigaword, consisting of newswire text data collected and archived by the Linguistic Data Consortiume[2], resulting in a dictionary of 400 thousand tokens. The second collection of pre-trained vectors we chose for our implementation was obtained from training Google's Word2Vec algorithm on 100 billion words on parts of the Google News Dataset[3]. To transform these word-vectors into one vector for the sentence to classify, two different approaches have been implemented. The first one calculates the mean of all word-vectors contained in the sentence. The second one uses the TF-IDF weights and computes the sentence vector as a rated average of the corresponding word-vectors.
Another possibility to obtain a vector representation for the whole sentence is to directly

[1] https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/word2vec/index.html
[2] https://catalog.ldc.upenn.edu/LDC2011T07
[3] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

learn a dense representation from it. This way, information like word-order and semantics can be taken into account. An algorithm to perform that task has been presented by [32] and is provided by the Gensim library[4]. We use this doc2vec implementation to train a sentence-vector representation for our corpus.

Part-of-Speech-Tagging Part-of-Speech POS tagging can be used to feed additional information to our model. Our implementation makes use of POS Tagger provided by the NLTK. We merge the word with its corresponding POS-tag (e.g. PRP for preposition, VB for verb and NN for noun), to include the obtained tags into our vector representation of the sentence, resulting in training samples like:

```
you_PRP may_MD update_VB or_CC correct_VB information_NN you_PRP have_VBP
provided_VBN to_TO us_'PRP'
```

Multi-Label Classification

After preparing the textual data and generating a suitable feature representation, supervised ML algorithms come into play. Sentence classification with respect to the data subject rights is a multi-label problem. This means that one sentence can contain multiple data subject rights and thus needs to be assigned to multiple classes at the same time. The initial dataset used in this thesis contains 79 and the extended one contains 185 sentences with two or more different labels. For example the sentence

> *You have the right to request that we confirm what personally identifying information (PII) we collect or hold about you, provide a copy of your PII to you in a machine readable format and to ask us to correct or update the PII.*

contains the right to access, rectification and data portability.
Most of the basic ML classifiers described in Section 2.3 are not able to handle multi-labeled data. In general, there are two different approaches to deal with this multi-label problem, algorithm-adoption or problem-transformation. [33]

Algorithm-Adaption The algorithm-adoption approaches work with extensions of existing binary classification algorithms to make them able to handle multi-label data directly. This means the algorithm is adapted to fit the data. In our implementation we

[4]https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/doc2vec/index.html

included the following two adopted algorithms. [33]

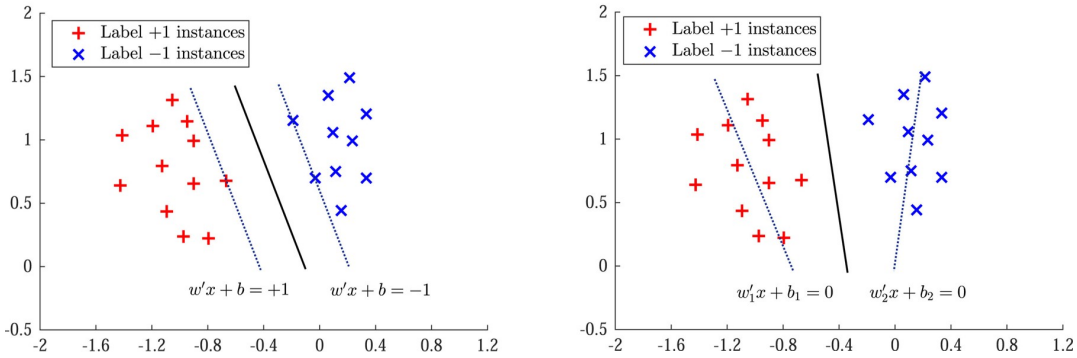
The **Multilabel-k-Nearest-Neighbor (MLkNN)** classifier is based on the traditional k-Nearest-Neighbor (kNN) method where a data-point is assigned to the most frequent class among its neighbors. Similarly, MLkNN determines the kNNs in the trainingset for every unseen data-point. Instead of choosing the most frequent class among the kNN_x of data-point x, statistical information obtained from its neighbors in the form of a membership counting $C_x(cl_i)$, describing the number of neighbors of x belonging to a class cl_i, and the maximum a posteriori principle are used to identify the corresponding labels. [34]

$$C_x(cl_i) = \sum_{n \text{ in } kNN_x} f(n, cl_i) \tag{5.3}$$

$$f(x,i) = \begin{cases} 1, & x \text{ in class } i \\ 0, & \text{else} \end{cases} \tag{5.4}$$

The scikit-multilearn [35] library provides an implementation of the MLkNN method which we used for our implementation.

Multi-Label Twin Support Vector Machine The standard SVM classifier tries to determine a hyperplane to separate two classes with a maximum margin between the data-points closest to the plane (support vectors) and the plane. As depicted in Figure 5.2a, the margin can be represented by two parallel hyperplanes, equidistant to the decision boundary plane. The Twin SVM approach relaxes this requirement and allows nonparallel hyperplanes (Figure 5.2b). Chen et al. proposed a modification, allowing the Twin SVM to deal with multi-label data. The aim of their Multi-Label Twin Support Vector Machine (MLTSVM) is to generate a hyperplane for each possible label, being as close to the corresponding data-points and as far away from the others as possible (see Figure 5.3a), resulting in decision boundaries as depicted in Figure 5.3b. This approach adapts the liner SVM classifier which determines a hyperplane to separate two classes with a maximum margin between the data-points closest to the plane (support vectors) and the plane. The scikit-multilearn library [35] also provides an implementation of the MLTSVM method which we used for this thesis.[36]



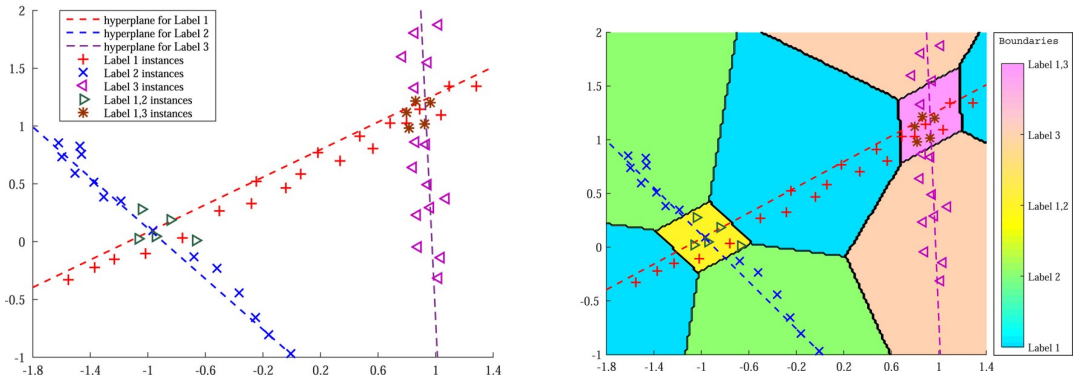
(a) SVM (b) Twin SVM

Figure 5.2: Geometric interpretation for SVM and Twin SVM

Source: [36]

Problem-Transformation The problem-transformation methods convert multi-label classification problems into one or multiple single-label classification problems. Tsoumakas et al. [33] describe existing methods which are based on deleting either whole entries with multi-labels or randomly all but one label for each entry in the dataset. Both of these approaches delete and ignore a lot of information. Our problem relies on manually labeled data and it is not an option to loose any of these information. Tsoumakas et al. and other researchers also describe alternative approaches without the loss of knowledge [33, 37, 38]. Our implementation contains the methods binary relevance, label powerset, random k-labelset, and classifier chain.

Binary Relevance The binary relevance method is also known as one-vs-all or one-vs-rest strategy. This algorithm performs an independent binary classification for the prediction of every label l in the set of all labels L by learning a binary classifier for each of them. It converts the corpus into $|L|$ different datasets (one for every possible label $l \in L$), containing binary labels l and $\neg l$. Here $l = 1$ if the sentence is in the class corresponding to label l and $\neg l = 1$ otherwise. One drawback of this approach is that it does not consider correlations between labels [38]. Table 5.2a shows an example for an original dataset with $|L| = 3$ different labels, whereas table 5.2b shows the resulting $|L|$ datasets, containing binary labels encoding each class. The scikit-multilearn library [35] provides an implementation of the binary relevance method which we used for



(a) Hyperplanes

(b) Decision Boundaries

Figure 5.3: Geometric interpretation for MLTSVM

Source: [36]

our implementation. [33, 38]

Label Powerset The label powerset method transforms the dataset in a way that it is possible to learn one single-label classifier by generating a powerset of all labels. This powerset consists of every single subset of labels, considering these subsets as a new label. This way, correlations between labels are taken into account, but the resulting dataset may consist of many more classes and fewer examples per class than before the transformation. Table 5.2c depicts an example of the dataset being transformed using the label powerset method, resulting in only one example per class. For this transformation method we used the implementation provided by scikit-multilearn [35]. [33, 38]

Random K-Labelset The **Ra**ndom **k**-lab**el**set (RAKEL) method was proposed by Tsoumakas et al. [38] and is an ensemble of label powerset classifiers. It aims to take advantage of the label powerset's ability to consider correlations between labels, while avoiding the problem of many classes with low number of samples. This is achieved by combining label powerset classifiers, each trained on a different random subset of the set of labels, using a threshold on the average binary decisions of each model per label. For this transformation method we also used the implementation provided by scikit-multilearn [35]. [38]

Sentence	Access	Object	Erasure
1	1	0	1
2	1	1	0
3	0	1	1

(a) Original dataset

	Access	¬Access
1	1	0
2	1	0
3	0	1

	Object	¬Object
1	0	1
2	1	0
3	1	0

	Erasure	¬Erasure
1	1	0
2	0	1
3	1	0

(b) Dataset transformed using binary relevance

	Access ∧ Object	Access ∧ Erasure	Object ∧ Erasure
1	0	1	0
2	1	0	0
3	0	0	1

(c) Dataset transformed using label powerset

Table 5.2: Illustration of the problem transformation methods binary relevance and label powerset

Classifier Chain The classifier chain method was presented by Read et al. [37] and performs independent binary classifications, similar to binary relevance, but is able to model correlations between labels, like RAKEL and Label Powerset. The classifier chain results from the training of every binary classifier on the input space extended by all previous classifiers in the chain. Same as for the other transformation methods, we used the implementation provided by the scikit-multilearn library [35]. [37]

After transforming the problem into single-label classification tasks, it is possible to apply standard classification algorithms like the ones described in Section 2.3. We decided to use traditional classifiers instead of Neural Networks, as we lacked a big training dataset. In our implementation we used the scikit-learn [39] implementations of standard SVM, as described in Chapter 5.2.1, Logistic Regression, DT, and RF.

5.2.2 Sequence Labeling

As mentioned before, we also tried to extract data subject rights using sequence labeling methods, where every single token is labeled, utilizing the token-based annotations of the dataset, as they were created for this purpose. The advantage of sequence labeling is that it is possible to take contextual information into account. In this thesis we used the CRF for the sequence labeling task which are based on HMMs, but relax their independence assumption.

Formatting of Dataset

Similar to sentence classification, the corpus needs to be processed and transformed in order to perform sequence labeling using CRFs. The labels are stored as start and end position of the relevant tokens, containing also whole expressions like "right to object". Furthermore, irrelevant tokens have no tag at all, while for sequence labeling every single token needs to have a label, we transformed the dataset into a list containing one list per sentence consisting of a tuple per word, adding the label "I" for not labeled and thus irrelevant tokens. This results in lists looking similar to the following example: :

```
[[('you', 'I'), ('may', 'I'), ('update', 'Rectification (Relevant)'),
('or', 'Rectification (Relevant)'), ('correct', 'Rectification (Relevant)'),
('information', 'I'), ('you', 'I'), ('have', 'I'), ('provided', 'I'),
('to', 'I'), ('us', 'I'), ('by', 'I'), ('going', 'I'), ('into', 'I'),
('the', 'I'), ('user', 'I'), ('account', 'I'), ('settings', 'I'),
('screen', 'I'), ('within', 'I'), ('the', 'I'), ('app', 'I'), ('', 'I')],  ...]
```

Data Pre-processing

The pre-processing of the corpus consists of the same methods as described in Section 5.2.1 and used for the sentence classification using ML classifiers. These methods are noise removal, lowercasing, stopword removal, stemming, and lemmatization. We implemented all of these methods in order to compared their impact on the sequence labeling results.

Feature Extraction

The features suitable for sequence labeling differ from the ones that were used for the sentence classification task, here we do not try to represent a sentence by one vector,

but every token of the sentence individually. Thus, we need to define characteristics of single tokens and we are able to take the word order and context into account.

Prefix and Suffix One feature we extracted are the prefix and the suffix of the token, by viewing the first and last three letters of the corresponding word. This is especially helpful for POS, which is a special case of sequence labeling. A suffix like *-ness* or *-ment* can be an indication for a noun (e.g. happiness, kindness, government), whereas the suffix *-ing* can be a clue for the present participle form of a verb [29]. This feature is highly affected by the pre-processing steps of stemming and lemmatization, since these methods remove the suffix of a token.

Part-of-Speech-Tagging As mentioned before, POS tagging itself is a sequence labeling task. It also can be used as a feature for other NLP tasks. Again, we used the POS-Tagger provided by the NLTK, but other than for the sentence classification, where we connected the POS-tag directly to each word, here we extended the elements of our list of tuples by the POS-tag. This results in a list of triples containing the token, the corresponding POS tag and the class label producing samples like:

```
[[('you', 'PRP', 'I'), ('may', 'MD', 'I'), ('update', 'VB', 'Rectification
(Relevant)'), ('or', 'CC', 'Rectification (Relevant)'), ('correct', 'VB',
'Rectification (Relevant)'), ('information', 'NN, 'I'), ('you', PRP', 'I'),
('have', 'VBP', 'I'), ('provided', 'VBN', 'I'), ('to', 'TO', 'I'), ('us',
'PRP', 'I')...],  ...]
```

Other Characteristics Furthermore, for each token we check if it is a number, the first or last word of a sentence or an upper-case word and if the sentence is a title (every word is upper-case). The last two features forfeit their information value, when applying the pre-processing step of lowercasing. The test for the word being a number can be interesting in cases, where the corresponding paragraph of the GDPR is mentioned next to a data subject right.

Surrounding To take context into account, we add the preceding and following word to our feature vector for every token where its is possible (not the first or last one of the sentence, where we only add the preceding or consecutive word respectively). Additionally, we include all the information we can get about these neighbors in the form of the features described above. Here, the pre-processing step of removing

stopwords interferes with the process, since it may change the predecessor and successor of a token.

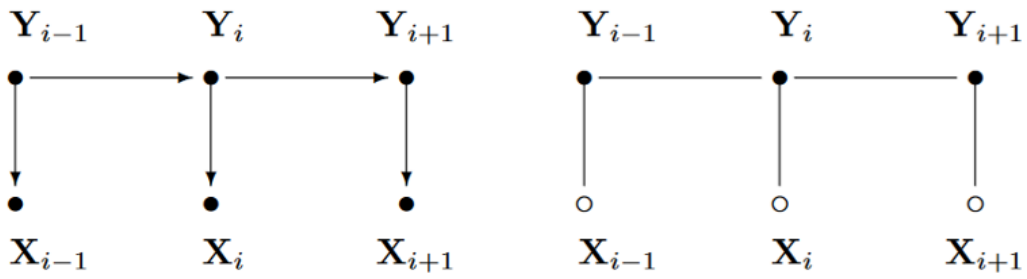
Model

As mentioned before, our model for the sequence labeling task is the CRF. This approach has been introduced by Lafferty et al. [15] and it is a discriminative, graphical model for sequential data. Other than a directed graphical model, such as a HMM (see Figure 5.4a), the CRF is undirected and knows the complete input sequence at any point of the modeling process. Thus, it does not model any observations, as indicated by the open circles in Figure 5.4b. [15]
For the implementation in this thesis, we used the library CRFsuite[5] and the python binding python-crfsuite[6] to access the API and train a linear chained CRF sequence labeling model. The model can be trained by creating a trainer and then passing the training data including the extracted features to this trainer.
The labels describe relevant tokens and contain also whole expressions like "right to object", resulting in class labels for the words "right", "to", and "object". For the final result, it is not crucial that all of these three words are labeled correctly, but that at least one of them is recognized. For this reason and because of a better comparability to the sentence classification approaches described in Chapter 5.2.1, we decided to use our CRF as a sentence classifier. We assigned a sentence to a class, if it contains at least one token with the corresponding label.

5.2.3 Rule Based

Additionally to the ML classification methods, we implemented a rule-based classifier based on a dictionary. Just like the sequene labeling algorithm, this approach works on the token-based annotations. The application area is limited to the seven classes of data subject rights and we generated a simple dictionary containing the relevant tokens of the sentences for each of them.

[5] http://www.chokkan.org/software/crfsuite/
[6] https://github.com/scrapinghub/python-crfsuite



(a) Hidden Markov Model (b) Conditional Random Field

Figure 5.4: Graphical structures of sequence models

Source: [15]

Data Pre-processing

The pre-processing of the corpus consists of the same methods as described in section 5.2.1 for the sentence classification using ML classifiers, namely noise removal, lower-casing, stopword removal, stemming and lemmatization. Similar to the sentence classification and sequence labeling approaches, we implemented all of these methods in order to compared their impact on the classification results.

Rule Based Model

For a rule based sentence classification, we used the token-based labeled corpus to automatically create a dictionary to look up all words in a sentence and check for keywords for a class of data subject rights.
This dictionary contains the labeled words as keys and the corresponding class labels as value. We introduced thresholds for adding the tokens to the dictionary, to check which of the labeled tokens are relevant for the classification task. We added a threshold t_{cl_i} which relates to the number n_{jcl_i} how often a token T_j appears for one class cl_i and a threshold $t_{\neg cl_i}$ which relates to the number $n_{j\neg cl_i}$ how often a token T_j appears for all the other classes. A token T_j is added to the dictionary as keyword for class cl_i if it appears more often than t_{cl_i} in sentences labeled with class cl_i and the number of occurrences in sentences of all the other classes is lower than $t_{\neg cl_i}$. This is expressed by the following rule:

$$T_j \text{ in dictionary for class } cl_i = \begin{cases} true, & n_{jcl_i} > t_{cl_i} \wedge n_{j\neg cl_i} < t_{\neg cl_i} \\ false, & \text{else} \end{cases} \tag{5.5}$$

For the classification task, all words of a sentence are matched with the keyword entries in the dictionary. If the sentence contains enough relevant tokens for one data subject right, it is assigned to the according class. For this purpose, we introduced a third threshold t_s which relates to the number of relevant tokens in a sentence n_{jcl_i}, being matched with an entry in the dictionary for a class cl_i of data subject rights, resulting in the following classification rule for each class cl_i:

$$\text{sentence in class } cl_i = \begin{cases} true, & n_s > t_s \\ false, & \text{else} \end{cases} \tag{5.6}$$

The optimal values for these thresholds were figured out by testing the performance of combinations of them on the labeled corpus and finding the peak. This process is described in more detail in Chapter 6.

5.3 User Interface

The UI was implemented using the JavaScript framework React. The final version with an exemplary analyzed policy is shown in Figure 5.5.
To meet the essential functional requirements specified in Chapter 4.1, it needs to provide the user with an input possibility for the privacy policy. This has been realized with a text input field, where the user can copy the corresponding text. Furthermore, we added a button to start the analysis, enabling the user to send the input data to the underlying Flask back-end, performing the classification with the selected machine learning algorithm. The connection to the Python server was realized over an HTTP POST request using ajax (Asynchronous JavaScript And XML) [40]. To display the results, the created UI contains an indicator for each subject right. When the corresponding right is included, the indicator turns green and orange otherwise. Additionally, the detected sentences are highlighted in the textfield in the color corresponding to the according right. We also added a menu for choosing between the three different

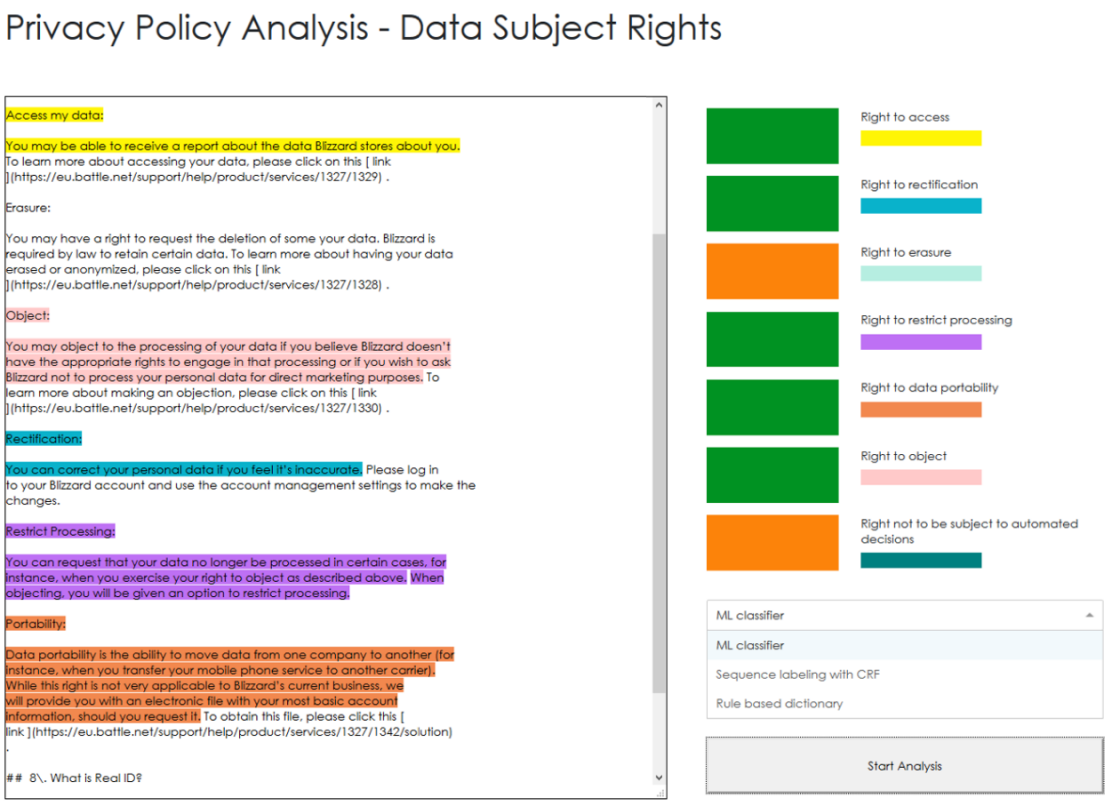


Figure 5.5: Final version of the UI after analyzing a policy

Source: Own screenshot

approaches, namely ML classifier, sequence labeling using CRF and the rule based approach. This feature fulfills one of the optional functional requirements.
Figure 5.6 shows an overview of the implemented React components and the corresponding states, props and methods, as well as raised events. The parent component **PolicyAnalyzer** renders the two components **TextInputField** and **OverviewPanel**. It reacts to text inputs by handling the event *onInput* which is raised by the **TextInputField**, when the contained text changes. This text is then stored in its state, in order to pass it to the **OverviewPanel** component. Furthermore, the parent component handles the result of the text classification, coming from the event *onResult*, raised by the **OverviewPanel**, and takes care of the highlighting of the according text in the **TextInputField**. Since it is not actually possible to change the style of text in a textarea

by default, we used the jQuery plugin "highlight-within-textarea"[7] to achieve our goal. The **OverviewPanel** component contains the **AnalysisButton**, the drop down menu **AlgorithmDropdown** for choosing the classification method, and a list of right **Indicator**s. It stores the names of the latter and whether they are included or not as a list of dictionaries in its state. This state, and thus the color of the indicators, is updated while handling the event *onAnalysis* which is raised by the **AnalysisButton** when clicked. The method *handleAnalysis* passes the text input data to the Flask back-end, receives the classification results and raises the event *onResult*. The component **AlgorithmDropdown** raises an event each time a new classification method is selected. This event is handled by the **OverviewPanel** in the method *handleChooseAlgorithm* which stores the chosen classifier in its own state in order to pass it to the Flask back-end for the text analysis.

5.4 Classification Back-end

The back-end to our React application consists of a locally running Flask web-server which provides a framework for Python. This enables us to easily transfer the in Python implemented classification methods and making them available for interacting with the React front-end. The server provides two routes, binding two different URLs to the corresponding functionality. The first one is the standard route rendering the UI HTML-page. The second route handles the HTTP-POST request sent by the React front-end. The request comes from the **OverviewPanel** component and contains the input text, as well as the selected classification method (ML classifier, sequence labeling or rule based). The Flask route returns an error message in the case of an empty input field, informing the user to enter the text of a policy in order to start the analysis properly. If valid input data was passed over the POST request, the Python program starts going through the text analysis pipeline, including the pre-processing and feature extraction steps which were described in the previous sections. For the classification, we stored the best performing classifiers of the categories ML classifier, sequence labeling, and rule based, such that our back-end only needs to load the corresponding model and apply it to the prepared textual data and classify all sentences in the text. The response contains a dictionary of all data subject right classes and a boolean describing if one or

[7] https://github.com/lonekorean/highlight-within-textarea

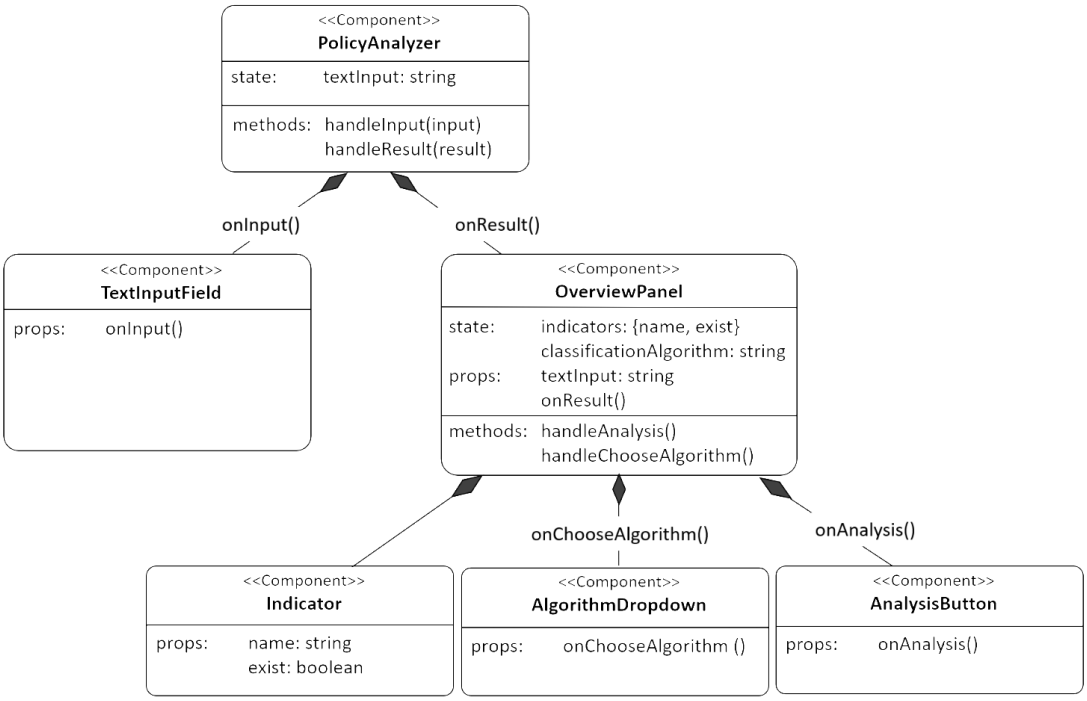


Figure 5.6: Components of the React Application

Source: Own illustration

multiple sentences are assigned to the according class. Additionally, the program sends an array of all extracted sentences containing at least one data subject right. This array is then used by the **PolicyAnalyzer** component to highlight the corresponding text in the text input field.

6 Evaluation

In this chapter, we will give an overview of the quantitative results of the different algorithms and the impact of the implemented preprocessing steps and different features. As mentioned in Chapter 5.1, the two initially provided classes "data subject rights paragraph" and "data subject rights title" are not directly relevant for the tool implemented in this thesis but can be interesting for future work. This is why we will provide a short overview of the results on these two classes in Section 6.4. In contrast, the null-class was not represented by the initial dataset, but is of great importance for the designed tool. The results for these classes (seven data subject rights and the null class) are described in Section 6.2. Besides that, we explain the used evaluation metrics and compare the three different approaches of sentence classification using traditional ML, sequence labeling, and the rule based method.

6.1 Evaluation Metrics

For the quantitative analysis of our implemented sentence classification approaches, we used different evaluation metrics. The precision P (6.1) is defined as the fraction of true positive labeled datapoints (TP) over all positive labeled points ($TP + FP$) and describes how many of the positively labeled points are labeled correctly. [41]

$$P = \frac{TP}{TP + FP} \tag{6.1}$$

The recall R (6.2) describes how many of the positive datapoints are correctly labeled as positive and is defined by the fraction of true positive labeled datapoints (TP) over the sum of TP and false negative (FN) labels. [41]

$$R = \frac{TP}{TP + FN} \tag{6.2}$$

The F1-score (6.3) is the weighted average of the two metrics described before.

$$F1 = 2 * \frac{P * R}{P + R} \tag{6.3}$$

These metrics are defined for binary classification tasks and need to be adapted for the multi-label case. Since our class distribution is unbalanced we use the weighted precision, recall, and F1-score (6.4 a-c) which compute the corresponding metric for each label and calculate the average weighted by the support s (number of true instances) of each class label l of all class labels L.

$$P_{weighted} = \frac{1}{\sum_{l \in L} s_i} * \sum_{l \in L} s_i P_i \tag{6.4a}$$

$$R_{weighted} = \frac{1}{\sum_{l \in L} s_i} * \sum_{l \in L} s_i R_i \tag{6.4b}$$

$$F1_{weighted} = \frac{1}{\sum_{l \in L} s_i} * \sum_{l \in L} s_i F1_i \tag{6.4c}$$

The presented resulting scores are based on a 5-Fold cross-validation, meaning the dataset was split in five disjunctive parts, where four are used for training the classifier and one is used as testset. The training of the algorithm utilizis 80% of the whole corpus, while the evaluation uses the remaining 20%. This is done five times, with the final result being the average of these five runs, reducing the chance of a biased result for one randomly chosen split of the dataset. This is beneficial especially in scenarios with a limited amount of data, since the chance of every possible observation appearing in the training and test-set rises.

6.2 Evaluation of Relevant Classes for Tool

This section gives an overview of the quantitative performance of the three presented classification methods on the corpus containing the seven data subject rights and the newly added null class. This evaluation describes the quality of the text classification as it is performed in the back-end of the policy analysis tool. We are also describing the results on the initial and the extended dataset. The initial one was created by domain experts, in contrast to the annotations for the extended set. This is why we show the results on both sets.

6.2.1 Sentence Classification

For the sentence classification algorithms, we compare the two approaches *algorithm adoption* and *problem transformation*. Additionally, we show the impact of different pre-processing methods like stopword removal, as well as the value of different features, and the quality of the diverse classification methods used with the problem transformation. Furthermore, the improvement of the classification by extending the dataset is presented.

Initial Dataset As Table 5.1 shows, the initial dataset contains 630 labeled sentences for the seven data subject rights ranging between 126 labels for the "right to access" and 21 for the "right not to be subject to automated decisions". For accurately representing the status of our policy analysis tool, we added a null class containing 100 examples of sentences of privacy policies not describing any data subject right. These 730 sentences are used to train our ML classification algorithms.

Problem Transformation We implemented four different problem transformation approaches, namely binary relevance, label powerset, random k-labelset, and classifier chain to deal with the multi-label classification problem resulting of sentences containing more than one data subject right. After converting the problem into binary classification tasks, we tested four different ML classifiers. We used SVM, linear regression, DT, and RF resulting in 16 combinations of transformation and classification algorithms. Besides that, we used different pre-processing steps, features, and word embeddings, increasing the number of possible combinations. In this section we only describe the most significant results and comparisons.
When using word embeddings (selftrained doc2vec and w2v embeddings and the pre-trained models GloVe and Google's Word2Vec) as text representation, the highest score weighted F1-score over all classes is 49.8%. It is achieved by the the label powerset transformation together with the SVM classifier based on the TF-IDF weighted GloVe embeddings, after lowercasing the text and removing the stopwords. Figure 6.1 gives an overview of the best resulting F1-scores achieved on the corpus processed this way, comparing the different kinds of embeddings. This distribution looks similar for all the different implemented pre-processing combinations. As depicted in the plot, the selftrained word and sentence embeddings are outperformed by the pre-trained models.

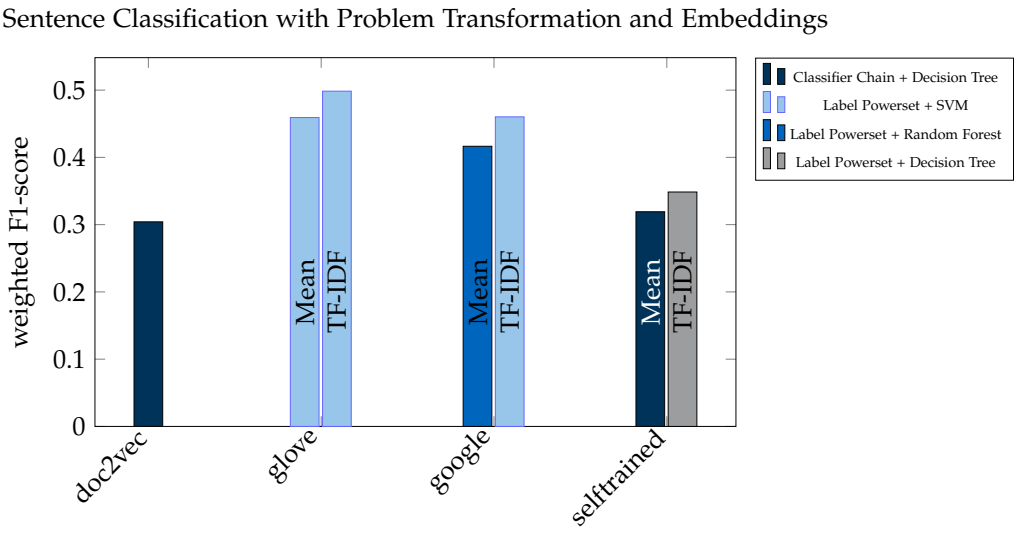


Figure 6.1: Maximum weighted F1-score results using embeddings on initial dataset on lower case input text, after removing stopwords with different problem transformation and classifier combinations

Source: Own illustration

Furthermore, adding TF-IDF weights when transforming the word embeddings into a sentence vector increases the quality of the classification. Besides that, it is shown that for different embeddings, different combinations of transformation method and classifier yield the best results.
Using BOW models for a sparse vector representation outperforms the classification based on the word embeddings. Here the transformation and classifier combinations reached weighted F1-scores up to 79%. This result is around 30% higher than the best value achieved using word embeddings. In general, classification based on TF lead to slightly better results than TF-IDF. Furthermore, adding the POS tags to the tokens did influence the scores negatively. These results can be seen in Figure 6.2 which gives an overview of the best achieved F1-scores using TF and TF-IDF both with and without POS tagging. These results were achieved on a dataset preprocessed using lowercasing and stemming. As shown in Figure 6.3, both stopword removal and lemmatization lead to marginally (two to five percent) lower results. The values depicted in the graphic are the scores reached by the best classifier and problem transformation methods in

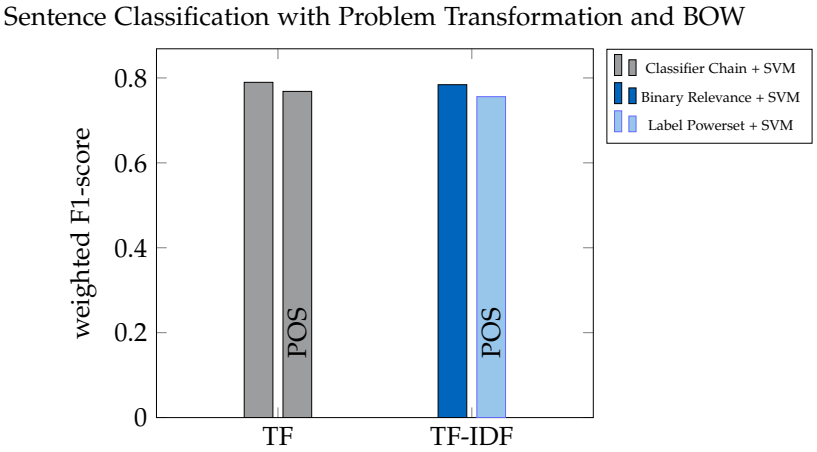


Figure 6.2: Maximum weighted F1-score results using TF and TF-IDF on initial dataset on lower case input text, after stemming with different problem transformation and classifier combinations and with or without usage of POS tags

Source: Own illustration

the case of TF (Classifier Chain and SVM) and TF-IDF (Binary Relevance and SVM) are shown.
The best performing combination is the classifier chain transformation in combination with the SVM classifier based on TF encoded sentences. For this combination, Figure 6.4 depicts the F1-score per class. As the graphic shows, in most of the cases using TF as encoder leads to better results than using TF-IDF. Furthermore, the classes with the lowest result, the "right to restrict processing", the "right to data portability", and the "right not to be subject to automated decisions" are the ones with the lowest amount of training samples (75, 66 and 21). The null-class has a low F1-score as well, despite the 100 created training sentences. This can be explained by the high variety of sentences not containing a data subject right.

Algorithm Adoption We tested two different algorithm adoptions, namely MLkNN, based on the kNN classifier, and MLTSVM, based on the SVM classifier. In this section, we describe the results of these two algorithms, together with the impact of different features and pre-processing steps.
The maximum weighted F1-score achieved by the MLTSVM on the initial dataset lies

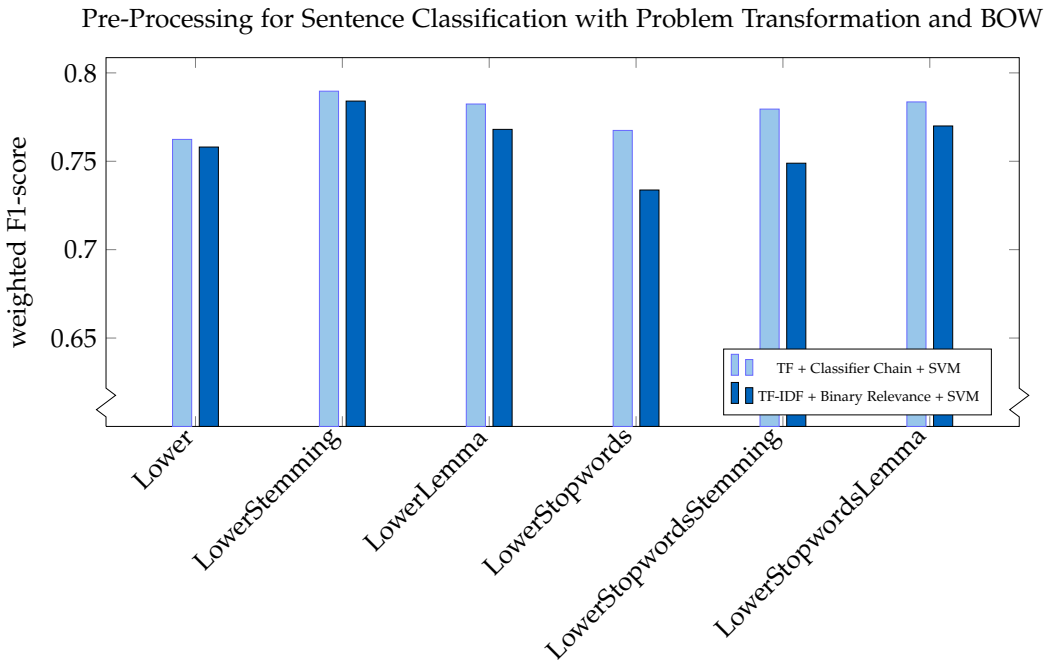


Figure 6.3: Maximum weighted F1-score results using TF and TF-IDF on initial dataset with the best performing problem transformation and classifier combination depending on different text pre-processing methods

Source: Own illustration

at 39%. It is achieved using TF after lowercasing and stemming the corpus. Figure 6.5 gives an overview of the best results achieved with TF and TF-IDF. As the plot depicts, MLTSVM performs better based on TF than TF-IDF. Furthermore, adding POS tags could not improve the results.

Figure 6.6 gives an overview of the achieved F1-scores of the MLTSVM on the initial dataset, using different combinations of pre-processing methods and the better performing TF encoding. Here one can see that the best results could be achieved using lowercasing and stemming, but the deviation when using other combinations does not exceed seven percent.

For the MLkNN we tested different numbers for k, the number of nearest neighbors. The highest F1-score of 69.5% was achieved using $k = 3$ neighbors on a lowercase corpus after stemming and, in contrast to MLTSVM, based on TF-IDF. Figure 6.5 shows

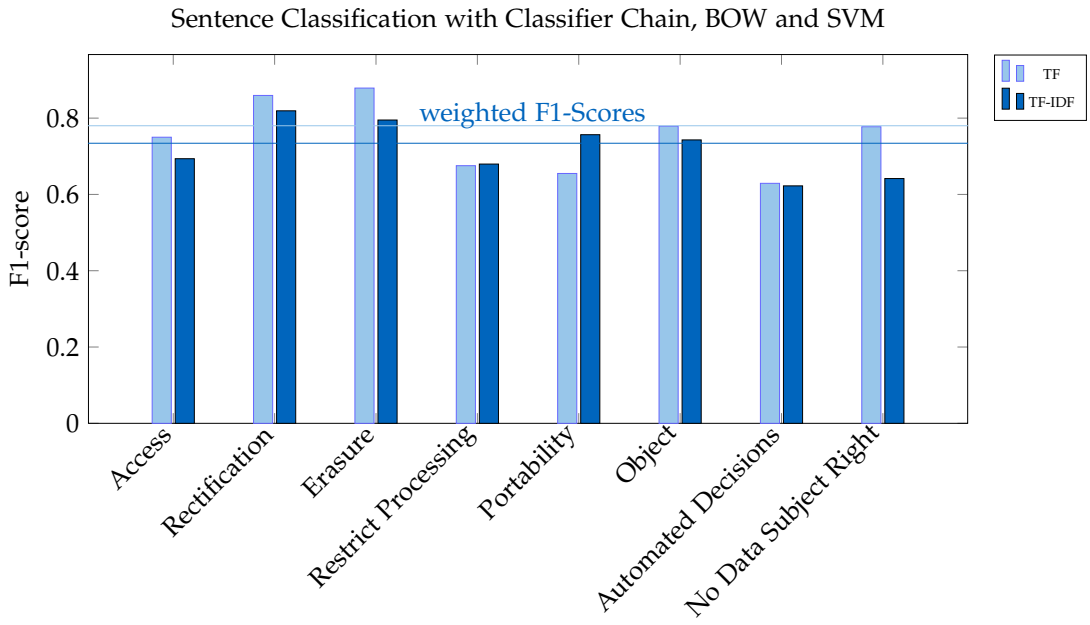


Figure 6.4: F1-Score results using BOW on initial dataset on lowercase input text, after stemming with the best performing problem transformation and classifier combination Classifier Chain and SVM based on TF and TF-IDF encodings

Source: Own illustration

the highest values for the different encodings. As shown in the graphic, adding POS tags did not improve the classification. The best value based on word embeddings was 38.3% and was reached using GloVe weighted with TF-IDF. Figure 6.6 shows an overview of the achieved F1-scores on the initial dataset using different combinations of pre-processing methods and the better performing TF-IDF encoding which shows that using lowercasing and stemming slightly outperforms the other combinations.
The performance on the individual classes is presented in Figure 6.7. Here, the lowest values with F1-scores around 60% are achieved for the classes "Restrict Processing" and "No Automated Decisions" which are among the classes with the lowest number of training samples.

In conclusion, the best classification result on the initial dataset was 79%, achieved by the classifier chain together with the SVM classifier, based on TF. The most effective pre-processing methods were lowercasing and stemming the corpus. In contrast, the

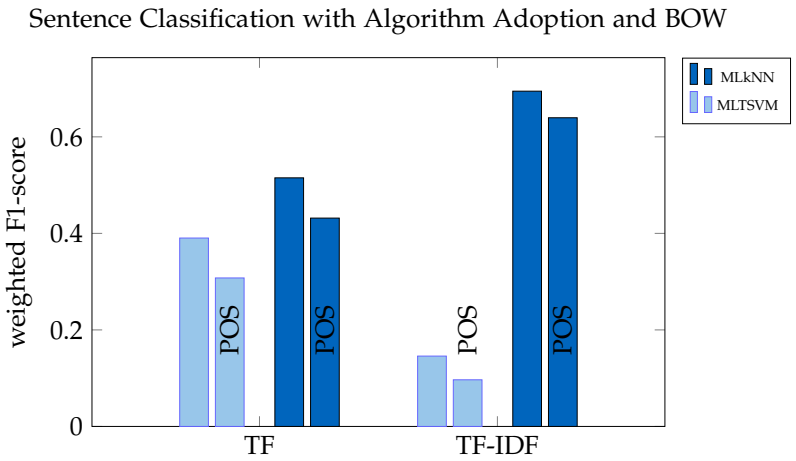


Figure 6.5: Maximum weighted F1-score results, using TF and TF-IDF on initial dataset on lower case input text, after stemming with MLkNN or MLTSVM and with or without usage of POS tags

Source: Own illustration

best result achieved by the algorithm adoption methods was ten percent lower and reached by MLkNN with k=3. To see if we can improve the classification results, we extended our dataset by roughly 700 sentence annotations.

Extended Dataset The extended dataset contains 1357 labeled sentences for the seven data subject rights ranging between 267 labels for the "right to access" and 60 for the "right not to be subject to automated decisions". We also extended the sample size for the null-class, now containing 200 examples of sentences of privacy policies not describing any data subject right. These 1557 sentences are used to train and evaluate our ML classification algorithms again.

Problem Transformation When looking at the word embeddings, the general distribution of performance looks similar to the problem transformation using the initial dataset. This means that the best results were achieved using the GloVe embeddings with TF-IDF weights in combination with the label powerset transformation and the SVM on the lowercase corpus not containing any stopwords. Figure 6.8 gives an overview of the highest achieved weighted F1-score on the initial dataset and on the

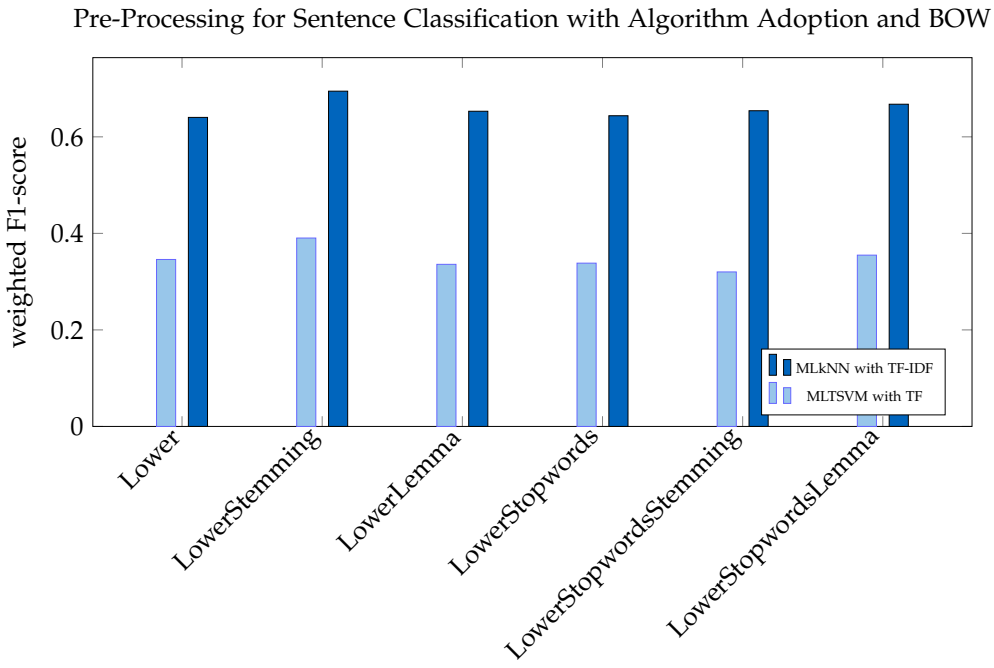


Figure 6.6: Maximum weighted F1-Score results on initial dataset, using MLkNN with TF-IDF encodings or MLTSVM with TF encodings depending on text pre-processing

Source: Own illustration

extended dataset, showing that all embeddings lead to better results (increase between four and ten percent), but the selftrained vectors are still outperformed by the pre-trained models.
Considering the BOW models, again the general distribution is similar to the initial dataset. Classifiers working on TF perform slightly better than on TF-IDF and POS tags do not improve the results. The highest weighted F1-score of 86.2% is achieved by applying the classifier chain in combination with the SVM on the lowercase text corpus, after removing the stopwords. This is the highest score our ML methods reached and thus the model which is used in the tool when selecting the ML classifier. Consequently, extending the dataset could increased the highest weighted F1-score by seven percent. For this model, Figure 6.9 shows the F1-scores per class, as well as precision and recall for the null class. This shows the reliability of the tool when choosing the ML classifier.

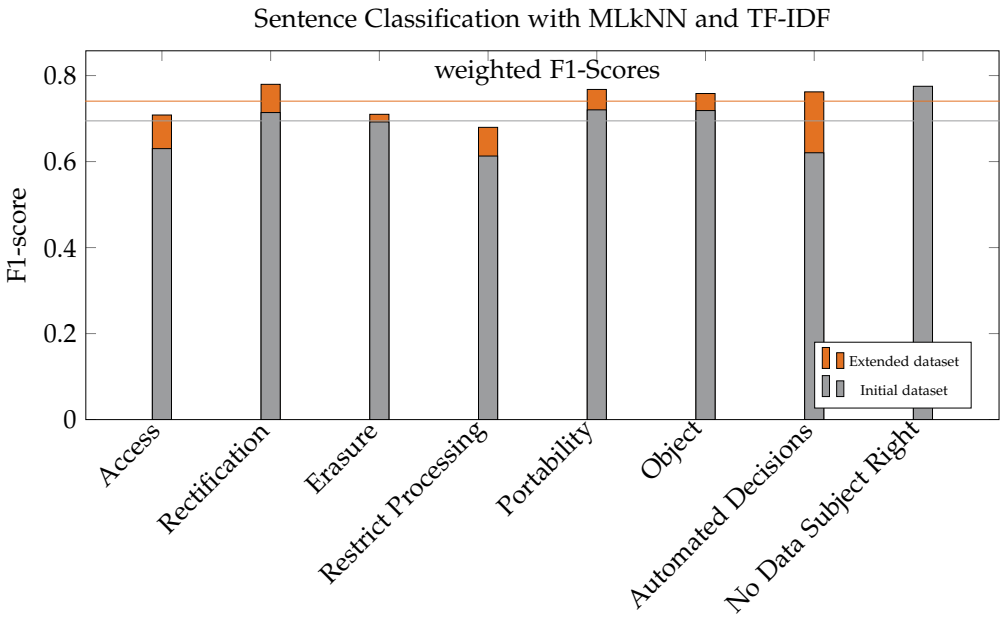


Figure 6.7: F1-Score results for each class on lowercase input text, after stemming with the MLkNN classifier, based on TF-IDF encodings

Source: Own illustration

The precision of 77.8% for the null class means that 78% of the not marked sentences do not contain any data subject right. Whereas the recall of 86.5% describes the amount of sentences not describing a right which are not marked in the tool.

When using BOW models, the best ML classifier for our problem is the SVM, whereas when using word embeddings, the tree classifiers perform best. Logistic Regression could not outperform any of the other classifiers. Regarding the problem transformation methods, there is no clear best, since depending on the combination of pre-processing, features and classifier, always a different one reached the highest results.

Algorithm Adoption The MLTSVM could improve the highest weighted F1-score on the extended dataset by almost ten percent from 39% to 48.1%. Here the best score was reached using the lowercase corpus after removing stopwords and stemming based on TF. TF-IDF reduces the quality by more than ten percent and adding POS labels performed around two percent worse.

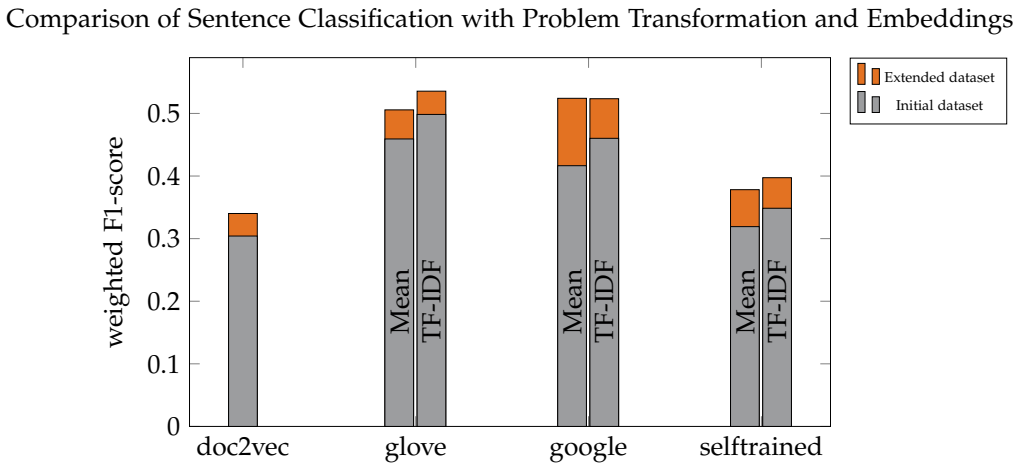


Figure 6.8: Comparison of maximum weighted F1-score results, using word embeddings on initial and extended dataset on lower case input, after removing stopwords

Source: Own illustration

Regarding the MLkNN, the weighted F1-score could be increased by five percent and reaches 74.1%. These results, as well as the scores for the individual classes are shown in Figure 6.7, in comparison to the results on the initial dataset. The value for the null class did not change significantly. When using the larger dataset, the algorithm performed best using five neighbors instead of three, compared to the initial dataset. Working on TF-IDF outperformed the results of TF by five to ten percent. Moreover, adding POS labels decreased the prediction accuracy.

In conclusion, problem transformation methods achieved better results than algorithm adoptions. Furthermore, adding POS tags the way we did, could not increase the prediction accuracy for both, algorithm adoption and problem transformation. In general, the F1-score is dependent on the number of labeled instances per class and in the matter of pre-processing combinations, it is not possible to point out a clear best. They lead to comparable results, but the highest scores were achieved using lowercasing together with stopword removal or stemming.

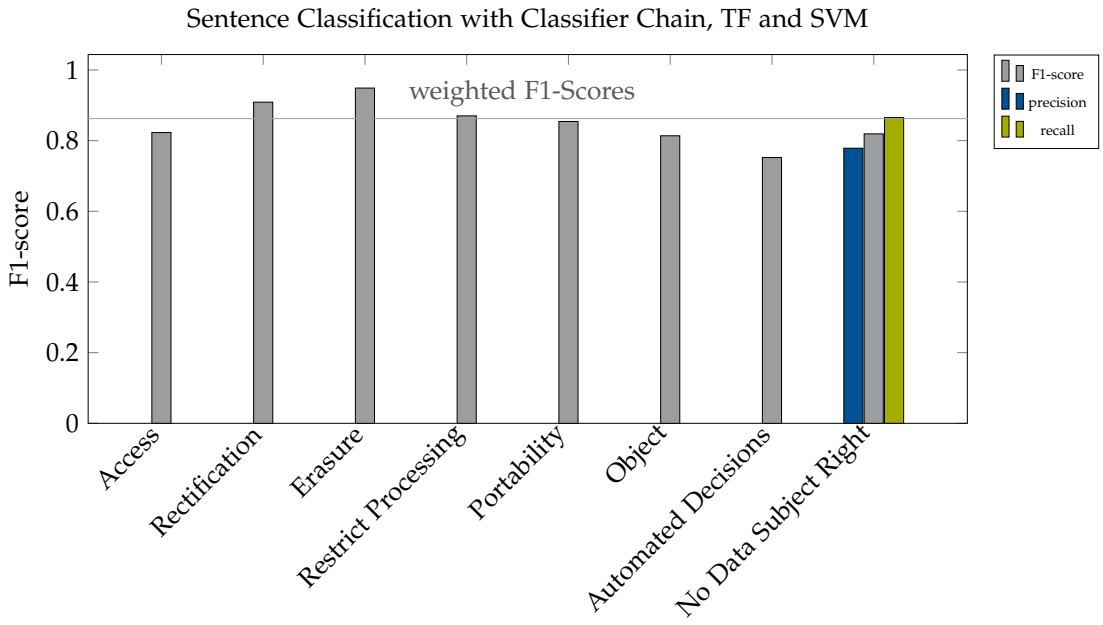


Figure 6.9: Results for each class using TF encodings on extended dataset on lowercase input text after stopword removal, using the combination of classifier chain and SVM

Source: Own illustration

6.2.2 Sequence Labeling

This section shows the results of the CRF as sentence classifier. It gives an overview of the impact of different pre-processing steps and features, such as POS, prefix and suffix, or the tokens surrounding which are described in Chapter 5.2.2. Equally to the sentence classification section, we present the impact of the dataset extension.

Initial Dataset As depicted in Table 5.1, the initially provided dataset contains 837 token based class labels. The numbers per class lie between 26 for the "right not to be subject to automated decisions" and 158 for the "right to rectification" and the "right to erasure". This was extended by 100 sentences without any relevant token label.
The best weighted F1-score of 74.4% is reached using all features on the corpus without stopwords and after lemmatization, but without lowercasing the text. This way, the feature if a token is upper or lower case can add additional information.

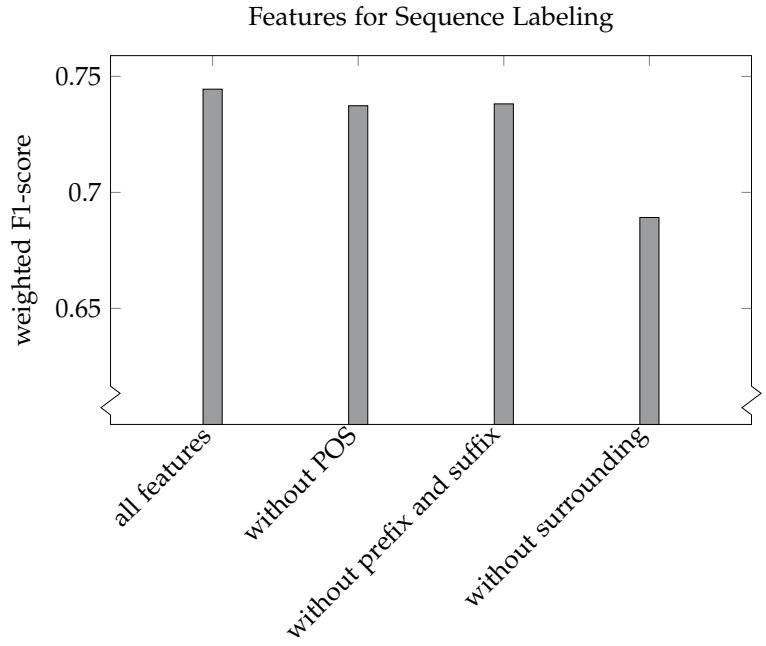


Figure 6.10: Maximum weighted F1-Score results of CRF on initial dataset after stopword removal and lemmatization depending on included features

Source: Own illustration

Figure 6.10 gives an overview of the effect of the different implemented features on the classification results. The values depicted are based on the corpus with removed stopwords and lemmatization, since in this case the CRF reached the best scores. The distribution looks similar for a differently processed texts. As clearly shown in the graph, using all features leads to the best results. POS tags and the prefix and suffix information do not yield high losses in the results, whereas dropping the information obtained by the surrounding tokens decreases the resulting value by sic percent.
Figure 6.11 depicts the best results of all different pre-processing combinations, showing that removing stopwords in combination with lemmatization performs only slightly better than the combination of lowercasing, removing stopwords, and stemming. In general, the scores depending on the different text pre-processing combinations only differ up to five percent.
An overview of the F1-scores of the individual classes is shown in Figure 6.12 together with the weighted F1-score of 74.4% in form of the horizontal line. The score varies

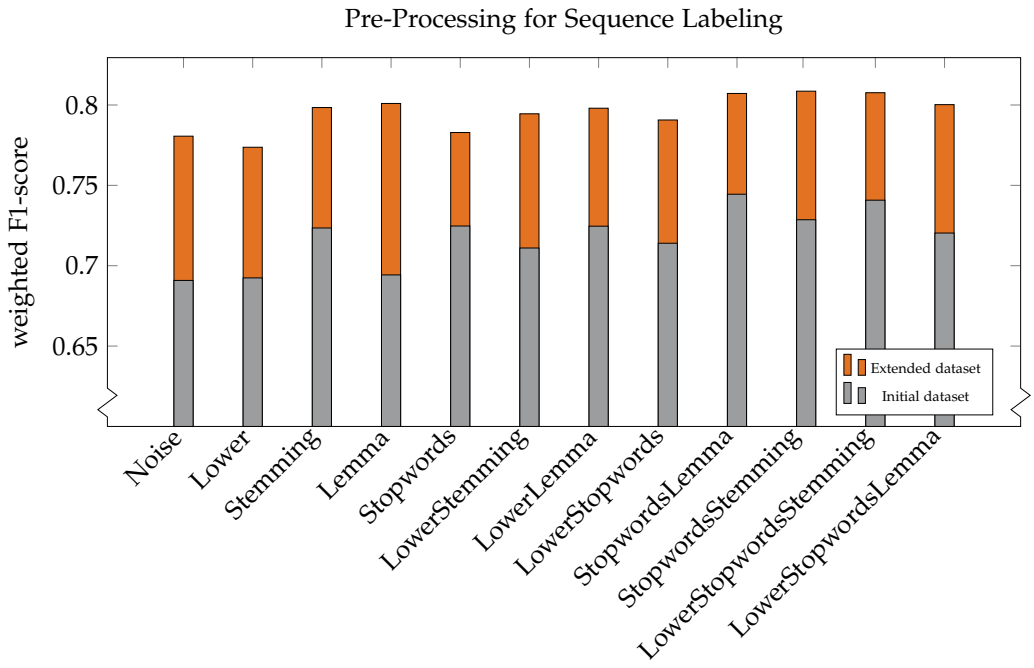


Figure 6.11: Maximum weighted F1-Score results using CRF depending on text pre-processing combinations

Source: Own illustration

between the individual classes which can be explained by the unbalanced dataset. A similar deviation can be observed over the range of feature and pre-processing combinations.
The best classification result using the CRF on the initial dataset is 74.4%. To see if these results improve by providing the algorithm with a larger training dataset, we added around 900 annotated tokens.

Extended Dataset The extended dataset contains 1764 labeled tokens for the seven data subject rights ranging between 331 labels for the "right to rectification" and 81 for the "right not to be subject to automated decisions". We also extended the sample size for the null class, containing now 200 examples of sentences of privacy policies, not describing any data subject right and containing only irrelevant "I" labels. This extended corpus is used to train and evaluate the CRF again.

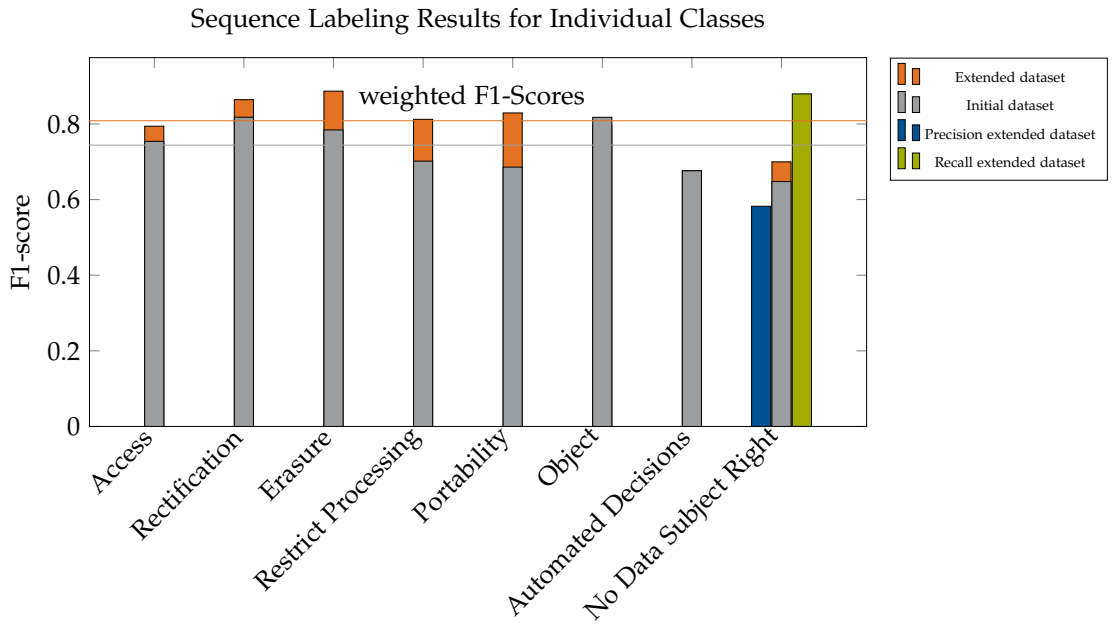


Figure 6.12: Best F1-Score results for each class using CRF, using all features on initial and extended dataset

Source: Own illustration

Figure 6.11 shows the difference in the weighted F1-score between the initial and extended dataset which lies in between 5.8% for the corpus with removed stopwords and 10.6% for the lemmatized corpus. The maximum weighted F1-score lies at 80.8% which is around sic percent higher than on the initial dataset. This value is reached by the combination of stopword removal and stemming, and the usage of all available features. This is the classification sequence used in the analysis tool when selecting CRF as classifier. The score distribution over the different classes is similar to the initial dataset and can be seen in Figure 6.12. Regarding the null class, in the case of sequence labeling the F1-score lies at 69.9%, whereas the precision is 58.2% and the recall 87.9%.

Using the CRF model the best results were achieved involving all features and the extended dataset with removed stopwords and lemmatization. Including the tokens surrounding contributes more valuable information than POS tags and considering the prefix and suffix of the word.

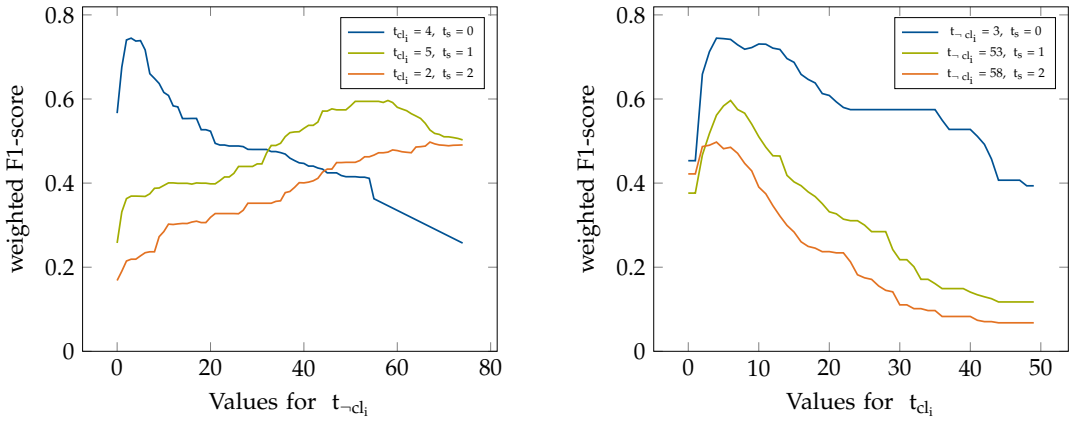
6.2.3 Rule Based Model

This chapter gives an overview of the quantitative performance of the implemented rule based classification algorithm. We show the impact of different pre-processing steps and how we determined the three utilized thresholds to create the dictionary of keywords. Again, we show the impact of the dataset extension and present the final rule based model used in the analysis tool.

Initial Dataset The initial dataset we used is the same as for the sequence labeling approach, since the dictionary is build on the token based labels. However, we do not need to consider the sentences for the null class for creating the dictionary, since the algorithm creating the dictionary only takes relevant labels (not "I") into account. We only used the null-class during the evaluation.
In order to determine the best combination of thresholds for the keywords to put in the dictionary, we tested possible combinations. The best weighted F1-score was achieved after performing lowercasing, stopword removal, and stemming and lies at 74.5%. The corresponding values for the thresholds are:

$$t_{cl_i} = 4, \quad t_{\neg cl_i} = 3, \quad t_s = 0$$

Figure 6.13a shows the development of the resulting weighted F1-score, depending on a changing value for $t_{\neg cl_i}$. The blue line depicts the case with the best classification described before. The values for $t_{cl_i} = 4$ and $t_s = 0$ are fixed . The curve is peaking around the value $t_{\neg cl_i} = 3$. The orange and green line show the development when using the values one and two for the threshold t_s with the best corresponding t_{cl_i}. As depicted, classification works not as good for a higher value for t_s and the optimum is reached for a much higher value for $t_{\neg cl_i}$. This means that much more tokens are added to the dictionary in order to mark more words in a sentence with the corresponding label and cross the value for t_s.
Figure 6.13b shows the development for the weighted F1-score again, but this time dependent on different values for t_{cl_i} and fixed values for the other two thresholds. Again, the blue line describes the case when using $t_s = 0$ which works best for the classification task. After the peak at four or the corresponding value for $t_s = 1$ and $t_s = 2$ respectively, an increasing value for t_{cl_i} decreases the added tokens in the dictionary and thus the resulting F1-score. In general, using a value higher than zero



(a) Effect of $t_{\neg cl_i}$ on the resulting weighted F1-score when fixating t_{cl_i} where classification works best

(b) Effect of t_{cl_i} on the resulting weighted F1-score when fixating $t_{\neg cl_i}$ where classification works best

Figure 6.13: Development of weighted F1-score depending on the defined thresholds based on initial dataset

Source: Own illustration

for the third threshold t_s yielded lower results. This means, this threshold can be neglected when applying the created dictionary for the classification task and every sentence containing one or multiple keywords for a specific class is labeled accordingly.

Table 6.1 shows the dictionary created using the best performing thresholds. It depicts the keywords per class and the corresponding count of occurrences. The "right to access" is represented only by the two keywords "access" and "review", whereas for instance the "right to rectification" contains seven keywords.

Figure 6.14 shows the best weighted F1-score results based on different pre-processing combinations. Here, the best results are achieved using all implemented pre-processing steps to decrease the size of the corpus and thus the size of the dictionary.

Figure 6.15 gives an overview of the F1-score results on the individual classes. For the class "Not to be subject to automated decisions" the support was zero, thus the F1-score is zero as well. This is owed to the low number of samples in the dataset. Noteworthy is the fact that the number of keywords does not directly correlate with the F1-scores reach for a class.

{Access:	{"access": 58,	"review": 8},
Rectification:	{"updat": 17,	"correct": 40,
	"rectif": 18,	"edit": 5,
	"incorrect": 7,	"rectifi": 7,
	"incomplet": 6},	
Erasure:	{"delet": 57,	"erasur": 16,
	"account": 8,	"eras": 13,
	"remov": 6},	
Restrict Processing:	{"restrict": 42,	"limit": 5}
Data Portability:	{"format": 14,	"portabl": 16,
	"structur": 8,	"commonli": 5,
	"machin": 10,	"readabl": 10,
	"transmit": 4,	"transfer": 4},
Object:	{"object": 56,	"withdraw": 14,
	"consent": 9,	"opt": 16}
Automated Decisions:	{"decis": 9,	"subject": 4,
	"base": 6,	"sole": 5,
	"autom": 8}}	

Table 6.1: Automatically generated dictionary on initial dataset

Extended Dataset The dictionary based one the extended dataset performs better when using $t_s = 0$ than the one created using the initial dataset. The best result achieved lies at 79% created on lowercase data without stopwords and after stemming. When using lemmatization instead of stemming, the results are comparable. This score is around five percent higher than on the initial dataset and is reached with

$$t_{cl_i} = 24, \quad t_{\neg cl_i} = 19$$

These thresholds are higher than in the initial model which can be explained by the higher number of keywords in the dictionary gained from the higher amount of input data. However, when using $t_s = 1$ or $t_s = 2$, the rule based classification using the larger dataset reaches lower scores than using the initial one. Figure 6.16 shows this development considering the influence of thr thresholds t_{cl_i} and $t_{\neg cl_i}$. Similar development can be observed using all different pre-processing combinations.
Table 6.2 shows the dictionary created using the best performing thresholds and the extended dataset. It depicts the keywords per class and the corresponding count of

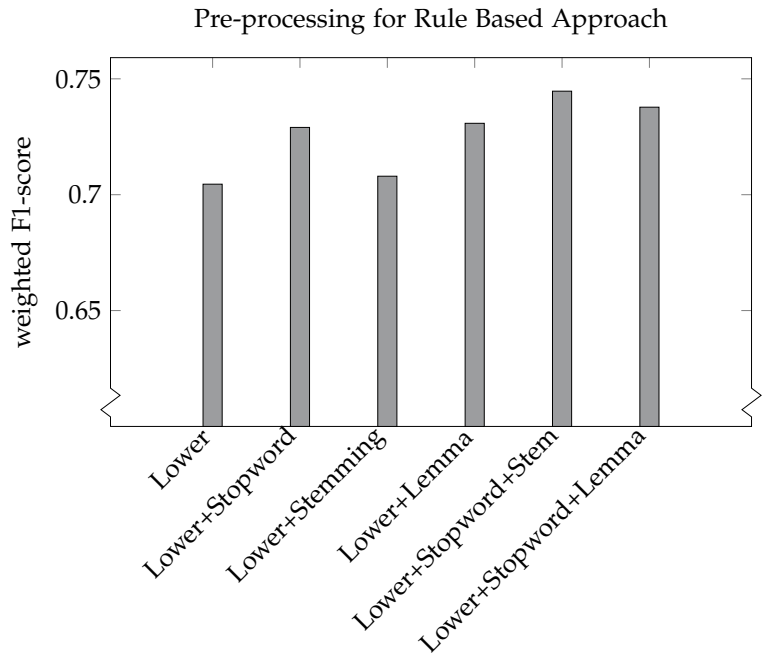


Figure 6.14: Maximum weighted F1-Score results on initial dataset using the rule based approach depending on different text pre-processing methods

Source: Own illustration

occurrences. This dictionary contains keywords with a larger wordcount than the dictionary created on the initial dataset. It also contains less keywords overall which is caused by the higher thresholds, sorting out less representative tokens with a lower number of occurrences. Figure 6.15 gives an overview of the F1-score results on the individual classes. For the class "Not to be subject to automated decisions" the F1-score is zero, which is caused by a support of zero, similarly as for the initial dataset. As shown in the graphic, increasing the amount of data did not have a positive impact on the results of all classes.

The analysis tool uses the depicted model with the F1-score of 80.1% when choosing the rule based option for text classification and analysis. For the null class, the reached score for precision lies at 52.4% and for recall at 75.4%.

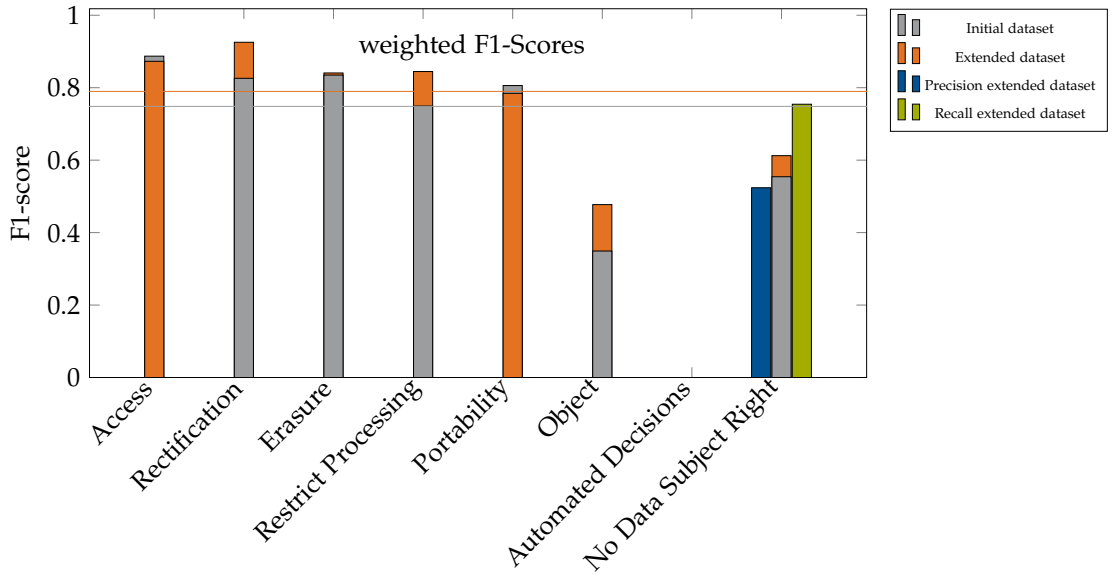
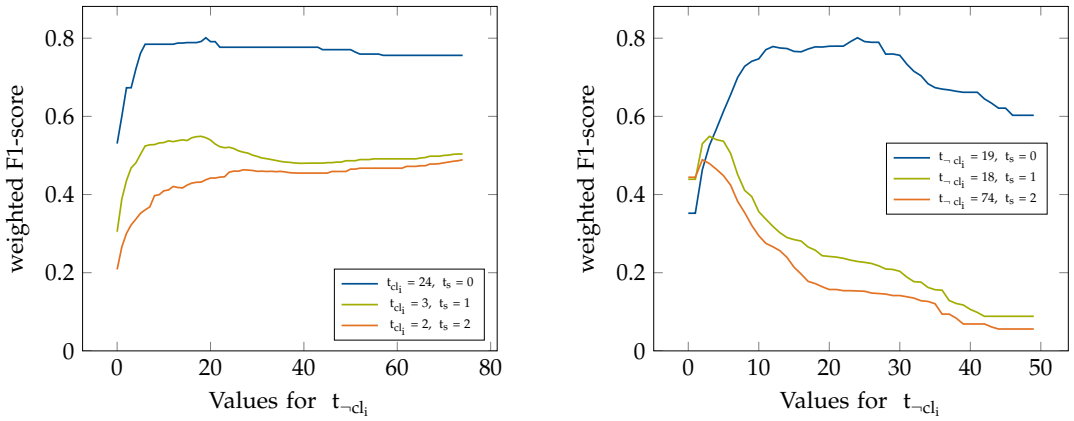


Figure 6.15: Best F1-score results for each class using the rule based approach and the extended dataset

Source: Own illustration

{Access:	{"access": 58,	"copy": 29},
Rectification:	{"correct": 76,	"update": 37,
	"rectification": 30,	
Erasure:	{"delete": 83,	"erasure": 34,
	"deletion": 25,	"erase": 24,
Restrict Processing:	{"restriction": 34,	"restrict": 58}
Data Portability:	{"portability": 32,	"structured": 38,
	"commonli": 24,	"used": 24,
	"machin": 42,	"readabl": 42,
	"format": 45,	"another": 35}
Object:	{"object": 112},	
Automated Decisions:	{"decision": 29,	"automated": 25,
	"base": 6,	"sole": 5}}

Table 6.2: Automatically generated dictionary on extended dataset



(a) Effect of $t_{\neg cl_i}$ on the resulting weighted F1-score when fixating t_{cl_i} where classification works best

(b) Effect of t_{cl_i} on the resulting weighted F1-score when fixating $t_{\neg cl_i}$ where classification works best

Figure 6.16: Development of weighted F1-score depending on the defined thresholds based on extended dataset

Source: Own illustration

6.3 Comparison of Different Methods

The previous sections depicted the results of the three different methods for extracting the data subject rights by classifying the sentences in a privacy policy. In this section, we compare these results to each other and work out differences of the best performing classifier of each method. For the traditional sentence classification, this was the combination of classifier chain and SVM on lowercase text after removing stopwords based on TF encodings. The CRF performed best, using all available features after stopword removal and stemming and the dictionary we use for comparison was created using $t_{cl_i} = 24$ and$t_{\neg cl_i} = 19$ after stopword removal and stemming.
In Figure 6.17, the highest reached weighted F1-scores for each of the mentioned approaches is presented. The results were the best for the sentence classification using a problem transformation approach and the SVM classifier. Using the CRF for sequence labeling reached a six percent lower score, whereas the dictionary performed eight percent lower.
For the two learning approaches, using problem transformation and a single label

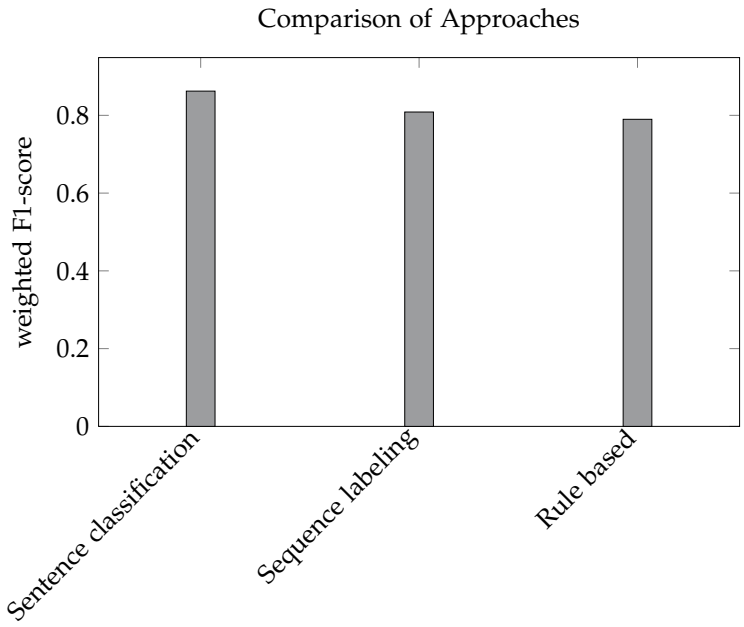


Figure 6.17: Comparison of highest weighted F1-scores reached by each of the implemented approaches

Source: Own illustration

classifier, as well as for sequence labeling using CRF, we examined the learning rate depending on the number of training samples. For this purpose, we used the extended dataset containing around 1200 sentences and reduced the number of samples in steps of ten percent of 1200. This was achieved by randomly choosing a subset with the desired size and performing the training and testing on this subset. Similar to the 5-Fold cross validation, this procedure was repeated five times in order to evaluate a possibly resulting bias by randomly choosing a subset. The results are depicted in Figure 6.18. As the plot suggests, in the case of sequence labeling the curve looks like it is stagnating, whereas the slope is still positive for the curve of the sentence classification method. This means that increasing the number of training samples could still improve the results of this classifier. It is notable that the CRF approach works better for a small amount of data than the other method, reaching a weighted F1-score of 70% on only 100 training samples.
Regarding the reliability of the three approaches concerning the recognition and

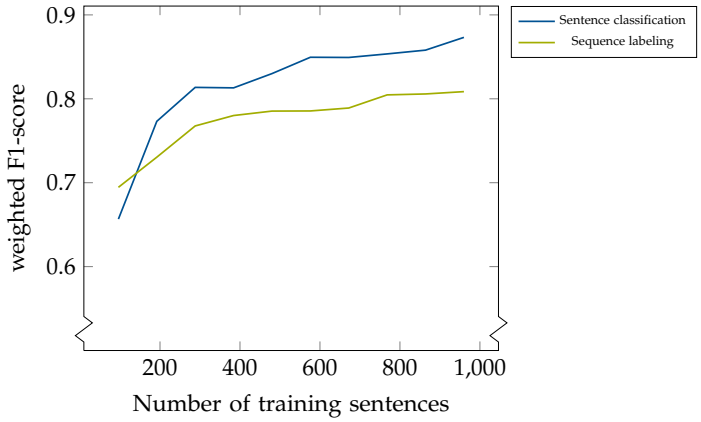


Figure 6.18: Effect of number of labeled sentences is the training dataset

Source: Own illustration

extraction of data subject rights, we examine the precision and recall values reached for the null-class. A high value for precision means that a small amount of sentences containing a data subject right are not recognized and thus not marked in the tool. However, a high recall value stands for a low percentage of sentences, not containing a right, being falsely extracted and marked in the tool. Figure 6.19 gives an overview of the highest reached values for precision and recall on the null-class by the three implemented approaches which represents the performance of the classification back-end of the analysis tool. The highest recall value of 87.9% is reached by the CRF, closely followed by the sentence classification using the SVM with 86.5%. This means, in these two cases only 12%, or 13.5% respectively of sentences not containing any rights are highlighted. For the dictionary approach this value lies at around 25%. Considering the precision, the highest score is 77.9% and is reached by the sentence classification method. Consequently, choosing this classifier, around 22% of sentences in the testset marked as not relevant did actually contain a data subject right. For the two other approaches, the precision values lie at 58.2% for the CRF and 52.4% for the dictionary approach.

In summary, the highest reached weighted F1-scores of the three approaches differs by at most eight percent, whereby the highest value is achieved using sentence classification with problem transformation and the SVM classifier. The lowest value is reached by the rule based approach. When looking at the precision and recall of the null-class, the

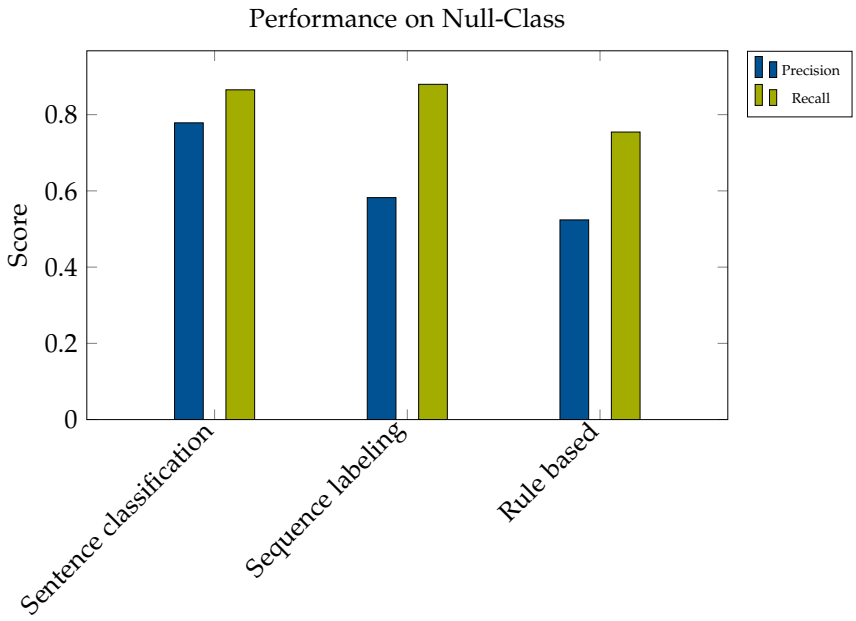


Figure 6.19: Comparison of highest reached precision and recall value for the null-class not containing any data subject right

Source: Own illustration

sentence classification reaches the highest reliability, again. When choosing CRF or the rule based approach, a higher number of sentences containing a data subject right will be unrecognized.

6.4 Evaluation of Initial Classes

In this section, we shortly describe the quantitative results of our implemented classification algorithms on the initially chosen classes, especially the two classes "data subject rights paragraph" and "data subject rights title". These two classes can be relevant for future work, when more content of a privacy policy is to be analyzed and a preliminary structuring of the text can be helpful. This evaluation is based on the initial dataset, since during the annotation in the course of this thesis, these two classes were neglected, because they were irrelevant for the designed tool. Besides that, we only evaluated these classes using the sentence classification method, as this approach was the most

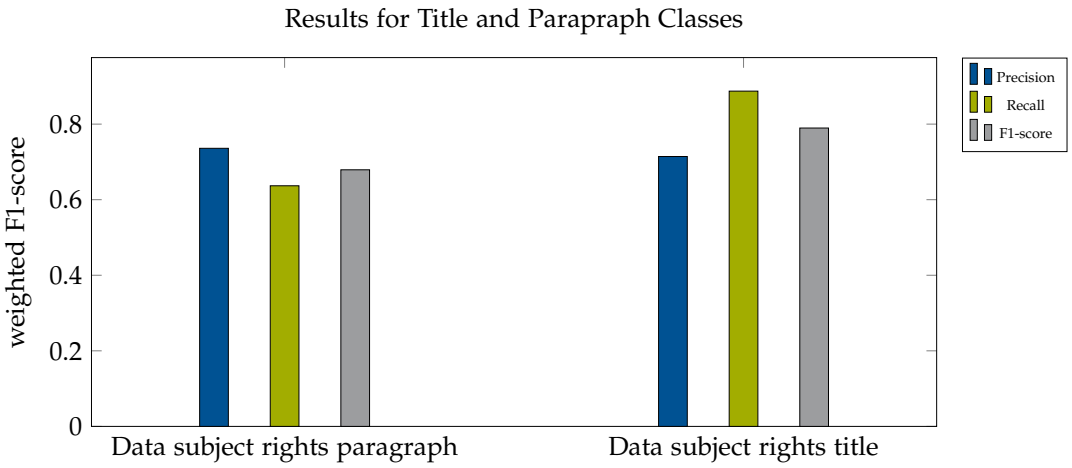


Figure 6.20: Comparison of highest reached F1-score, precision, and recall values for the data subject right title and paragraph classes, using sentence classification with problem transformation and the SVM classifier

Source: Own illustration

promising during the previous evaluation.
Figure 6.20 depicts an overview of the according scores. As shown in the graph, for the "data subject rights title", a high recall value of 88.7% was achieved on the initial dataset. That means that 88.7% of all sentences labeled as "data subject rights title" were recognized as such. In contrast only 63.7% of all "data subject rights paragraph" instances were recognized. The precision was comparable for both of these classes at around 70-75%, whereas the F1-score with 78.9% was higher for "data subject rights title".

These numbers are promising for further research regarding segmentation of privacy policy text, before starting the actual sentence classification. Especially the "data subject rights title" is recognized with a high probability and can provide insight into the document structure.

7 Discussion and Future Work

In this chapter we will review the results of this thesis and elaborate on the research questions and how they were reflected upon. We will discuss limitations of the developed solutions and suggest possible next steps and future works to eliminate or improve on these limitations.

7.1 Discussion

In this section we recap the research questions defined in Chapter 1 and discuss the acquired insights on them. This way, we will give an overview of the results of this thesis, including developed solution approaches and discuss their strengths and limitations.

RQ1: Which approaches exist to automatically analyze legal texts and privacy policies?
Chapters 2 and 3 explain the foundations of automatically analyzing texts, as well as related work on privacy policies using ML and rule based approaches. They follow the workflow of CATA using different pre-processing techniques and various features, such as POS tags and Named Entity Recognition. There is a lot of research going on using supervised ML, rule based and combined approaches to extract information of privacy policies. Techniques used are decision rules, pattern matching, and different ML classifiers like Naive Bayes or Random Forest. Besides these automated analysis, there exist tools to visualize the content of privacy policies dependent on information provided by the website owner.

RQ2: Are supervised machine learning methods suitable to analyze privacy policies with regard to the coverage of the data subject rights?
The approaches to automatically analyze privacy policies presented in Chapter 3 mostly contain supervised ML in some form, sometimes in combination with decision rules. Also, the presented solutions show promising results on extracting information from privacy policies. The classification algorithms implemented in this thesis, with problem

transformation and a traditional classifier, as well as the sequence labeling using CRF, show promising results on recognizing the data subject rights. The best approach extracts 77.9% of the data subject rights correctly and only 13.5% of the highlighted sentences are not relevant. Besides that, the ML approaches outperformed the also tested rule-based classification and can possibly be improved with a larger and more balanced dataset.

RQ3: What are the strengths and weaknesses of sentence classification and sequence labeling based approaches?
Sentence classification using sequence labeling with CRF is able to take contextual information into account, since it models the whole sentence and we can add the surrounding words as features, having great impact on the classification results (as described in Section 6.2.2). This is not possible for the presented ML methods using traditional classifiers. Here, the best performing representation is the TF which describes the number of occurrences of a token and does not consider the context. Besides that, the CRF is able to reach quite high scores on a low number of training samples. On only 100 presented sentences it reached a weighted F1-score of around 70%. The classical ML methods only reach a weighted F1-score of around 65% for the same amount of training data. However, the classical ML methods improved their performance with increasing number of samples more than the CRF, ending up with better results on the whole dataset used in this thesis. The recognition distribution over the individual classes is similar for both ML approaches and suggest that they are sensitive to the imbalance in the dataset.

RQ4: Which supervised machine learning approach performs best at extracting data subject rights from privacy policies?
The best performance on both, the initially provided dataset and the extended version, is reached using a problem transformation method together with a traditional ML classifier. It reaches a maximum weighted F1-score of 86.5%. However, the highest scores reached by the sequence labeling and the rule based approaches lie only six and eight percent below this value. Regarding the learning curve depending on the number of training samples, the problem transformation method together with a traditional ML classifier looks more promising when increasing the amount of training data even more than the CRF.

RQ5: Which performance can be achieved on the test data and can the automated

extraction add value for private consumers regarding the understanding of privacy policies?
As mentioned before, the best weighted F1-score of 86.5% is reached by applying the classifier chain problem transformation together with the SVM classifier. Also interesting is the performance on the newly added null-class, since it can provide insight into the reliability of the classification methods when analyzing a whole policy. It is not only important that the algorithm can differentiate between the different data subject rights, but it needs to be able to separate sentences not containing any right from the interesting ones. This is the only way for the algorithm to be able to extract a high percentage of data subject rights correctly, without marking many sentences not containing any right. This property can be described by the precision and recall values achieved for the null-class. For the precision, the sentence classification using SVM reaches the highest score with 77.9% in comparison to 58.2% for the CRF and 52.4% for the rule based method. The highest recall for the null-class lies at 87.9% and is reached by the CRF. The problem transformation method together with the SVM classifier achieved a score of 86.5%. This means that less than 13.5% of the sentences in the testset, not containing any right, are falsely classified as positive. The recognized and highlighted passages provide guidance for the user and show sections, where data subject rights, making the scanning process of the document easier. For clearly stating the added value for the end-user, interviews with domain experts need to be conducted in order to judge the performance and usefulness of the implemented tool.

7.2 Future Work

An outlook on possible subsequent future work is given in the following paragraphs, elaborating on points that remain unresolved by this thesis and suggest improvements to the limitations of the implemented approaches.

As mentioned in the previous section, a structured qualitative evaluation of the implemented tool is necessary to assess the quality and helpfulness, but was not in the scope of this thesis. To evaluate the performance of the tool and the reliability in recognizing the correct sentences, interviews with domain experts need to be conducted. Interviews with non-expert users can be another helpful method to evaluate the structure and clarity of the UI. The gained information will provide insight to what extent the non-

functional requirements, regarding accuracy and understanding of the text compared to a non-expert user.
The findings during the thesis suggest several procedures to further improve the quantitative results of the different classification algorithms. An extensive parameter tuning can be performed, maximizing the full potential of the individual classifiers. Additionally, combinations of the three described methods could be implemented to make use of each of their strengths. Also, domain experts annotating and increasing the number of training samples and creating a balanced dataset is promising to increase the performance. As the results of this thesis show, the classification accuracy is lower for classes with little training data and expanding the dataset by a non-expert could increase the overall results. Considering an imbalanced dataset, for the rule based classification method in particular, different thresholds for the individual classes could help the classification, since the count of occurrences for keywords for a class with low numbers of samples is lower than for a class with many training samples.
Unfortunately, we did not fulfill all specified optional requirements for the UI. To meet all of them, a functionality to add a privacy policy as PDF file or per URL needs to be added. Not entering the text directly involves changes to the processing of the policy, since the raw text needs to be extracted from the PDF or the website.
As already discussed, the initial dataset contained labels for the two classes "data subject rights title" and "data subject rights paragraph" which can be relevant for future work. As we have seen, the reached classification scores, especially for "data subject rights title", were promising. These two classes can be used to pre-structure the policy to reduce the amount of text that needs to be analyzed by the specialized classifier (for data subject rights or any other content of the privacy policy), when widening the area of application to more contents of privacy policies.

8 Conclusion

In this thesis, a tool was designed, developed, and evaluated, assisting the internet user to understand privacy policies of service providers and recognize contained data subject rights, using an annotated dataset of privacy policies. The specified requirements for this policy analysis tool are based on literature research regarding related solution approaches for extracting content of privacy policies and presenting the results to the user. The three implemented solutions using traditional ML classifiers, sequence labeling for sentence classification, and a rule based approach are also found in related work regarding privacy policy analysis and are foundational methods of text categorization. These three classification approaches together with text processing methods embedded in a locally running Flask system form the back-end of the implemented analysis tool. The UI was designed using insights gained during literature review and uses simple focal points and indicators to give an overview of the contained rights to the user.

Another research goal of this thesis was the evaluation of the suitability of supervised ML algorithms for analyzing privacy policies with regard to the coverage of data subject rights. Besides the fact of supervised ML algorithms being used for multiple related privacy policy analysis tools, the qualitative results of the implemented algorithms on the annotated dataset were very promising. The imbalance of the dataset was a problem for the accuracy concerning the individual classes with lower number of samples and thus influencing the overall score. Balancing and increasing the annotated dataset is assumed to further improve the overall performance, because of the observation of uneven recognition accuracy depending on the number of annotated samples and the fact that the classification improved on an extended dataset.

In conclusion, privacy policies are the most important source of information for internet users, interested in making informed decisions about the processing and protection of their personal data. Many studies showed the poor readability of these policies and the consequentially caused information overload. Thus, it is important to provide the users with tools to help them understand the policy content more easily and with

less expenditure of their time. The results of this thesis demonstrate the suitability of supervised ML algorithms to automatically analyze the privacy policy text and afterwards present the extracted contents to the user.

List of Figures

3.1 Extract of a P3P Expandable Grid 13
3.2 Extract of Privacy Nutrition Labels by Kelley et al. 14

4.1 UML use-case diagram for analysis tool GUI 21
4.2 UML deployment diagram for the analysis system 23
4.3 First prototypical draft of the UI 24

5.1 Structure of dataset in JSON-format 26
5.2 Geometric interpretation for SVM and Twin SVM 33
5.3 Geometric interpretation for MLTSVM 34
5.4 Graphical structures of sequence models 39
5.5 Final version of the UI after analyzing a policy 41
5.6 Components of the React Application 43

6.1 Maximum weighted F1-score results using embeddings on initial dataset on lower case input text, after removing stopwords with different problem transformation and classifier combinations 48
6.2 Maximum weighted F1-score results using TF and TF-IDF on initial dataset on lower case input text, after stemming with different problem transformation and classifier combinations and with or without usage of POS tags 49
6.3 Maximum weighted F1-score results using TF and TF-IDF on initial dataset with the best performing problem transformation and classifier combination depending on different text pre-processing methods 50
6.4 F1-Score results using BOW on initial dataset on lowercase input text, after stemming with the best performing problem transformation and classifier combination Classifier Chain and SVM based on TF and TF-IDF encodings 51

6.5 Maximum weighted F1-score results, using TF and TF-IDF on initial dataset on lower case input text, after stemming with MLkNN or MLTSVM and with or without usage of POS tags 52
6.6 Maximum weighted F1-Score results on initial dataset, using MLkNN with TF-IDF encodings or MLTSVM with TF encodings depending on text pre-processing 53
6.7 F1-Score results for each class on lowercase input text, after stemming with the MLkNN classifier, based on TF-IDF encodings 54
6.8 Comparison of maximum weighted F1-score results, using word embeddings on initial and extended dataset on lower case input, after removing stopwords 55
6.9 Results for each class using TF encodings on extended dataset on lowercase input text after stopword removal, using the combination of classifier chain and SVM 56
6.10 Maximum weighted F1-Score results of CRF on initial dataset after stopword removal and lemmatization depending on included features . 57
6.11 Maximum weighted F1-Score results using CRF depending on text preprocessing combinations 58
6.12 Best F1-Score results for each class using CRF, using all features on initial and extended dataset 59
6.13 Development of weighted F1-score depending on the defined thresholds based on initial dataset 61
6.14 Maximum weighted F1-Score results on initial dataset using the rule based approach depending on different text pre-processing methods . . 63
6.15 Best F1-score results for each class using the rule based approach and the extended dataset 64
6.16 Development of weighted F1-score depending on the defined thresholds based on extended dataset 65
6.17 Comparison of highest weighted F1-scores reached by each of the implemented approaches 66
6.18 Effect of number of labeled sentences is the training dataset 67
6.19 Comparison of highest reached precision and recall value for the nullclass not containing any data subject right 68

6.20 Comparison of highest reached F1-score, precision, and recall values for the data subject right title and paragraph classes, using sentence classification with problem transformation and the SVM classifier . . . 69

List of Tables

5.1 Labeled instances per data subject rights classes on two levels and three additional classes on sentence level for the initial and extended dataset 28
5.2 Illustration of the problem transformation methods binary relevance and label powerset 35

6.1 Automatically generated dictionary on initial dataset 62
6.2 Automatically generated dictionary on extended dataset 64

Bibliography

[1] J. R. Reidenberg, T. Breaux, L. F. Cranor, B. French, A. Grannis, J. T. Graves, F. Liu, A. McDonald, T. B. Norton, and R. Ramanath. "Disagreeable privacy policies: Mismatches between meaning and users' understanding." In: *Berkeley Tech. LJ* 30 (2015), p. 39.

[2] J. A. Obar and A. Oeldorf-Hirsch. "The biggest lie on the internet: Ignoring the privacy policies and terms of service policies of social networking services." In: *Information, Communication & Society* (2018), pp. 1–20.

[3] A. M. McDonald and L. F. Cranor. "The cost of reading privacy policies." In: *ISJLP* 4 (2008), p. 543.

[4] D. S. E. Kettner and P. D. C. Thorun. *PGuardGemeinsamer Abschlussbericht*. Bundesministerium für Bildung und Forschung, Institut für Angewandte Informatik e. V. (InfAi), mediaTest digital GmbH, Quadriga Hochschule Berlin and Selbstregulierung Informationswirtschaft e.V. (SRIW).

[5] S. Wilson, F. Schaub, A. Dara, S. K. Cherivirala, S. Zimmeck, M. S. Andersen, P. G. Leon, E. Hovy, and N. Sadeh. "Demystifying privacy policies with language technologies: Progress and challenges." In: *Proceedings of LREC 1st Workshop on Text Analytics for Cybersecurity and Online Safety*. 2016.

[6] A. Hotho, A. Nürnberger, and G. Paaß. "A brief survey of text mining." In: *Ldv Forum*. Vol. 20. 1. Citeseer. 2005, pp. 19–62.

[7] C. C. Aggarwal and C. Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[8] A. G. Jivani et al. "A comparative study of stemming algorithms." In: *Int. J. Comp. Tech. Appl* 2.6 (2011), pp. 1930–1938.

[9] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. "From word embeddings to document distances." In: *International Conference on Machine Learning*. 2015, pp. 957–966.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[12] J. Pennington, R. Socher, and C. Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[13] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[14] D. Jurafsky and J. H. Martin. *Speech and language processing*. Vol. 3. Pearson London, 2014.

[15] J. Lafferty, A. McCallum, and F. C. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." In: (2001).

[16] L. F. Cranor. "Necessary but not sufficient: Standardized mechanisms for privacy notice and choice." In: *J. on Telecomm. & High Tech. L.* 10 (2012), p. 273.

[17] A. M. Mcdonald, R. W. Reeder, P. G. Kelley, and L. F. Cranor. "A comparative study of online privacy policies and formats." In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2009, pp. 37–55.

[18] P. G. Kelley, L. Cesca, J. Bresee, and L. F. Cranor. "Standardizing privacy notices: an online study of the nutrition label approach." In: *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. ACM. 2010, pp. 1573–1582.

[19] *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. URL: `https://www.w3.org/TR/P3P/`.

[20] R. W. Reeder. *Expandable Grids: A user interface visualization technique and a policy semantics to support fast, accurate security and privacy policy authoring*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2008.

[21] P. G. Kelley, J. Bresee, L. F. Cranor, and R. W. Reeder. "A nutrition label for privacy." In: *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM. 2009, p. 4.

[22] E. Costante, Y. Sun, M. Petković, and J. den Hartog. "A machine learning solution to assess privacy policy completeness:(short paper)." In: *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM. 2012, pp. 91–96.

[23] W. Ammar, S. Wilson, N. Sadeh, and N. A. Smith. "Automatic categorization of privacy policies: A pilot study." In: *School of Computer Science, Language Technology Institute, Technical Report CMU-LTI-12-019* (2012).

[24] *TermsofService;Didn'tRead*. URL: `https://tosdr.org/`.

[25] W. B. Tesfay, P. Hofmann, T. Nakamura, S. Kiyomoto, and J. Serna. "PrivacyGuide: towards an implementation of the EU GDPR on internet privacy policy evaluation." In: *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*. ACM. 2018, pp. 15–21.

[26] E. Costante, J. den Hartog, and M. Petković. "What websites know about you." In: *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2012, pp. 146–159.

[27] H. Cunningham, H. Cunningham, D. Maynard, D. Maynard, V. Tablan, and V. Tablan. *JAPE: a Java Annotation Patterns Engine*. 1999.

[28] S. Zimmeck and S. M. Bellovin. "Privee: An architecture for automatically analyzing web privacy policies." In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 2014, pp. 1–16.

[29] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[30] M. F. Porter. "An algorithm for suffix stripping." In: *Program* 14.3 (1980), pp. 130–137.

[31] G. A. Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[32] Q. Le and T. Mikolov. "Distributed representations of sentences and documents." In: *International conference on machine learning*. 2014, pp. 1188–1196.

[33] G. Tsoumakas and I. Katakis. "Multi-label classification: An overview." In: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3 (2007), pp. 1–13.

[34] M.-L. Zhang and Z.-H. Zhou. "ML-KNN: A lazy learning approach to multi-label learning." In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[35] P. Szymański and T. Kajdanowicz. "A scikit-based Python environment for performing multi-label classification." In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.01460 [cs.LG].

[36] W.-J. Chen, Y.-H. Shao, C.-N. Li, and N.-Y. Deng. "MLTSVM: a novel twin support vector machine to multi-label learning." In: *Pattern Recognition* 52 (2016), pp. 61–74.

[37] J. Read, B. Pfahringer, G. Holmes, and E. Frank. "Classifier chains for multi-label classification." In: *Machine learning* 85.3 (2011), p. 333.

[38] G. Tsoumakas and I. Vlahavas. "Random k-labelsets: An ensemble method for multilabel classification." In: *European conference on machine learning*. Springer. 2007, pp. 406–417.

[39] *scikit-learn Machine Learning in Python*. 7–2019. URL: https://scikit-learn.org/stable/.

[40] J. J. Garrett et al. "Ajax: A new approach to web applications." In: (2005).

[41] *Model evaluation: quantifying the quality of predictions*. 7–2019. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics.