DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Management & Technology

An Expert-Defined Rule-Based Approach for Generating Vector Representations to Classify Texts

Andrei Kreinhaus

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Management & Technology

An Expert-Defined Rule-Based Approach for Generating Vector Representations to Classify Texts

Ein von Experten Definierter Regelbasierter Ansatz zur Generierung von Vektordarstellungen für die Klassifizierung von Texten

Author:	Andrei Kreinhaus
Matriculation nr:	03677710
School:	TUM School of Management
Study program:	M.Sc. Management & Technology
Supervising Chair:	Software Engineering for Business Information Systems
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Stephen Meisenbacher, M.Sc.
Submission Date:	15.12.2022

I confirm that this master's thesis in management & technology is my own work and I have documented all sources and material used.

Munich, 15.12.2022 Andrei Kreinhaus

Acknowledgments

As I write these lines, when all the work is done, I can't help but think of all the people who have left their positive footprints on my life and helped me become the person I am today. I am well aware that I cannot mention everyone here, but I want to try to acknowledge at least those who have brought life-changing experiences into my life.

First of all, I want to start expressing my gratitude to the most important people - my family.

My beloved mother Tamara Nemtsova, thank you for letting me into this world and giving me wings and complete freedom to live the way I want. Thank you for your acceptance. Thank you that your heart, soul, and doors are always open for me and that you are always ready to listen and support me, even if we are far away from each other.

My dear father Boris Kreinhaus, I thank you for your unconditional love and for all that you have been doing for me. Your attitude toward parents, children, and family in general has always been a great example to me.

My dear uncle Viktar Matsveyenka, thank you for all your wisdom and life lessons. You showed me chess, you showed me the world. Your tremendous care and support when I was a child and your readiness to always help me now are incredibly precious to me.

Next, I want to briefly talk about my amazing journey in Germany over the past 7 years, which led me to this research in the Department of Informatics.

Initially, with no plans to move abroad, it took one Olga to show me the opportunity to study in Germany (but could I really do this!?), and another Olga to personally submit all my documents to the admissions office just hours before the deadline. Special thanks to Vladimir for instilling in me a love of the German language and tirelessly sharing your knowledge. I hit the jackpot with your course, which allows me to feel in my element living in Germany. Also a huge thank you to Marina for accepting a literally stranger into your WG, and I'm so glad you're still around for me.

I can't stop thinking about all the twists and turns of life. Initially hesitant and ignoring everything about coding, I wanted to work my way toward a career as an entrepreneur through countless meetings, side projects, startup internships, and related Bachelor's thesis. My side hustles included volunteering at student clubs in marketing and finance, having nothing to do with the tech world. Then, in search of "fun", while locked down, I took a Python Bootcamp and received endorphin kicks after each successful program launch. Although I applied for internships in finance, later on, I got an interview for a technical

position and ended up getting my first hands-on experience with data wrangling. My passion for data has been thriving ever since.

I would like to express my enormous gratitude to all the people who have made the Technical University of Munich a place where I feel like I belong. Thank you for creating an atmosphere that allowed me to get the most up-to-date theoretical and practical knowledge, enjoy social events, occasionally bump into smart and inspiring students, and meet members of numerous student initiatives on campus! I want to give special thanks to the TUMexchange team for giving me two unforgettable opportunities to study, enjoy student life, and represent TUM in opposite parts of the world. By now I feel that I have had a good time as a student and feel mature enough to embark on the adventure of working life.

This section would not be complete without mentioning the people who made this research possible. I would like to thank Prof. Dr. Florian Matthes for supervising this thesis and giving me the opportunity to work in his department. My special thanks go to my advisor Stephen Meisenbacher for his 360 support of this thesis, including weekly meetings, brainstorming sessions, thought-provoking ideas, and no pressure on me! I cannot fully express my admiration for your willingness to share feedback with me even a few hours before your early morning conference presentation, and for finding time for me even during the holidays.

I would like to thank Dr. Florian Jaeck for giving me the opportunity to do research in his team at MHP. Many thanks to Dr. Christoph Höllig, who has been a great advisor and helped me a lot throughout the research, from selecting topics to sharing industry-specific insights and providing constant feedback! I really enjoyed my time at the company and would like to thank all 44 of my colleagues for their support in researching and completing the questionnaires. Without you, this work would not have been possible!

Finally, I want to express gratitude to my marvelous friends Denys the Open-Hearted, Oleg the Gummibärchen, and Alex the Rock-n-Work for spending a thrilling time together and for your support when it was especially needed.

And thank you Anastasia for our incredible adventure over the past years.

Abstract

According to recent studies, about 80% of data in companies are unstructured and hold enormous potential for industry insights [1]. Often, these data need to be classified to provide valuable insights into various domains such as process improvement, increased employee engagement, and cost reduction. Text classification is a classic supervised learning task [2], but to train a classifier using a modern neural network requires labeled data, which are scarce [3]. Manual creation of labeled data sets for specific tasks is often used as a solution. However, manual labeling is time-consuming and therefore leads to high labor costs [4, 5, 6, 7, 8].

In this Master's thesis, we investigate the use of a combined approach of *Natural Language Processing* (NLP) techniques to better understand safety issues in the automotive industry. Using an open-source data set from the *National Highway Traffic Safety Administration* (NHTSA) containing customer safety-related complaints over the past 25 years, we develop an algorithm to map a customer problem to predefined classes of vehicle components. Our goal is to develop a framework that can be flexibly adapted to multiple application areas to obtain multi-class labeled data for predefined classes that can be further used for supervised machine learning tasks.

We investigate the combination of expert knowledge and pre-trained word and sentence embedding models to obtain labeled data from the data set. The classification process is based on the similarity of a single document sentence to each of the target classes represented by a set of expert-defined descriptions, and is evaluated based on the cosine similarity of their vector representations. The results are presented as a structured labeled data set with a similarity score for each sentence-class pair. In addition, the challenges during the process are described and the usefulness of using further unsupervised learning methods to achieve the described goals is discussed.

To corroborate our findings, employees of the cooperating company helped us validate the results. We show that using our approach saves up to 79% of labor time compared to manual annotation while providing 3.5 to 20 times more training data. The Sentence-BERT-based embedding models perform the classification task well, achieving high F1-scores of up to 1.0. We also believe that the proposed approach has the potential to be used for unsupervised text classification.

Kurzfassung

Aktuellen Studien zufolge sind etwa 80% der Daten in Unternehmen unstrukturiert und bergen ein enormes Potenzial für Brancheneinblicke [1]. Häufig müssen diese Daten klassifiziert werden, um wertvolle Erkenntnisse in verschiedenen Bereichen wie Prozessverbesserung, Steigerung des Mitarbeiterengagements und Kostensenkung zu gewinnen. Die Textklassifizierung ist eine klassische Aufgabe des überwachten Lernens [2], aber um einen Klassifikator mit einem modernen neuronalen Netzwerk zu trainieren, sind gelabelte Daten erforderlich, die nur selten vorhanden sind [3]. Die manuelle Erstellung von gelabelten Datensätzen für bestimmte Aufgaben wird oft als Lösung verwendet. Das manuelle Labeling ist jedoch zeitaufwändig und führt daher zu hohen Arbeitskosten [4, 5, 6, 7, 8].

In dieser Masterarbeit untersuchen wir die Verwendung eines kombinierten Ansatzes von *Natural Language Processing* (NLP) Techniken, um Sicherheitsprobleme in der Automobilindustrie besser zu verstehen. Unter Verwendung eines Open-Source-Datensatzes der *National Highway Traffic Safety Administration* (NHTSA), der sicherheitsrelevante Kundenbeschwerden der letzten 25 Jahre enthält, entwickeln wir einen Algorithmus zur Zuordnung eines Kundenproblems zu vordefinierten Klassen von Fahrzeugkomponenten. Unser Ziel ist es, ein Rahmenwerk zu entwickeln, das flexibel an verschiedene Anwendungsbereiche angepasst werden kann, um mehrklassige gelabelte Daten für vordefinierte Klassen zu erhalten, die für überwachte maschinelle Lernaufgaben weiter verwendet werden können.

Wir untersuchen die Kombination von Expertenwissen und vortrainierten Wort- und Satzeinbettungsmodellen, um beschriftete Daten aus dem Datensatz zu erhalten. Der Klassifizierungsprozess basiert auf der Ähnlichkeit eines einzelnen Dokumentensatzes mit jeder der Zielklassen, die durch einen Satz von durch Experten definierten Beschreibungen repräsentiert werden, und wird auf der Grundlage der Kosinusähnlichkeit ihrer Vektordarstellungen bewertet. Die Ergebnisse werden in Form eines strukturierten, beschrifteten Datensatzes mit einer Ähnlichkeitsbewertung für jedes Satz-Klassen-Paar präsentiert. Darüber hinaus werden die Herausforderungen während des Prozesses beschrieben und die Sinnhaftigkeit der Verwendung weiterer unüberwachter Lernmethoden zur Erreichung der beschriebenen Ziele diskutiert.

Um unsere Ergebnisse zu untermauern, halfen uns Mitarbeiter des kooperierenden Unternehmens bei der Validierung der Ergebnisse. Wir zeigen, dass die Verwendung unseres Ansatzes im Vergleich zur manuellen Annotation bis zu 79% der Arbeitszeit einspart und gleichzeitig 3,5 bis 20 Mal mehr Trainingsdaten liefert. Die auf Sentence-BERT basierenden Einbettungsmodelle erfüllen die Klassifizierungsaufgabe gut und erreichen hohe F1-Werte von bis zu 1,0. Wir glauben auch, dass der vorgeschlagene Ansatz das Potenzial hat, für die unüberwachte Textklassifizierung verwendet zu werden.

Contents

1. Introduction

1.1. Background

This master's thesis was done in collaboration with a German Management- and IT-Consulting company with an international presence (hereafter referred to as "the company"). As a digitization expert, it provides consulting services worldwide to optimize and digitize customers' processes across the entire value chain with its Management Consulting, System Integration, Managed Services, and Digital Services & Solutions service areas. As an industry expert - especially for mobility and manufacturing - it offers its customers not only comprehensive IT expertise, but also in-depth management and process know-how. The company also transfers strategic innovations to other industries. Headquartered in Ludwigsburg, north of Stuttgart has more than 3200 employees advising over 300 clients from 19 locations - Ludwigsburg (3x), Berlin, Frankfurt am Main, Ingolstadt, Munich, Nuremberg, Dusseldorf, Wolfsburg (2x), England (Reading), the USA (Atlanta), China (Shanghai), Romania (Cluj-Napoca, Timișoara), the Czech Republic (Prague) and Austria (Zell am See). The company supports its customers on a national and international level both strategically and operationally. With its extensive international project experience, an established partner network, and international locations, it supports its customers in both national and international projects and globalization plans.

On the one hand, this mutually beneficial collaboration provided access to the company's employees, who have significant working experience in the automotive industry and therefore are considered *domain experts*. Their expertise is useful in evaluating the results of this thesis. On the other hand, the company is able to expand its experience in text mining and possibly develop a use case for one of its service lines.

As the amount of text and unstructured data is rapidly increasing, it accounts for 80% of internal data [1], but only 18% of organizations are able to capitalize on these data [9]. Every day consumers give their feedback and describe the positive and negative sides of their vehicles. Among these data, valuable insights could be found to help automotive manufacturers achieve advantages in various areas such as assessing the strong and weak sides of their vehicles, staying informed about customer preferences and industry trends, and potentially improving the processes across the entire value chain. The improvements in this area could benefit both the company and its customers.

The growing potential of machine learning applications is remarkable in the automotive sector. Due to recent advances in the field of natural language processing (NLP), organizations are able to extract valuable information from text. The company noticed the growing interest of customers in text mining tasks and started investigating use cases that could be built upon. One of the attractive use cases is classifying automotive consumer complaints with respect to vehicle components. This could lead, for example, to faster response to safety-related

issues, better personalization of customer service, or detection of brand-related trends. This implementation could make the processing of customer complaints more efficient.

1.1.1. Problem Description

Text classification is a typical supervised machine learning task where the model goes through a training, testing, and validation phase. To train a classifier, a labeled data set is required. However, many tasks in the industry are customer-specific, and labeled data is scarce. We can consider manually creating an annotated data set to train the classifier on these data, but in this case, we need to consider the following issues:

- First, if we want to conduct a supervised classification method, we need to annotate our data manually, which is time-consuming because supervised text classification methods rely on a large amount of manually annotated data [6, 10], and therefore incur high labor costs.
- Second, in the case of multi-class classification, a manually labeled data set cannot be easily adapted to changes in the problem description, e.g., when additional target classes are added or the hierarchy between classes is changed. As Romera-Paredes and Torr [11] showed, many approaches cannot keep up with new classes when they appear after the training phase. In such cases, much additional effort may be required to adapt the classifier, up to and including a complete relabeling of the data.
- Third, we may not want to derive all available topics from the data set, but are only interested in a few relevant classes. Using a supervised classification method would require either annotation of documents that are not of interest or additional data cleaning steps to remove irrelevant documents from the data set. In either case, significant labor costs could be incurred.
- Fourth, in industrial practice, it is also often necessary to classify not only in terms of topics, but also in terms of other features such as sentiment. For example, we are interested in classifying documents that relate to vehicle components **and** have a negative sentiment in the form of a problem description or a complaint.
- Fifth, the length of documents can vary considerably, from a few words to several dozen sentences reflecting either one, several, or no target topic. In this case, we also want to capture plural target topics hidden behind single sentences.

Costly data preparation leads to business incentives to explore the area of multi-class text classification. The question arises whether training data for multi-class text classification can be obtained by combining the knowledge of domain experts, who not only have experience in their field but also a good understanding of the underlying data set, with pre-trained, transformer-based language models. We also want to investigate whether the resulting approach can help overcome classification challenges, such as adapting flexibly to changing requirements.

1.1.2. Project Goals

The current approach to solving the described problem requires that each document is labeled individually to create a vocabulary-specific training data set for further classification. This process must be done manually and is therefore time-consuming and not flexible. For example, for a data set of 100,000 documents, all documents must be manually read and labeled. In addition, this approach is not adjustable to changing requirements because the labels remain static and cannot be quickly customized. For example, if a higher level of granularity is required, the entire data set must be labeled from the beginning, which in turn causes high personnel costs and time delays.

We propose a combined approach involving domain expertise and various NLP techniques to solve the training data preparation problem. The underlying problem is divided into simple steps performed by domain experts aiming to solve the problems described above.

The process involves the creation of a pipeline with several steps such as data preprocessing, creation of *class descriptions*, extraction of *context windows* [12] and their evaluation to become *context rules*, computation of the *class vectors* with a subsequent document classification, and result validation. Our goal is to provide a framework for creating training data from unlabeled text corpora that can be generalized to classification tasks in other industries. A supervised text classifier is supposed to be trained using the output of our approach. Thus, it represents the first stage of a semi-supervised classification approach.

The results are presented as a structured data set consisting of a unique complaint identifier, a full text of the document, a classified sentence, a corresponding class, and a *similarity score*. The similarity score serves as an indicator of belonging to the target class and as a threshold for separating the classified data from the less relevant data. The assessment of the results is conducted in both intrinsic and extrinsic ways with the involvement of industry experts. The results are quantified using standard evaluation metrics such as precision, recall, and F1-score. The agreement rate between the validators is also reported.

1.2. Research Questions

The following research questions are investigated in this thesis:

- RQ1: What are the challenges faced when trying to create structured data sets from unstructured documents?
- RQ2: Which NLP methods can be combined with the domain expertise to facilitate the extrapolation from context rules to training data?
- RQ3: How do these novel methods compare to current methods of unsupervised learning in the context of automotive customer data?

The first research question aims to evaluate the viability of the proposed approach and identify limitations and potentials for future research. The second research question examines different NLP approaches to classification, particularly those based on the creation of rules

for target classes. The different methods are then combined to meet the specific goals of our task. The third research question describes the advantages and disadvantages of common unsupervised learning techniques applied to our task and explores their potential for facilitating classification activities.

1.3. Delimitations

For doing data and NLP-related experiments, Python is commonly regarded as the language of choice among data scientists and academics. In the last several years, a large number of NLP packages with Python-enabled bindings have appeared. Therefore, for all experiments in this thesis, we utilize the Python programming language and its accompanying libraries. For our goals, we mainly rely on well-known data analytics Python libraries such as pandas [13], spaCy, sci-kit learn [14], and Hugging Face [15].

We obtain our results using an open-source data set from the automotive industry that reflects a very specific task. We do not test our approach on common data sets for the classification tasks, but we assume that our approach can be generalized for other purposes since the classes for each task are described manually.

The process of identifying and describing the target classes is emulated using our own competence and comprehensive online resources.

The obtained results are not supposed to be expected as final results for a multi-class classification. The proposed method solely aims to decrease time spent for manual data annotation for specific topics where industry expertise is available and target topics are predictable. Hence, a multi-class text classifier is supposed to be consequently trained using supervised learning approaches based upon our labeled results.

2. Theoretical Foundation

This chapter provides information about the essential concepts for this thesis, reviews the historical development of natural language processing, and introduces the techniques required for text processing. In this context, the methods by which texts can be represented in vector form are also presented. At the end of this chapter, similarity computation between vectors is discussed and metrics for obtaining quantifiable classification results are given.

2.1. Machine Learning

Machine learning is a subcategory of *Artificial Intelligence* (AI) and encompasses applications in search algorithms, text processing, predictions, planning and scheduling, computer vision, speech recognition, and many other areas [16]. Instead of following a static algorithm described by a human, the system learns patterns from data and improves the results by working through the training data set multiple times. Machine Learning is divided into three areas: supervised learning, unsupervised learning, and reinforcement learning. For the sake of this thesis, the first and the second area are briefly described below.

2.2. Supervised Machine Learning

Supervised machine learning leverages labeled input for the training lifecycle [16]. Usually, the labels are provided by a data scientist in the preparation phase before feeding the data into the system. After the training phase is completed, the model can be used for further predictions. Supervised machine learning is used for classification and regression problems [16]. Through classification, the model maps input data to a predefined number of categories, for example, classifying handwritten numbers in classes from zero to nine. Regression tasks, on the other hand, are used to map the input data to a real value, for example, predicting the prices of houses based on their characteristics.

2.3. Unsupervised Machine Learning

Unsupervised machine learning learns from unlabeled training data to combine data into categories [16]. This technique is often used to better understand the data set by means of clustering data into meaningful groups or identifying the similarity between data pairs. Though the model does not require human supervision in form of labels, the explainability of results could be limited. An example of unsupervised machine learning is grouping customers based on their buying preferences.

2.4. Brief History of Natural Language Processing

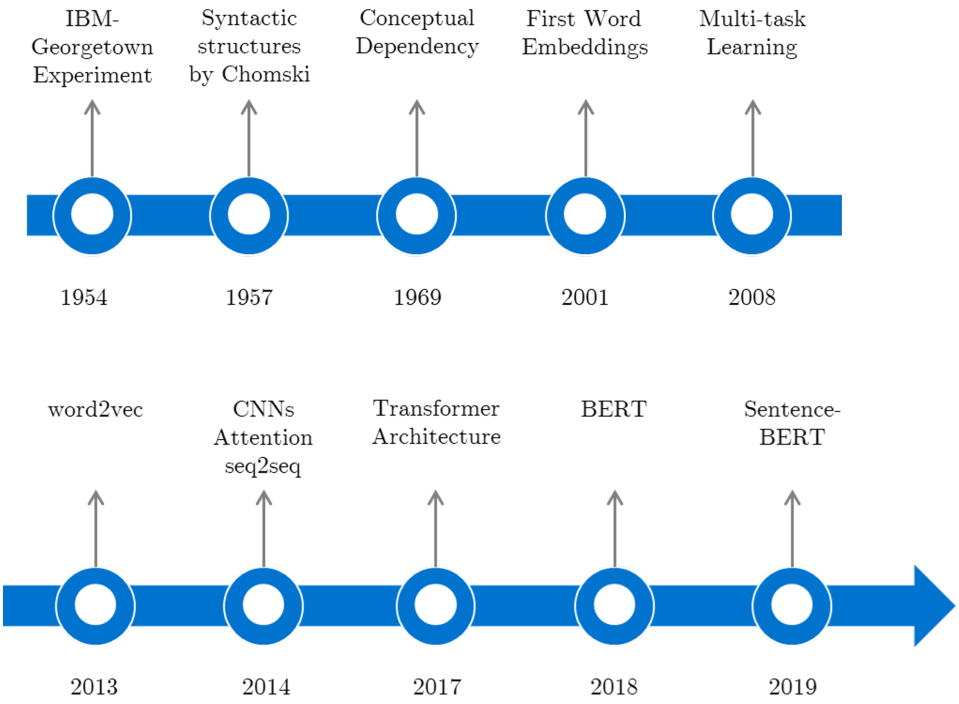


Figure 2.1.: A brief history of NLP.

NLP is an exciting field of computer science that focuses on understanding human communication. It includes methods for assisting machines in comprehending, interpreting, and producing human language. *Natural Language Understanding* (NLU) and *Natural Language Generation* (NLG) terms are often used to distinguish between two applications. One should not undervalue the depth and complexity of human language. At the same time, there is a rising demand for algorithms that can understand language, and natural language processing fulfills this requirement. A linguistics-based approach is used in traditional NLP techniques, which start with the fundamental semantic and syntactic components of a language, such as *Part of Speech* (POS). Modern methods can eliminate intermediary components and even develop their own hierarchical representations for generalized tasks.

In this section, we focus on some significant events that have influenced Natural Language Processing as we understand it now.

Although there were appealing experiments in the 1940s, the IBM-Georgetown experiment of 1954, which demonstrated the automatic translation of about 60 phrases from Russian into English, can be seen as the first significant achievement in the field [17]. Despite the limitation of software and hardware resources, some of the syntactic, semantic, and linguistic difficulties were identified, and an effort to solve them was made. The next remarkable event was the introduction in 1957 of the book Syntactic Structures by Noam Chomsky where he highlighted the significance of sentence structure for language comprehension [18].

Though the NLP researchers had access to significant funding to support their efforts, the United States *National Research Council* (NRC) was established in 1964 to assess the development of NLP research. Emphasized in the report challenges associated with machine translation highlighting the implementation cost had a serious impact on funding decisions that almost brought the NLP research to an end [19].

The study of world knowledge had a phase from the 1960s to the 1970s during which semantics took precedence over syntactical structures. One important discovery in this period was Schank's conceptual dependence, which described the language in terms of semantic primes without syntactical processing [20].

The grammatico-logical phase, in which linguists established various grammar structures and began associating meaning in phrases concerning users' intentions, commenced in the early 1980s. This period was remarkable for the creation of popular tools for information retrieval, parsing, and machine translation such as METEO [21].

Most NLP-based systems adopted numerous novel concepts for data collection throughout the 1990s, such as comprehending words based on their occurrence and co-occurrence utilizing probabilistic-based techniques or employing corpora for linguistic processing [22]. For many NLP applications, supervised and unsupervised techniques such as n-grams and a *bag-of-words* (BoW) representation with machine learning algorithms like multinomial logistic regression, support vector machines, Bayesian networks, or expectation-maximization were prevalent [23, 22].

In the early 2000s, a paper by Bengio et al. [24] introduced the "dense vector representation" as an alternative to common approaches of "one-hot vector" or bag-of-words. This period was also marked by significant research by Collobert and Weston who in addition to bringing attention to concepts like pre-trained word embeddings and convolutional neural networks for text also shared the lookup table or embedding matrix for multitask learning [25]. By eliminating the hidden layer and using approximations for a softmax function, Mikolov et al. [26, 27] increased the effectiveness of training the word embeddings given by Bengio et al. This resulted in "word2vec", an effective large-scale implementation of the embeddings. The word2vec architecture is represented through two implementations: *continuous bag-of-words* (CBOW), which predicts the center word based on the words around it, and skip-gram, which does the reverse and predicts the words around the middle word. This method and the large text corpora used for the training allowed for capturing multiple semantics and relationships. The model proposed by Sutskever et al. [28] called the sequence-to-sequence model is a general neural framework composed of an encoder that processes items in the input sequence one by one to compile the captured information into a vector and a decoder that predicts the output item by item based on both input sequence and current output states. Increasing discussions around deep learning led to the expanded implementation of neural networks for text processing. Previously used for the processing of visual content, *Convolutional Neural Networks* (CNNs) became common for various NLP-related challenges such as Sentence Classification [29], Text Classification [30], Sentiment Analysis [31], Text Summarization [32], Question Answering [33], and Machine Translation [34].

A further innovation was suggested by Bahdanau et al. [35] featuring the attention mech-

anism that allows the model to conduct an automatic search for the relevant part of the input text to predict the output word avoiding the hard text segmentation. Combined with the usage of transformers, this method became the cutting-edge architecture for NLP tasks, surpassing other neural models like Convolutional and *Recurrent Neural Networks* (RNNs) for the tasks in both NLU and NLG [36]. An introduction of pre-trained models such as *Bidirectional Encoder Representations from Transformers* (BERT) allowed models in the first step to use generic text corpora for training and subsequently being adapted for specific downstream tasks [37]. As a response to the huge computational overhead for identifying similarity pairs in text using the BERT model, the *Sentence-BERT* (SBERT) modification was introduced, which dramatically reduced the time required for finding similarity pairs of text embeddings [38].

2.5. Basics of Text Processing

Natural language is intrinsically ambiguous, especially when it is in a written form. Out of 170,000 words in a dictionary, only roughly 10000 are used on a regular basis [39]. When processing a natural language, it is often useful to group language features into multiple categories that could be identified on the basis of linguistic features. For example, such features can have morphological, lexical, semantic, or syntactic natures. In this thesis, primarily morphological and lexical analysis come into consideration.

2.5.1. Morphological analysis

Morphology describes the shape of a word and its internal structure. For morphological analysis, we rely on the smallest parts of the word holding a unique meaning which are called *morphemes*. They could describe the meaning of the word, its origin, or its grammatical role. Some tasks that are simple for humans such as recognizing the morphological meaning of the words *"walk, walking, walked"* are complex for a computer. A large number of various combinations substantially increases the comprehension complexity for computational calculations. To reduce this complexity, morphological analysis is often performed. Two common approaches are *stemming* and *lemmatization*.

Stemming is used to convert words to their base morphemes or to their root. For example, Kamath et al. [16] used the following example to represent stemming:

$$\text{works} \rightarrow \text{work}$$
$$\text{worked} \rightarrow \text{work}$$
$$\text{workers} \rightarrow \text{work}$$

One advantage of stemming is that it is generally robust to spelling errors since the correct morpheme can be recognized anyway. On the other hand, it increases ambiguity. For example, although all three words have the same root, the first word describes items and the third word describes people.

Lemmatization is able to mitigate the disadvantages of stemming by reducing the words to their dictionary form or their lemma. With reference to the example from above given by Kamath et al. [16], the following lemmas are obtained:

works → works
worked → work
workers → worker

The advantages of lemmatization are the preservation of the context and the comparatively high robustness against spelling errors. The disadvantage is that this algorithm is usually slower than stemming.

2.5.2. Lexical Representations

Lexical means the segmentation of the text into meaningful components such as words. Although words may contain multiple morphemes, they are often considered elementary particles of natural language. It is often good practice to process text on the basis of individual words. The process of breaking down a text into meaningful words or units is called textittokenization [16]. A simple process of separating text by a space sometimes produces meaningful results. Consider the following examples from Kamath et al. [16]:

The rain in Spain falls mainly on the plain.
|The|, |rain|, |in|, |Spain|, |falls|, |mainly|, |on|, |the|, |plain|, |.|

But sometimes this method fails [16]:

Don't assume we're going to New York.
|Don|, |'t|, |assume|, |we|, |'|, |re|, |going|, |to|, |New|, |York|, |.|

The problem with this division is that sometimes we want to consider multiple words like "New York" as a single token. Also, problems with tokenization can be caused by built-in punctuation. Consider the following example [16]:

Dr. Graham poured 0.5ml into the beaker
|Dr.|, |Graham poured 0.|, |5ml into the beaker.|

To overcome punctuation-related issues, we remove special characters from the documents. The procedure is described in chapter 5.

One of the common techniques to improve text processing is removing *stop words*. Stop words are the most common words that do not add meaning to the context [16]. In English, examples of such words are "a", "the", "is", "are". The list of words to be excluded is commonly called a *stop word list*.

2.6. Vector Representation Techniques for Text Data

Computational linguistics employs a variety of document representation techniques when dealing with a corpus of documents. Some are character-based, token-based, or multi-token-based. Both sparse and dense representations of documents are possible. When working with

multiple corpora, we need to decide on a suitable representation for a given task. Some text representation techniques are discussed below.

In 1954, Harris proposed a *distributional hypothesis* arguing that the words occurring in the same contexts tend to have similar meanings [40]. This idea became influential for a number of studies and plays an important role in current research [26, 41, 42]. The hypothesis could also be explained the other way around, by assuming the distribution of morphemes with different meanings to be different. For example, *table* and *sun* are two words with unrelated meanings and they are not frequently used together in a sentence. On the other hand, the words table and chair are more likely to be mentioned together in a sentence, suggesting a certain similarity in the meanings of these words.

A straightforward approach to represent text as a vector is *one-hot encoding*. In this approach, each word and symbol is represented as a vector consisting only of one and zero. Each word in a sentence can be represented by a one-hot vector, each of which is unique. In this way, the word can be uniquely recognized by its one-hot vector and vice versa, which means that no two words have the same representation of a one-hot vector. Consider the following example:

The dog sat on the rug
The: [1 0 0 0 0 0]
dog: [0 1 0 0 0 0]
sat: [0 0 1 0 0 0]
on: [0 0 0 1 0 0]
the: [0 0 0 0 1 0]
rug: [0 0 0 0 0 1]

It is noteworthy that the words "The" and "the" are encoded as distinct words. Thus, words and symbols can be encoded as unique vectors containing one and zero as constituent elements. To represent a text document, it could be thought of as an array of vectors or a matrix. If multiple documents need to be encoded, they can be represented as a three-dimensional array. The resulting vectors are sparse because all elements except one are zeros.

An early mention of a one-hot encoding application was given by Harris [40] under the title bag-of-words, in which punctuation and grammar are neglected in favor of a multiplicity [43]. The process of converting text documents into a *document-term matrix* representing documents in rows and unique tokens in columns is commonly known as *count vectorization* [16]. Practical applications of the BoW model are mainly in feature generation. The most commonly computed measure is term frequency, i.e., the frequency with which a term occurs in the text. However, the weights of the model are primarily related to the number of term occurrences, and although stop words are removed before count vectorization, it is often found that the words with fewer occurrences are more informative [16].

These considerations lead us to an improved model called *term frequency-inverse document frequency* (TF-IDF) that introduced the way to give greater weights to underrepresented terms w by multiplying the *term frequency* (TF) by the *inverse document frequency* (IDF) of each

token [44]:

$$w = tf \times idf \tag{2.1}$$

In this equation, the term frequency is calculated as the logarithmically scaled count of a term with respect to the total number of documents, and inverse document frequency as the logarithmically scaled ratio of the total number of documents N with respect to the number of documents with the considered term n_i. The above equation can thus be also represented in the following way:

$$w = (1 + log(TF_t) \times log(\frac{N}{n_i})) \tag{2.2}$$

The tf factor increases the value of w proportionally depending on the number of times it appears in a given document, while the idf factor reduces the value of w depending on the number of occurrences across all documents.

A more advanced representation of words as vectors was suggested by Salton by using the *Vector Space Model* (VSM) to utilize an n-dimensional vector space in which each point can be expressed as an n-dimensional vector [45]. VSM provides a framework in which word vectors can be represented and compared given that semantically similar words are supposed to have similar representations. It is common to call word vectors *embeddings* [16] that are often used throughout this thesis.

2.6.1. Word2vec Model

Since most of the work in this thesis is based on operations with text embeddings, we want to briefly explain how word embeddings are obtained in practice.

The architecture proposed by Mikolov et al. [26] in 2013 uses large data sets to obtain continuous vector representations of words. The authors avoided the commonly used dense matrix multiplications, making their model fast and efficient at learning vector representations from much larger text corpora than previous technologies. They named their architecture *word2vec* and used CBOW and *skip-gram model* as main components. The authors stated that relational and linguistic similarities between words can be found using vector arithmetic. For example, they showed that the word "queen" can be found by searching for the closest vector based on cosine similarity after performing arithmetic operations on the words "king", "man", and "woman":

$$\text{v(queen)} \approx \text{v(king)} - \text{v(man)} + \text{v(woman)}$$

This concept can be also applied to other objects. For instance, using the embedding vectors for the words "Germany", "Berlin" and "Austria" and calculating Berlin – Germany + Austria results in a vector that is very close to the vector representing the word "Vienna". Analogically, this leads to similarities between pairs "Man - Woman" and "Boy - Girl". Visualizations of these word embedding pairs are presented in Figure 2.2.

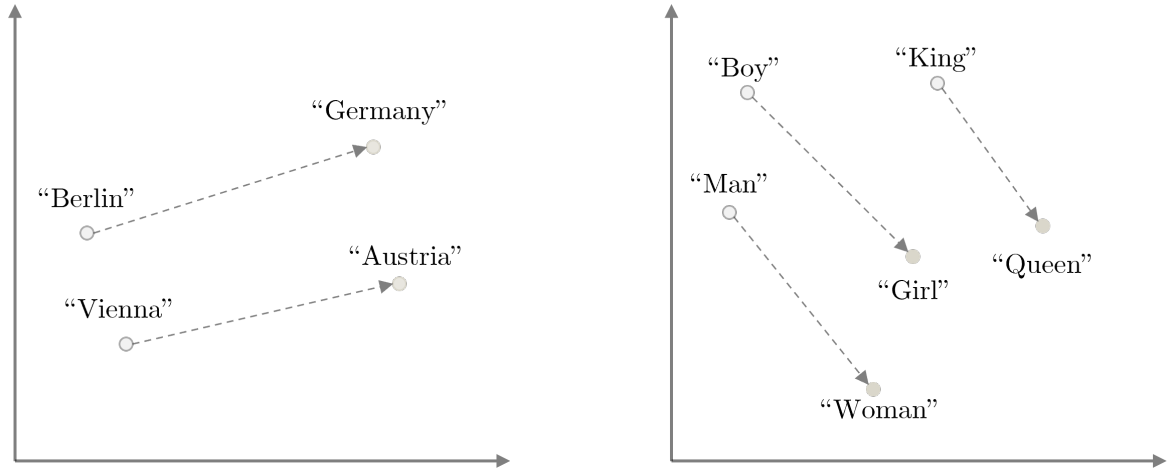


Figure 2.2.: Representation of dependencies between word embedding pairs in two dimensions

The terms CBOW and skip-gram model introduced above are briefly explained below. The CBOW method is used to predict the central word depending on the surrounding words and a certain context window size. The Skip-Gram model performs the reverse process, i.e., in this case, the neighboring words are to be predicted based on the word in the center. Both techniques are shown using the context window length of three. The target words are colored red and the context words are colored blue:

CBOW

Natural language processing is fun
Natural language processing is fun
Natural language processing is fun

Skip-gram

Natural language processing is fun
Natural language processing is fun
Natural language processing is fun

In our thesis, we conducted experiments using *tok2vec* and embeddings by spaCy's *en_core_web_lg* model that relies on word2vec representations. The procedure is described in chapter 6.

2.7. Similarity Calculation for Embeddings

The similarity between two words or texts can be measured quantitatively and the words or texts representing similar contexts are supposed to have higher *similarity scores*. We introduce a technique to compute the similarity between two vectors in this section.

Cosine similarity is a measure of similarity between two vectors in a multidimensional space and it is defined as the cosine angle between these vectors. It is calculated as the dot product of the vectors divided by the product of their lengths. This measure is not determined by

the dimensionality of the vectors but only by the angle between them. The cosine similarity varies in the interval $[-1, 1]$.

The dot product between two N dimensional vectors x and y is calculated in the following way:

$$a \cdot b = \sum_{i=1}^{N} a_i b_i \tag{2.3}$$

The length of a vector x is defined as follows:

$$|a| = \sqrt{\sum_{i=i}^{N} x_i^2} \tag{2.4}$$

Han et al. [46] provide the following formula to compute the cosine similarity:

$$similarity(x, y) = \cos(\theta) = \frac{x \cdot y}{|x||y|} \tag{2.5}$$

The result of this computation is also called the similarity score throughout this thesis.

2.8. Precision Metrics

Classification tasks can be evaluated by using multiple metrics. Many of them rely on the terms *true positive*, *false positive*, *true negative*, and *false negative*. As "true" or "false" is described whether the classification task was performed correctly. "Positive" and "negative", in contrast, indicate whether a sample belongs to a target class or not. Further, we introduce commonly used classification metrics, which are precision, recall, and F1-score.

Precision represents correctly predicted samples with respect to all samples predicted as positive [47]:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.6}$$

Recall stands for a fraction of correctly predicted samples to all relevant samples [47]:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.7}$$

F1-score is another common classification metric. It is computed as a harmonic mean of precision and recall [47]:

$$F1 - score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

These metrics are used for the evaluation of the classification algorithm in chapter 6.

3. Related Work

In this chapter, we introduce the concepts on which the implementation of this thesis is built. We start with an overview of the NLP frameworks that serve as inspiration for our approach and introduce the idea of pre-trained models. We then describe the transformer-based architecture and introduce the models used in this work. Finally, two unsupervised learning approaches commonly used to facilitate text classification tasks are presented.

3.1. Dataless Classification

A significant part of the work related to our thesis stems from the *zero-shot* setting, which means that we try to classify documents without any labels [10]. The mostly related framework was introduced by Chang et al. [48] under the term *dataless classification*. It uses *Explicit Semantic Analysis* (ESA) [49] for a representation and further comparison of the documents and classes in the same latent space. The idea behind this framework is to mimic how easily people can classify documents based on their understanding of the underlying classes [48]. The framework has a broad application area allowing the use of any approach that aims to classify data based only on the label description. A slight difference with our approach is that we are not attempting to assign the documents to an arbitrary topic of no interest, but we want to classify the documents into certain pre-defined classes. However, we refer to dataless classification as it has comparable characteristics to our approach, such as classifying documents into multiple classes in the context of zero-shot learning. It can be further divided into two sub-categories:

1. Unsupervised vs. semi-supervised approaches.
2. Training a tailored model vs using pre-trained models.

3.1.1. Unsupervised Approaches

The document classification procedure based only on unsupervised approaches aims to classify unlabeled documents by computing a similarity between the documents and target classes.

Haj-Yahia et al. [3] used a combination of *keyword enrichment* (KE) and unsupervised classification technique based on cosine similarity after performing *Latent Semantic Analysis* (LSA) [50]. Ha-Thuc and Renders [5] utilized ontological knowledge to find pseudo-relevant documents from the Web without labeled data, and employed topic modeling with ontological guidance to obtain classified documents. Meng et al. [8] used the labels of each class only for

training classification models on unlabeled data. Song and Roth [51] argued that teaching the model to understand labels can be used without additional supervision to accurately categorize the documents.

We expect that our approach can be used completely unsupervised in the future after a successful investigation of the challenges and trade-offs described in chapter 6. However, we currently propose to use it for labeling a subset of documents in combination with training a supervised text classifier.

3.1.2. Semi-Supervised Approaches

Semi-Supervised approaches are performed using a two-step process. First, an unsupervised learning approach is used to extract meaningful labels for the underlying data. Second, a text classifier is trained upon the derived annotated data. Liu et al. [4] use the naïve Bayesian classification method [52] and *Expectation Maximization* (EM) [53] algorithms to build a classifier based on a subset of unlabeled documents and user-provided keywords describing target classes. They first identify documents that could match the given keywords and then use the data as input for a text classifier. Analogically, Johnson and Zhang [54] propose a CNN model that first utilizes unlabeled data to learn the embeddings of small text regions and then labels the documents. Our approach falls into the first step of this category, as it identifies labels for documents based solely on the vector similarity between document sentences and the target class. Then, the sentences with high similarity to the target classes are used to train a multi-class text classifier.

3.1.3. Training a Tailored Model

A model can be trained from scratch by relying on available data sources, either derived from a comprehensive, freely available world knowledge such as Wikipedia with 2.5B words and BookCorpus with 800M words used for the training of BERT [37]. However, this approach requires significant investment in time and resources. We omit the training process of a tailored model in favor of a pre-trained model.

3.1.4. Using Pre-Trained Models

Another option to obtain text embedding was to leverage a state-of-the-art pre-trained model for our work. Pre-trained language models are commonly used for text classification and achieve state-of-the-art performance on most data sets [10]. Meng et al. [8] used pre-trained neural language models for understanding the target classes and further for document classification. Chai et al. [55] employed pre-trained models to propose a framework for text classification, which is expected to connect each category label with a category description. The mechanism behind the pre-trained models utilized for our approach is described below.

3.2. Pre-trained Models for Natural Language Processing

Recent studies on *pre-trained models*(PTMs) have shown that learning a language model from large unstructured corpora is possible and more beneficial for downstream NLP tasks than training a new model from scratch. This method is used to pre-train a single model to work with multiple different downstream tasks, a method called *transfer learning*. Rising computational power and the introduction of deep models such as Transformer [36] have led to remarkable advances in the architecture of PTMs [56].

The number of model parameters has significantly expanded as a result of the development of deep learning. Proper training of model parameters with overfitting prevention requires a data set of significant size. However, the high annotation cost of creating large labeled data sets [4], especially with high semantic and syntactic similarity, makes this a major challenge for NLP activities [56]. For this reason, using large unlabeled corpora is a much more straightforward task. In this case, word representations can first be learned from unlabeled text data and then used for other tasks. The advantages of using word representations extracted from large unlabeled text corpora are supported by recent studies [10].

Erhah et al. [57] summarized the advantages of pre-training as follows:

1. Due to the use of huge text corpora, universal language representations can be learned in the pre-training process.

2. A model can be easier initialized increasing generalization performance and making the convergence on the downstream task faster.

3. Pre-training can avoid overfitting with small amounts of data.

All modern PTMs can be divided into two generations. The first-generation PTMs are able to learn good word embedding capitalizing on shallow architectures. For example, Skip-Gram [26] and *Global Vectors for Word Representation* (GloVe) [58] have a simple implementation, although they can capture high-quality semantic similarities among words. However, these models are context-free and are not able to capture more advanced concepts such as polysemous disambiguation, syntactic structures, semantic roles, and anaphora [56].

The first successful second-generation pre-trained model for NLP was proposed by Dai and Le [59]. They used *Long Short-Term Memory* (LSTM) [60] for the instantiation of the *Language Model* (LM) and observed improvements through pre-training in multiple classification tasks. The main feature of the second-generation PTMs is their focus on the extraction of contextual word embeddings. Among successful implementations are ELMo [61], OpenAI GPT [62], and BERT [37].

The difference between non-contextual and contextual embeddings is whether or not the word embedding dynamically adjusts to the context where it appears, and is illustrated in Figure 3.1.

Since the use of non-contextual or contextual embeddings has been an essential part of this work, the difference between them is further explained in more detail.

To obtain non-contextual embeddings, first, separate words have to be mapped on the distributed vector space. This could be formally explained in the following way: we map

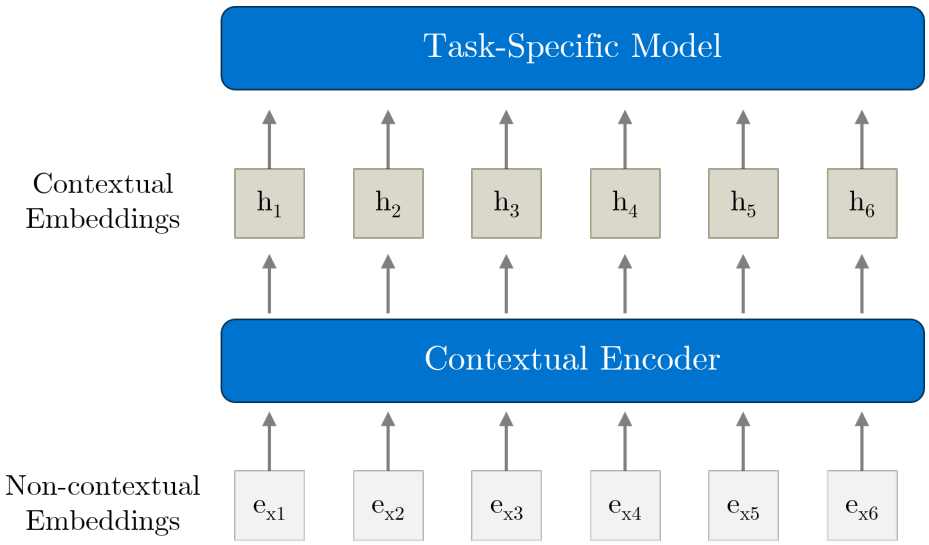


Figure 3.1.: Generic Neural Architecture for NLP

each word x to a vector $e_x \in \mathbb{R}_e^D$ with a lookup table $\mathbf{E} \in \mathbb{R}^{D_e \times |\mathcal{V}|}$, where D_e relates to the dimension of token embeddings and $\mathcal{V}$ stands for the vocabulary [56]. In the next step, the training of these embedding along with other parameters takes place. The main disadvantage of such models is that they are static and therefore cannot represent words with multiple meanings [56].

The problem with the dynamic representation of word embeddings can be tackled by distinguishing word semantics in different contexts. In a text containing tokens $x_1, x_2, \cdots, x_T$ where $x_T \in \mathcal{V}$, the contextual embedding of x_T is dependent on the entire text. Qiu et al. [56] represented this dependency in the following equation:

$$[h_1, h_2, \cdots, h_T] = f_{enc}(x_1, x_2, \cdots, x_T), \tag{3.1}$$

where $f_{enc}(\cdot)$ is a *neural encoder* and h_t is called *contextual embedding* as it contains additional information from the context. Neural encoders are commonly divided into two models: sequence models and non-sequence models. Sequence models are represented through models with short memory such as LSTM [60]. Non-sequence models include Transformer-based models. In practice, sequence models are easier to train and good results can be achieved on various NLP tasks. In contrast, the Transformer requires a larger corpus and more time to be trained but can model a dependency between every two words in a sequence of words making it a state-of-the-art architecture for the pre-trained models.

3.3. Transformers

Before the introduction of Transformer in 2017 by Vaswani et al. [36], sequence processing was primarily conducted employing recurrent neural networks, gated recurrent neural

networks [63], and LSTM [60] architectures. Training an RNN-based model is slow and it can not deal well with a vanishing gradient problem that occurs when sequences are long. Its successor, LSTM [60] can reduce the negative effects of the backpropagation problem, but the model training requires even more time because of higher model complexity. In addition, each of the models processes text sequences token by token not allowing to make use of modern GPUs to process input in parallel. The authors of the Transformer architecture suggest an architecture that allows processing data in parallel and relies on the mechanism called *self-attention* [36].

3.3.1. Architecture

The Transformer architecture comprises two components, which are the encoder (shown in Figure 3.2) and the decoder. Both of them are composed of components that can be stacked on top of each other multiple times, which is described by N_x in the figure. Most of these components are multi-head attention and feed-forward layers. The role of the encoder is to extract features from the input sentence, and the decoder produces the output sequence of symbols, for example in the case of a translation downstream task. In this thesis, we rely on the encoder part of the transformer since we are only interested in extracting features of the input sequence.

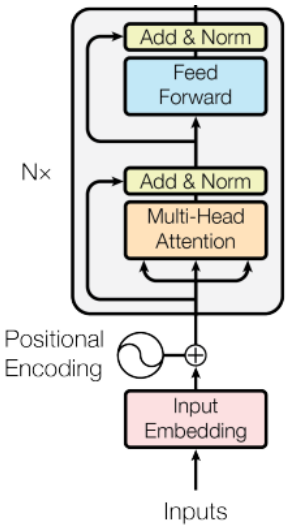


Figure 3.2.: The representation of the encoder part of the Transformer architecture (retrieved from [36])

3.3.2. Attention

In 2014, Bahdanau et al. [35] proposed an *attention* model by giving relative importance to each word in a sequence in contrast to taking into consideration all words with the same importance. This construct was further developed by Vaswani et al. [36] and described under

the term self-attention, along with the Transformer architecture, in the acclaimed paper *Attention Is All You Need*. The difference between attention and self-attention is where they are applied. Attention connects both input and output and allows one to consider input when the output is generated, while self-attention allows input sequences to interact with each other. Further mentions refer to the concept of self-attention, as it is more relevant for this thesis since only the input part is considered.

The self-attention mechanism allows the model to focus on the meaningful parts of the input sequence incorporating contextual information based on the word position in a sequence. For instance, if a model without self-attention gets the sentence "The fox did not jump over the lazy dog, because it was scared", it could struggle with referencing the word "it" to the word "fox" and not "dog". Self-attention could be useful in this case by putting more weight on "the fox" when encoding the word "it". The self-attention mechanism could be briefly described as giving weights to the input elements, and considering them when the model performs its task [36]. It allows one to deal with long input sequences without forgetting information from the beginning, and process these sequences in parallel.

To initialize the attention mechanism, query (Q), key (K), and value (V) matrices are obtained from each position in a sequence. For this purpose, a matrix of input embeddings (X) is multiplied with three previously trained weight matrices W_q, W_k, W_v [36]:

$$Q = XW_q \tag{3.2}$$

$$K = XW_k \tag{3.3}$$

$$V = XW_v \tag{3.4}$$

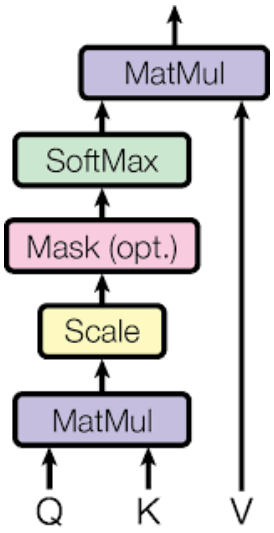


Figure 3.3.: Scaled Dot-Product Attention (retrieved from [36])

The dot product of queries and keys with dimensionality d_k is computed and the result is

divided by $\sqrt{d_k}$ to improve the stability of the model during training, followed by scaling through a softmax layer to an interval of $[0, 1]$, and obtaining the weights on the values [36]. The authors called the chosen constellation *Scaled Dot-Product Attention*, the structure of which is depicted in Figure 3.3 and the equation for calculating the attention is given below [36]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.5}$$

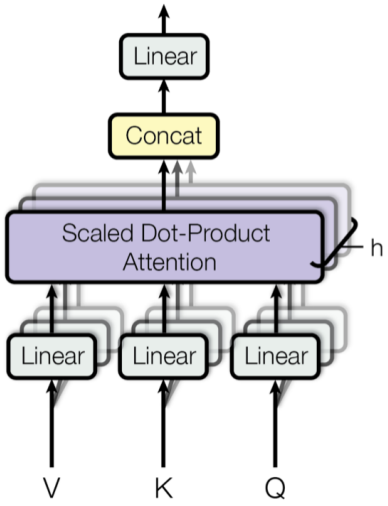


Figure 3.4.: Multi-head-attention (retrieved from [36])

An extension of the attention mechanism is called *multi-head attention* and its representation is given in Figure 3.4 [36]. It improves the performance of the attention layer in two ways:

1. The ability to focus on multiple positions is expanded. For instance, when encoding the word "it" from the sentence "The fox did not jump over the lazy dog, because it was scared", more attention could be paid not only to the word "fox" but also to the word "scared".

2. Multiple sets of query, key, and value matrices are considered expanding the opportunities to represent the input sequence [36].

3.4. Models

This section introduces two Transformer-based models, which are BERT and its modification Sentence-BERT, which was created through the efforts of the team behind the original BERT model. Two variations of the latter model are applied to this thesis. The models used are open-source and accessible through the *Huggingface* Transformers library.

3.4.1. BERT

In 2018, Devlin et al. [37] presented BERT, which outperformed previous models in several Natural Language Processing tasks. The innovative component of BERT was the detection of context from both the left and right directions of the target word, making the architecture bidirectional. In contrast, previous models were able to consider the context in the left-to-right direction or vice versa, limiting the understanding of context because words often depend on the two surrounding sides of the sentence [37].

Bidirectional training became possible due to leveraging the technique called *Masked Language Modelling* (MLM). The authors masked 15% of the input words with special tokens and had the model predict the masked words based on the context from both sides, which allowed for pre-training of a bidirectional transformer [37]. To capture the semantic relationships between multiple sentences, the pre-training was extended by the technique called *Next Sentence Prediction* (NSP). The idea is to train a model that correctly recognizes which sentence is a predecessor and which is a successor from two sentences.

For BERT pre-training, large corpora from the BooksCorpus and English Wikipedia containing 800 and 2500 million words respectively are used [37]. A trained WordPiece model [64] was used to create embeddings for all words. This model breaks long words into chunks and is therefore able to create embeddings of previously unseen words. In the BERT architecture, token, position, and segment embeddings are added to each token. Based on the token embeddings, a vector representation of each vector is created. Position embeddings are used to model the word order in a sequence. Segment embeddings allow the model to divide a sentence into two parts and further distinguish to which sentence each token belongs. A visual representation of different embedding types used in BERT is given in Figure 3.5.

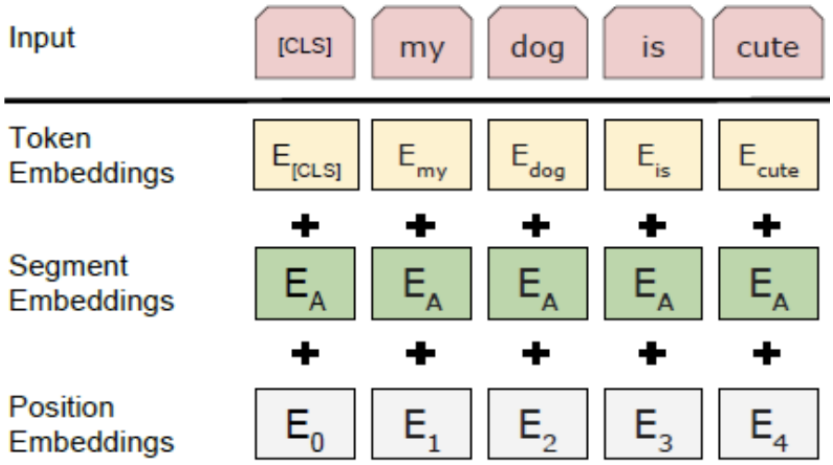


Figure 3.5.: The input representation used in BERT (retrieved from [37])

One of the most important implications of BERT for this work is that it can capture contextual meanings of words with multiple meanings. For example, the word "charge"

might be used in the context of feeding electricity into an electric vehicle, or it might refer to an amount of money for something. The possibilities of models without this contextual distinction are considerably limited, compare in this respect with the word2vec model described in subsection 2.6.1. BERT is released in two model sizes - $BERT_{BASE}$ and $BERT_{LARGE}$ which differ in the number of transformer block layers, embedding dimension, and the number of training parameters [37].

The introduction of BERT significantly altered the NLP landscape due to its efficiency in training and, comprehensive context understanding and applicability for a wide spectrum of NLP tasks. BERT inspired multiple recent NLP architectures, training approaches, and modifications such as RoBERTa [65] and Sentence-BERT [38].

3.4.2. RoBERTa

A *Robustly Optimized BERT Pretraining Approach* (RoBERTa) [65] is an improved version of BERT introduced in 2019. It features the dynamic masking method that creates the masking pattern after every sequence input into the model. In this work, more data for pre-training are used and the influence of various hyperparameters and the size of training data are evaluated. The main difference to the BERT model are as follows [10]:

- A longer training time, a larger batch size, and more training data are used.
- The next sentence prediction task is removed.
- A more extended training sequence is utilized.
- The masking mechanism is dynamically adjusted and the full word mask is used.

3.4.3. Sentence-BERT

In 2019, Reimers and Gurevych [38] presented Sentence-BERT, which is a BERT-based modification to compare sentence embeddings using cosine-similarity. It utilizes the siamese neural network [66] which is also called a *twin neural network*. The motivation behind creating SBERT is that BERT does not compute any independent sentence embeddings and therefore requires extensive time to perform tasks in sentence pair regression such as estimating similarity between multiple sentences [38]. For example, finding two sentences with the highest similarity score from a collection with $n = 10,000$ sentences using the BERT architecture can be performed by passing two sentences through the transformer network. However, this task requires $n(n-1)/2 = 49,995,000$ computations and would take 65 hours to train on a modern V100 GPU [38]. To overcome the problem of sentence embedding, SBERT was developed, which takes only 5 seconds to complete the task described above [38].

The model is trained using the mean of all computed tokens (MEAN-pooling) using *Stanford Natural Language Inference* (SNLI) [67] and the *Multi-Genre NLI* (MG-NLI) [68] data set to perform on classification and regression objective functions [38]. The classification objective

function is calculated by concatenating the token-wise differences between two sentences u and v and multiplying the result of $|u - v|$ with the weight matrix W_t [38]:

$$o = \texttt{softmax}(W_t(u, v, |u - v|)) \tag{3.6}$$

SBERT allows one to process two sentences simultaneously and in the same way. This twin network starts with a BERT layer, followed by a pooling layer enabling the creation of a fixed-size representation for input sentences of varying lengths. In the last step, the cosine similarity between two sentence embeddings is calculated. The structure of the SBERT network is represented in Figure 3.6.

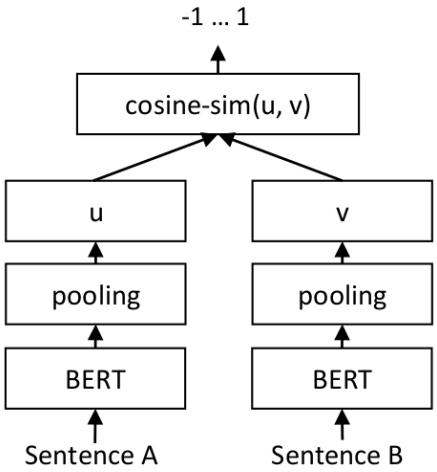


Figure 3.6.: The SBERT architecture with classification objective function (retrieved from [38])

In our work, we employ a distilled version of RoBERTa for sentence embeddings. *Knowledge Distillation* is a process to make models faster, cheaper, and lighter. It describes a traditional approach in which a student model learns to imitate the behavior of a previously trained teacher model. The goal of this approach is to significantly reduce processing speed while accepting minor performance loss. Distilled models based on SBERT achieve 97.5%-100% performance of the original model while the processing speed is improved many times over [69].

3.5. Further Unsupervised Learning Methods

In this section, we briefly introduce unsupervised learning methods for text processing used to answer the third research question.

3.5.1. Clustering

While text classification is a common text analytics approach that relies on annotated data, clustering is an unsupervised learning method that attempts to combine documents with common linguistic features or meanings [70]. One of the most widely used approaches for clustering texts is the *k-means* algorithm [71]. It aims to divide n documents into k clusters. In the course of the k-means algorithm, text documents are subjected to tokenization, stemming or lemmatization, and stop word removal. This is followed by vectorization using bag-of-words or TF-IDF. Finally, the resulting document term matrix is used as the basis for the k-means algorithm.

When using k-means, two factors need to be considered. First, the distance between two text documents is usually measured using Euclidean distance aiming for minimizing squared Euclidean distances. Second, some knowledge of the underlying data is required to determine the value of k, which indicates how many clusters the data must be divided into. Shafiabady et al. [7] applied an unsupervised clustering approach to train the Support Vector Machine for text classification.

We used clustering as one of the unsupervised research methods to answer the third research question. The results can be seen in chapter 6.

3.5.2. Topic Modeling

A collection of documents can be divided into several "topics" that can broadly represent the content of the underlying documents. Topic modeling is an unsupervised learning model that is commonly used to broadly understand the topics behind available documents. It is a text-mining method that clusters the documents into meaningful groups. It has a bright application area, for instance, to analyze news, historical documents, scientific publications, or fiction [72].

Latent semantic analysis is one of the oldest topic modeling methods used to identify relationships between individual words and documents, based on the idea that words with similar meanings occur more frequently in the same documents [50]. LSA is an easy-to-train and tune model, but it is quite slow when applied to large data sets. Later, the *non-negative matrix factorization* (NMF) [73] was introduced to make it easier to inspect the data by removing negative values that do not represent the true world data. *Latent Dirichlet allocation* (LDA) [74] is a model that creates two lower-order document-topic and topic-word matrices from a document-term matrix. LDA uses a stochastic, generative modeling approach that explains a set of observations such as a collection of text documents through latent variables that can be inferred mathematically from observable variables. LDA is often considered one of the most popular approaches for topic modeling and the topic modeling baseline [75], and for this reason, is used in this thesis to answer the third research question.

4. Data Set

In this work, we use a data set from the *National Highway Traffic Safety Administration* (NHTSA). It contains safety-related defect complaints received by NHTSA since January 1, 1995. The database encompasses over 1.3 million unique incident reports. The complaints were collected through multiple communication channels including the Internet website, a hotline, a questionnaire, and consumer letters. The database is freely available for download on the NHTSA website.[1] In this chapter we present statistics on the data set and explain meaningful details about it.

To begin with, we want to briefly explain the current data collection process using the report form on the Internet. The current web-based report submission form begins with the input of the *Vehicle Identification Number* (VIN) to obtain the vehicle information such as make and production year, and provide information about available recalls and safety issues. In the next step, the incident information is collected that includes affected components and a problem description. The affected components and a short description for each of them are given as a multiple choice, whereby one to three components can be selected, and the problem description is manually written with respect to provided general guidelines. Additional details such as the date of the incident, the car crash, and the number of injured people can be filled in. In the last step, personal information is entered and the form is submitted. The original complaints are then published online.

The data were downloaded in several files, as the data are stored in a separate file every five years, and concatenated in the next step. The original data set contains over 1.8 million records and 49 fields. The fields include features such as a unique ID, complaint description, and a corresponding vehicle component. When a complaint reports problems with multiple vehicle components, the entry fields are duplicated except for the component description. More than 300.000 records in the data set refer to two to three components. A rough estimation of $0.3/1.3 \approx 0.23$ tells that around a quarter of all records relate to multiple vehicle components.

Since we are primarily interested in the fields related to the complaint and component description, we briefly describe the specifics of these fields. The complaints are written by individuals to describe an incident with their car. It is therefore expected that many issue descriptions have noise in form of spelling errors or missing punctuation. Additional noise comes from the fact that the issue can have a track history and include tracking numbers and specific words. Regarding the component description, the number of categories is limited, but the data set is built in a hierarchical manner and features a few dozen of root classes and multiple hundreds of sub-categories. The overview of the used fields of the data set is given in Table 4.1.

[1] https://www.nhtsa.gov/nhtsa-datasets-and-apis

Table 4.1.: NHTSA fields used for processing the data.

Field Name	Description
CMPLID	NHTSA's internal unique sequence number
ODINO	NHTSA's internal reference number
COMPDESC	Specific component description
CDESCR	Description of the complaint

Since our thesis primarily aims to investigate the approach for the creation of the annotated data, we simplify the complexity of the data set by flattening the hierarchy down to the root classes, combining semantically similar classes, renaming them to match our needs, and cutting the topics with a low number of samples. The full data preprocessing procedure is described in section 5.4. We also reduce the size of the data set to reduce the processing time by randomly selecting 100.000 documents. Thus, we come up with a data set containing 25 classes and 100.000 unique reports. The distribution of the number of documents with respect to the target classes is shown in Figure 4.1

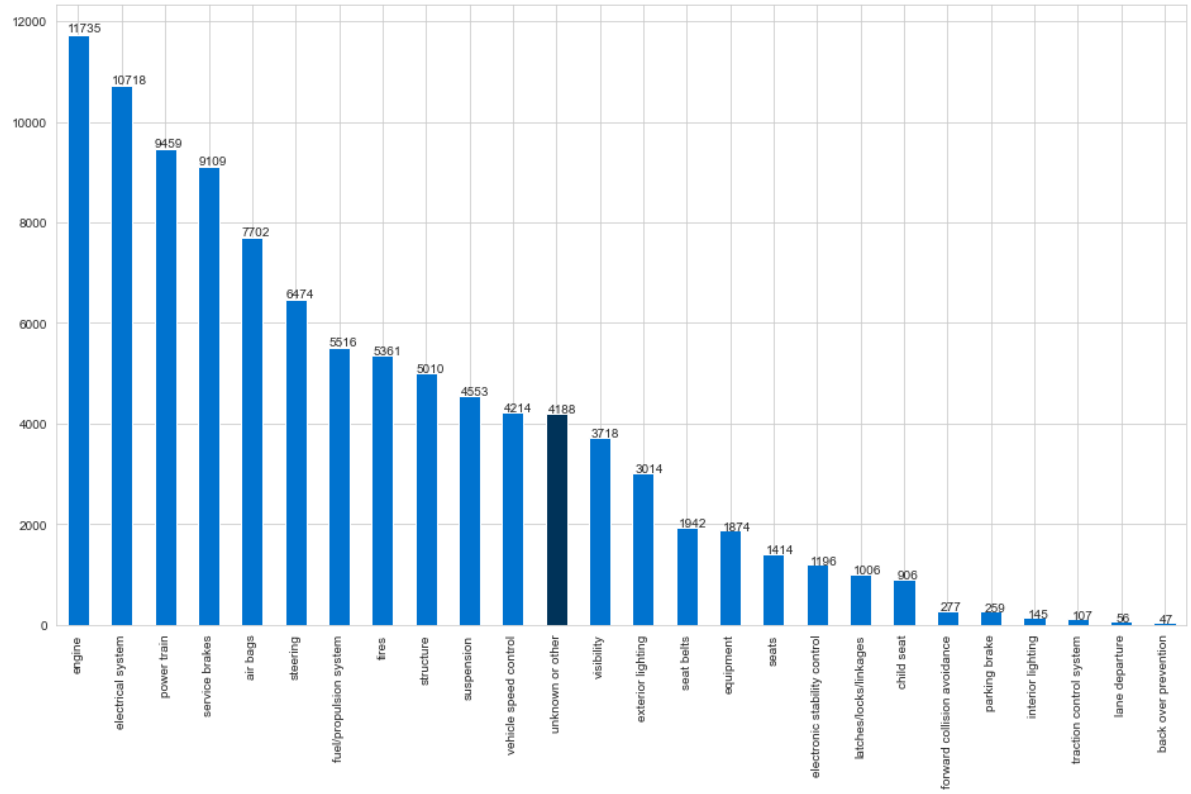


Figure 4.1.: The distribution of documents over 25 pre-defined categories used for classification including the unclassified samples.

The final sample is highly unbalanced containing the data from 47 documents for the "back

over prevention" to 11735 documents for the "engine" class. The sample also contains 4188 unclassified documents. The Table 4.2 presents examples of complaints from the preprocessed data set. From them, we observe that some complaints are correctly identified, while some are assigned to arbitrary vehicle components.

Table 4.2.: Examples of complaints in the data set.

Complaint number	Complaint	Vehicle component
766266	when we took our vehicle in for the brakes to be checked the rotors were so badly damaged from the metal peeling that they had to be replaced. the mechanic told us that for a vehicle with the amount of miles that ours had on it that this should never have happened unless the parts were defective. he told us to check for a recall on this part.	service brakes
10115116	while the vehicle was parked the driver noticed a brownish liquid leaking from underneath the vehicle. the driver took the vehicle to the dealer and the mechanic determined that the filter and other parts needed to be replaced due to corrosion. the vin number entered per consumer did not match the vehicle given.	engine
529088	front seat belt attachments are too low when sitting in seat it is very difficult to see release button.	seat belts
508445	hatchback latch failed causing door to open when secured also replaced front brake rotorspadsshoes also transmission leaked oil and ac compressor failed.	power train
8013957	sunroof popped completely off the vehicle while parked.	visibility

5. Methodology

This chapter describes the methodology used to answer the research questions of this thesis. We begin by defining the terms introduced. Then, the general approach and methodology used to answer each research question are explained. Further, six steps required for classification using the proposed approach are discussed in detail. At the end of this chapter, the application of unsupervised learning methods is described.

5.1. Definitions

To begin, we want to explain the terms that are essential for understanding the concepts of this thesis.

- **Categorization:** In the context of this thesis, the term categorization if used as a synonym for classification.
- **Target class:** The terms class, target class, pre-defined class, and topic are used as synonyms in the context of this thesis.
- **Keyword:** A keyword describes important components of a target class and can be represented by single words, word combinations, and acronyms. They can be synonyms or words that are commonly used when referring to the class. In our work, we usually use multiple keywords to describe the class. For example, when describing the class "engine", we use such words as "engine", "ignition", and "generator".
- **Class description:** A class description is a set of specific keywords describing the class.
- **Class dictionary:** A class dictionary contains a set of pre-defined classes as dictionary keys and one or multiple keywords as dictionary values.
- **Label:** A label indicates that a document belongs to one of the classes. It is an $N : M$ relationship meaning that one document can have from 0 to N labels meaning that it either describes none or multiple target topics. In turn, the same label can be assigned to multiple documents.
- **Context:** A context provides information about the environment in which a described event takes place.
- **Context window:** A context window is a part of a document that has a variable length and surrounds a class keyword. For example, in the case of the class engine, the sentence

"yesterday I went home, and my engine stalled on the uphill slope" can be considered a context window because it has the word "engine" inside of it, which is one of the predefined keywords.

- **Evaluation of context windows:** The process performed by domain experts to assess which of the extracted context windows meet the criteria to qualify as context rules.
- **Context rule:** A context rule is a context window that satisfies required conditions, such as the description of a correct vehicle component and the issue with the regarded component. For this reason, the above sentence is eligible as a context rule. In contrast, the sentence "there is the word engine inside of this sentence", although it contains a prescribed keyword and can be suggested as a context window, would not be accepted as a context rule because it does not describe a problem.
- **Class vector:** A class vector is a vector representation of the class resulting from the embeddings of the context rules for the corresponding class.
- **Similarity score:** A similarity score is a number between -1 and 1 calculated as the cosine similarity between a class vector and a sentence embedding.
- **Minimum threshold:** A minimum threshold is a hyperparameter that is defined as the lowest acceptable similarity score for storing classified documents, which avoids storing redundant documents with low scores.
- **Class threshold:** A class threshold is a hyperparameter that is set individually for each topic and describes a minimum similarity score for a document required to become part of a particular class.
- **Domain expert:** A domain expert is a person with special knowledge, skills, or working experience in a particular area. The person is familiar with a given data set, task, and requirements. In our definition, we assign this feature to all professional employees of the cooperating company.
- **Voter:** A voter, validator, or validation participant is a domain expert involved in the validation process.
- **Validation class:** A validation class is one of the classes used in the validation process.

5.2. General Approach

The idea behind our approach is that semantically similar documents are closer in the vector field. For this reason, we strive to describe the classes as well as possible using the embeddings of the context rules and consequently obtain the class vectors that hold the so-called "ground truth" for each class. It gives our model an understanding of the meaning of the target classes and where to place them in the vector space. A two-dimensional representation of the vector space for the class "engine" is shown in Figure 5.1. Here, a green label "engine failure" in

the center of the image embodies the representation of the class vector for the class "engine". Then we follow our assumption regarding the semantic similarity of documents describing engine problems. For this reason, we compare each sentence of each document with the class vector of the engine to obtain the similarity scores. The higher the similarity score, the closer the documents are to the original class vector. A threshold can be applied to exclude the documents with low semantic similarity.

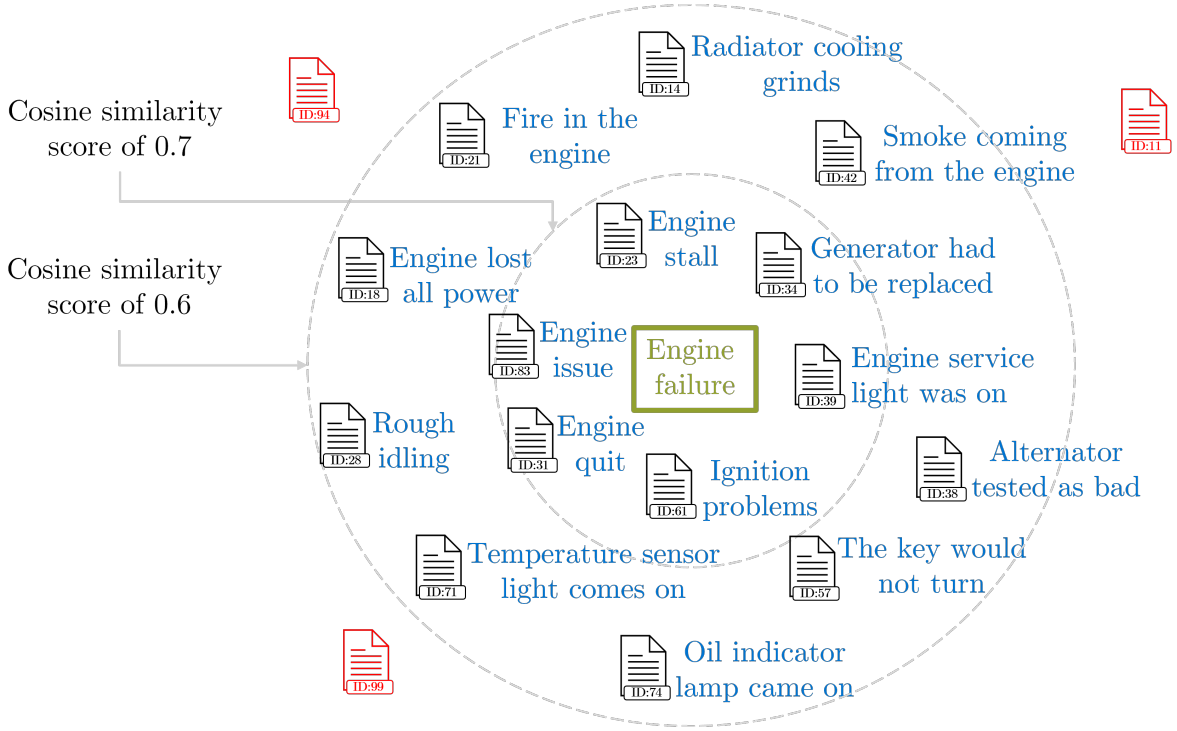


Figure 5.1.: Example representation of a semantic feature space around the class Engine, supplemented by a typical representation of negative sentiment. Grey dashed circles: Similarity threshold for engine-related issues. Blue: Semantically close embeddings for engine-related issues. Black: Document sentences containing these problem descriptions. Red: Outlier document embeddings. Green: Target class embedding (inspired by Schopf et al. [76])

5.3. General Methodology for Answering Research Questions

The first research question describes a continuous investigation of challenges, trade-offs, and limitations throughout the algorithm development process. It involves continuous reflection during each step of data processing, and the answers are intended to form the basis for further research. This continuous process is illustrated in Figure 5.2. An overview of the challenges for each step of the second research question is discussed later in the thesis in chapter 6.

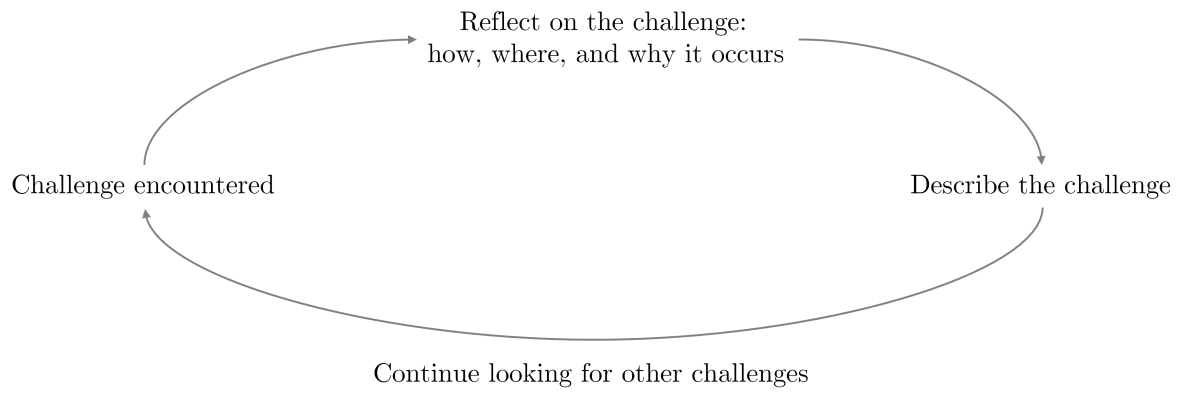


Figure 5.2.: The continuous reflection process on challenges, trade-offs, and constraints to answer the first research question.

The first and second research questions are related to each other in that the answer to the first question comes about through the process of reflection along the way to creating an algorithm for unsupervised data classification. To answer the second research question, a methodology consisting of six steps was developed. It starts with a data preprocessing step in which text cleaning is performed. Next, the target classes for multi-class classification are defined and each class is described by at least one keyword. Then, the class keywords are used to extract context windows for each class and obtain the context rules for them through the evaluation process. Afterward, the context rules are used to obtain the class vectors. After this step, the classification process of the selected documents starts. Finally, the classification results are tested in the validation procedure. The methodology is depicted in Figure 5.3

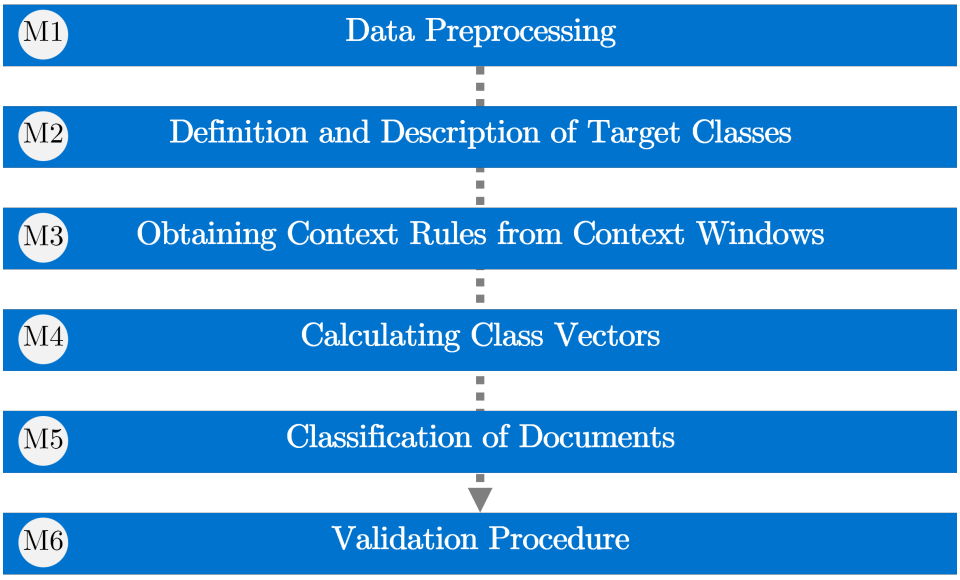


Figure 5.3.: Methodology for unsupervised text classification.

In the third research question, we investigate whether common unsupervised learning methods could be useful for any of the data processing steps described above, such as retrieving keywords for pre-defined classes or detecting new and undiscovered topics that could be added for classification. We apply a descriptive approach to the results to assess their significance and uncover potential for future research. The scheme for this process is given in Figure 5.4.

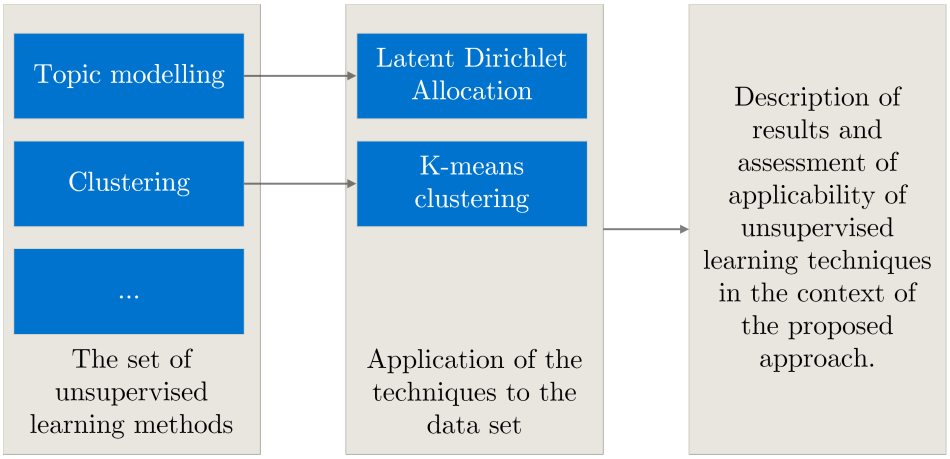


Figure 5.4.: Methodology for applying unsupervised learning methods to facilitate the proposed approach.

5.4. Data Preprocessing

Data preprocessing is an important part of any data science project. It aims to remove incorrect, missing, misleading, or unusable data and provide a clear data set for further processing. The data fed into the model is critical to the quality of the model. In data mining processes, some data may be missing or contain noise. Bad data fed into the model would lead to poor results. As the well-known concept of "garbage in garbage out" states, if the model is fed with poor data, the expected result will also be poor. Since the automotive complaints used for this thesis were collected from individuals, we expect inconsistencies of various kinds in the data, such as missing or incorrect data, misspellings, or missing punctuation.

Several data types may be present in a data set, requiring the application of different preprocessing techniques. The most common data types include the following: numeric, text, categorical, ordinal, date and time, and images. Some approaches, such as removing *Not a Number* (NaN) values, can be applied universally to all data types, while others are tailored to a specific data type. In our case, we are investigating written complaints, and for this reason, we primarily consider text preprocessing techniques.

When choosing the techniques for the text preprocessing, we refer to Danny and Spirling [77] who argued that the choice of the text preprocessing techniques should depend on

the expectations of what outcome is supposed to be produced. It allows us to choose the approaches that preserve the text readability required for the evaluation of the context windows and the validation procedure described later. We primarily preprocess text documents with respect to two features, which are component description (COMPDESC column) and complaint description (CDESCR column).

First, the general preprocessing steps required for all types of data are completed to bring the data in a suitable format for further processing:

1. Five data sets containing customer complaints are concatenated.
2. Rows with missing values in either COMPDESC or CDESCR are removed.
3. Rows with empty values in either COMPDESC or CDESCR are removed.

Then, we treat two columns in a different way depending on the information they contain. For the preprocessing of the component description we select the following steps:

1. Punctuation and special characters ($, %, &, etc.) as well as spaces at the beginning and at the end of the string are removed [77].
2. The text is brought in a lowercase format [77].
3. The data set-specific noise is removed.
4. Extra white space characters are removed [77].

The code snippet representing the preprocessing steps from above is depicted in Figure 5.5.

```
for i, complaint in result.iteritems():
    # strip, lower and remove special characters
    result[i] = re.sub('[^A-Za-z0-9 :,/]+', '', complaint).strip().lower()
    # remove data set specific characters at the end of the complaint
    result[i] = re.sub(r"\W [a-z]{2}$", '.', result[i])
    # remove multiple spaces
    result[i] = ' '.join(result[i].split())
```

Figure 5.5.: Preprocessing steps for the COMPDESC column.

Then we reduce the number of vehicle components used for classification as target classes. As a basis for the selection, we use the vehicle components presented in the COMPDESC column. The documents there are presented hierarchically and comprise several hundred sub-classes. Our goal is to obtain a reasonable number of classes that covers interesting vehicle components and is not beyond the scope of this thesis. For this reason, additional preprocessing steps are introduced:

1. Since the purpose of this thesis is not to study hierarchical classification, we remove an internal hierarchy down to the root component. In this way, we reduce the complexity caused by multiple hierarchical dependencies.

2. We combine classes that have high semantic similarity. For example, the class “tires” was merged with the class “wheels”.

3. We change the names of some classes to have a title that can be better understood.

4. We remove the classes with less than 1000 documents and reduce the number of classes to 25.

Then we describe the preprocessing steps for the CDESCR column. We aim to apply reasonable text reprocessing approaches to make the documents understandable for a model and preserve the readability required for manual evaluation. We apply the following techniques to our text corpora:

1. Special characters ($, %, &, etc.) are removed [77]. We leave only four punctuation characters (., !, ?, and ‘), which are useful for semantics and separating sentences.

2. The spaces at the beginning and at the end of the document are eliminated [77].

3. The text is converted into the lowercase format [77].

4. As part of the noise removal [77], the following text elements are removed: website-like strings, data set-specific characters at the beginning and end of the documents, references to other complaints, multiple punctuation marks, multiple spaces between words, and spaces before punctuation marks. We replace the removed words and character chains with a space to prevent the concatenation of words and then remove multiple spaces.

These preprocessing stages are represented in Figure 5.6.

```
for i, complaint in result.iteritems():
    # strip, lower and remove special characters
    result[i] = re.sub("[^A-Za-z0-9 '.!?]+", ' ', complaint).strip().lower()
    # remove website-like strings
    result[i] = re.sub(r"www\S+", " ", result[i])
    # remove data set specific characters at the beginning of the complaint
    result[i] = re.sub(r"^tl the|^tlthe", "the", result[i])
    # remove references to customer complaint number
    result[i] = re.sub(r"\d{2}\w{6,8}", "", result[i])
    # remove specific characters at the end of the complaint
    result[i] = re.sub(r"\W ?[a-z]{2}$", '.', result[i])
    # remove multiple punctuation marks
    result[i] = re.sub(r"[.!?]{2,}", '.', result[i])
    # remove multiple spaces
    result[i] = ' '.join(result[i].split())
    # remove spaces in front of punctuation marks
    result[i] = result[i].replace(' .', '.').replace(' !', '!')
        .replace(' ?', '?')
```

Figure 5.6.: Preprocessing steps for the CDESCR column.

5.5. Definition and Description of Target Classes

The first step in classifying available data is to make a decision about which classes to put the data into. In our case, the data come from the automotive sector and consist of complaints submitted by individuals who also selected the vehicle components that are addressed in their complaint. Our task is to identify representative vehicle components for classification. Therefore, we use the distribution statistics from the data set containing an estimated number of documents for each component. However, we use it only as a guide and not as a basis for validation because we cannot guarantee that the information in the complaint matches the component choices of the individuals. Therefore, we use the information as a rough estimate of the number of documents per component that are addressed in the data set.

Originally, several hundred classes are present in the data set. We simplify their distribution by flattening the class hierarchy and assigning the documents of the child classes to the parent class. We also remove underrepresented classes with less than 1000 documents. In this way, we narrow the number of classes to 25 classes. An overview of the target classes used in this thesis is shown in Figure 5.7.

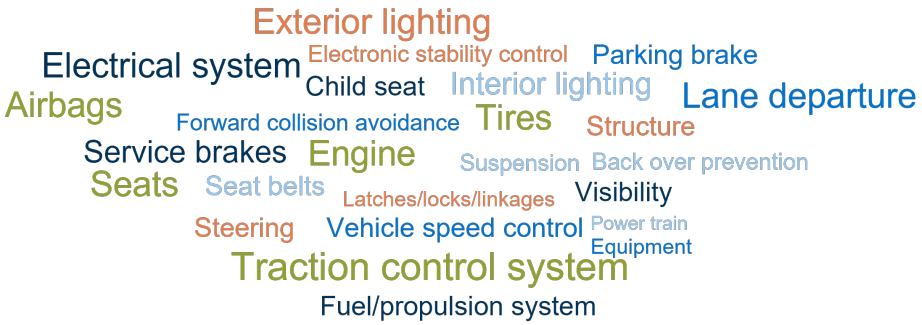


Figure 5.7.: Target classes used for the classification represented as a word cloud.

Then we describe the classes using keywords. Keywords describe the class from multiple perspectives and can be represented by synonyms to the class title or by words commonly used when referring to the class. Initially, we attempted to use the information from the data set to create class descriptions based solely on the titles of the underlying hierarchically dependent sub-classes. However, our experiments showed that most of these words are either too specific or rarely used by the complainants and for these reasons are not suitable for our goals.

To determine appropriate keywords for each topic, we followed the expert knowledge extraction procedure described by Haj-Yahia et al. [3]. we take the role of an industry expert who can describe the class only by its title. The proposed guidelines for creating class descriptions are presented below:

1. Each class must be described by at least one keyword.

2. We use specific words that have close semantic meaning to the target class, e.g. "engine"

and "motor", and can be distinguished from semantically related classes such as "parking brake" and "brakes".

3. We use representative words that are expected to be used in the underlying data set. We take into account differences such as formal and informal tone, e.g., in our case the word "light" is expected to be used instead of "illumination".
4. A keyword can contain a single word, e.g. "engine", or a word combination, e.g. "adjustment rod".
5. Different writing options of the same term can be included, e.g. "adjustment rod" and "adjusting rod".
6. Keywords do not contain special characters such as hyphens removed in the preprocessing step, e.g. we use "self driving" instead of "self-driving".
7. Some keywords may be separated and may be joined together. For this reason, we often use both variants, e.g. "powertrain" and "power train".
8. Abbreviations can be used as keywords. For example, "ABS" is more often used than "anti-lock braking system".

For the description process, industry experts can rely on their work experience and understanding of the problem. However, various supporting sources of information can be utilized for the process of creating class descriptions. The following sources are used in our thesis:

- To begin, we use some related keywords from the hierarchical dependencies of the underlying data set.
- We then make extensive use of online search engines such as Google to find additional descriptive keywords by, for example, appending the word "synonym" to the name of the class or finding similar words in Wikipedia.
- We also rely on visual representations by adding "word cloud" or "description" to our class name. For example, we use queries like "Word Cloud Engine" and "Engine Description" on the "Images" tab to explore meaningful components of the engine.
- To expand the list of class keywords, we also randomly select a few documents from the data set and manually explore the common vocabulary.

Expertise is crucial when identifying appropriate keywords, as dealing with the specifics of the field and predicting the vocabulary of the target audience require special skills. Undoubtedly, the quantity and quality of the selected keywords have a significant impact on the results of the next steps. When domain experts are involved in the process of the class description, they can describe each class in a short time by using their expertise about the domain and the data set and selecting mainly specific and representative keywords and reducing the number of rare and redundant ones. An example of classes and applied keywords is given in Table 5.1.

Table 5.1.: An example of classes and corresponding keywords.

Class	Keywords
Airbag	airbag, air bag, knee bolster, inflator, clock spring, srs
Engine	engine, ignition, screen filter, pressure sensor, generator
Seats	seat, carseat, headrest, slide adjuster, adjuster rod, cushion
Tires	tire, wheel, tread wear, flat spot
Traction control	traction, stabilitrak, vsc light

5.6. Obtaining Context Rules from Context Windows

In this step, we use the preprocessed data from section 5.4 and produced class descriptions from section 5.5. Similarly to Liu et al. [4], we use identified keywords to extract context windows for each class. Context windows are identified based on the selected keywords and represent a span of text with one or multiple keywords in it. They can have variable lengths based on either the number of tokens or the number of sentences. Therefore, we experiment with two approaches for extracting the context windows, token-wise and sentence-wise. Token-wise means that the identified keyword in the document is set as the center and additional tokens are taken to the left and right of it. We test different lengths of fixed-size context windows with 5 to 15 tokens to the left and right of the identified keyword. In this case, we encounter the problem that many context windows contain a truncated representation of sentences containing a few last words of one sentence and a few initial words of another sentence. This makes the context windows ambiguous and difficult to interpret for human evaluation, which is required in the next phase. The sentence-wise extraction used in this thesis assumes that the context can be described in a single sentence of a document. Once the keyword is identified in the document, the surrounding sentence enclosed by punctuation marks is used as the context window. The intuition behind using the sentence-wise approach is that taking into account the linguistic peculiarities of how we humans convey our thoughts in a text format, a sentence contains at least one statement and is complete in itself. For this reason, it is much easier for a human reader to understand and interpret.

The procedure for extracting context windows is described as follows: First, duplicate complaints are removed for each class and its keywords. Then, all complaints in the data set are shuffled to diversify the source of context windows. Next, a spaCy tokenizer is applied to each document. In the next step, a spaCy matcher is initialized to find matches between the class keywords and the lemmatized version of the document tokens. When a match is found, the sentence containing the matching keyword is identified. This sentence is then added to the list of context windows for the representative class. If multiple matches exist in the same document, only the first match is counted as a context window to diversify the provenance of the data. We also limit the length of the context window to 100 tokens to exclude particularly long sentences. The process then continues until 50 context windows are identified for the class. These are then saved as a ".csv" document along with the full complaint text, matching keyword, and class name, and the extraction process for the next class begins.

```
for class_name, class_keywords in dict_of_classes.items():
    # remove duplicates and shuffle complaints
    shuffled_complaints = df.drop_duplicates('CDESCR').sample(frac=1)
    # create spacy matcher
    matcher = create_spacy_matcher()
        for i, doc in shuffled_complaints['CDESCR'].iteritems():
            tokenize_document()
            find_keyword_matches()
            identify_sentence_of_match()
            extract_the_sentence()
            append_to_the_list()
```

Figure 5.8.: Pseudo code for the context windows extraction procedure.

In the following step, the context windows are evaluated by an expert with domain knowledge. Again, we assume the role of domain experts to perform the evaluation. To be considered a context rule, a context window must satisfy task-specific conditions. In our case, it must both contain information about a vehicle component and describe a problem related to that component. The evaluation process consists of thoroughly reading through the extracted context windows and deciding whether a sentence qualifies as a context rule or not. Two previously established criteria are used for the evaluation. In the positive case, the context window is marked with a dummy 1; if it does not meet at least one of the criteria, it receives a dummy 0. Examples of the evaluated context windows for the "engine" class are presented in Table 5.2.

Table 5.2.: Examples for the evaluation of context windows for the "engine" class.

Context window	Evaluation
while driving at any speed the engine will stall without a prior warning.	1
vehicle experienced a fire in the engine.	1
purchased car july '09 with 64000 on engine.	0
we have 17k on this engine and all components are supposedly on a full and extended warranty.	0
dealer replaced relay by the radiator and the computer power train module but the problems remain.	1

5.7. Calculating Class Vectors

Now we have context rules for each of the 25 classes and can calculate the class vectors. The number of context rules varies for each class because we extract a fixed number of context windows, but in the evaluation procedure, many context windows are sorted out.

After evaluation, we get between 2 and 43[1] context rules for our classes. These context rules are further used to obtain the class vectors. The class vector embodies the task-specific representation of a target class in the vector field. In our case, it has the properties of both a component description and a related issue.

The process of calculating the class vectors starts with reading the files containing the evaluated context windows. Then a pre-trained model is applied to obtain the embeddings of the context rules. These embeddings are then used to compute the class vectors. The class vectors are finally saved as individual files in ".npy" format. This process is explained in more detail below.

In this thesis, we investigate three pre-trained models for text embeddings. Two of them consider context and are based on SBERT [38], and the third model is a non-contextual tok2vec model by spaCy[2], which is used as a benchmark. The SBERT models we use are *all-MiniLM-L6-v2* and *all-distilroberta-v1*, which are commonly employed for cosine similarity tasks. Both models are state-of-the-art in text embedding but differ in size and processing speed. The comparison of the models used is presented in Table 5.3. A complete list of available pre-trained models, including their properties and explanations, can be found on the SBERT website[3].

Table 5.3.: Comparison of used pre-trained SBERT-based models.

Characteristics	all-MiniLM-L6-v2	all-distilroberta-v1
Performance Sentence Embeddings (14 Datasets)	68.06	68.73
Speed	14200	4000
Max Sequence Length	256	512
Dimensions	384	768
Size	80 MB	290 MB
Huggingface downloads in October 2022	2,073,697	89,538

The method of obtaining the class vectors is further explained. We apply one of the pre-trained models to the context rules to obtain their embeddings. The class vectors are then calculated as the average vector of their context rules. This approach is called *mean pooling* or *average pooling* and is widely used by many NLP researchers [37, 38, 78]. A code snippet depicting this step is shown in Figure 5.9.

[1]The class "electrical system" received 58 context rules because it was used for experimental purposes and the rules came from several tests.

[2]https://spacy.io/api/tok2vec

[3]https://www.sbert.net/docs/pretrained_models.html

```
for context_rules_df in context_rules_dfs:
    df = context_rules_df[['evaluation', 'cw_embedding']]
    class_vector = 0
    for i, val, vector in df.itertuples():
        if val == 1:
            class_vector += vector
    number_of_context_rules = len(df['evaluation'])
    class_vector = class_vector/number_of_context_rules
```

Figure 5.9.: A code snippet for the calculation of class vectors using the average pooling approach.

5.8. Classification of Documents

The document classification procedure proposed in our approach is similar to the way people classify documents by assigning words from documents to similar concepts gained through experience and reflection on the lexical and semantic meaning of words and phrases in different situations [3]. The intuition behind the process of bringing together the embeddings of the document sentences and class vectors is presented in Figure 5.10. In short, the complaint classification process comes down to comparing each of the class vectors to each sentence of each document and measuring similarity scores for each pair of vectors.

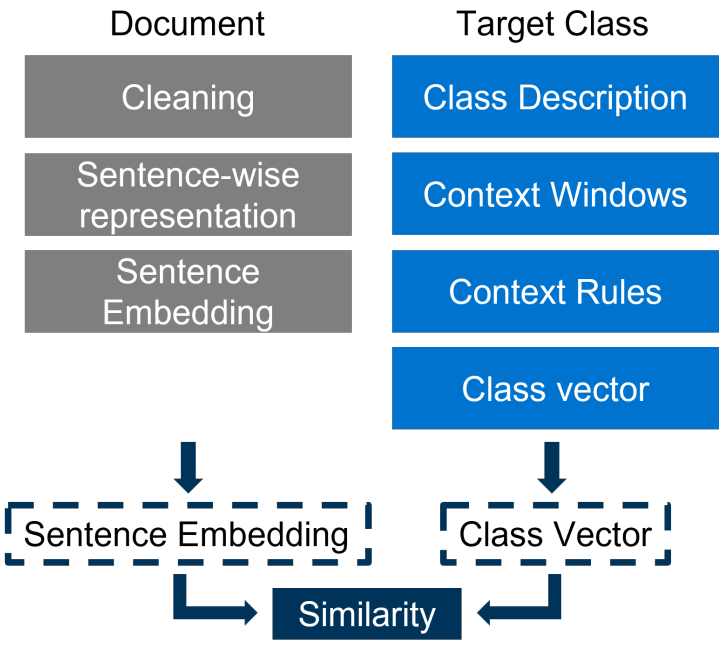


Figure 5.10.: The intuition behind the proposed approach is to explain the target class features to the model and then determine the semantic similarity between the sentence embedding and the class vector by computing their cosine similarity. Inspired by [3].

For classification, 100,000 unique documents are randomly selected from the data set. The reason for this decision is to obtain a representative number of documents while keeping the processing time within reasonable limits. Then, all 25 class vectors are loaded into a matrix and a sentence transformer model is initialized. The classification process begins by tokenizing a document using the spaCy tokenizer. Then, the document is divided into sentences and a sentence embedding model is applied. Then, a similarity score is determined for each sentence of each document by computing the cosine similarity between the sentence embedding and the class vector. We do not store all the classified results, as this is not space efficient. If we assume that a document consists of three sentences on average, we have 25 classes and 100000 documents, $3 * 25 * 100,000 = 7,500,000$ entries need to be stored. However, many of these entries contain low or even negative similarity scores, indicating an admittedly low relationship to the class vector. For this reason, we store only those documents whose threshold is above 0.5 to avoid redundant entries. Classified documents with a similarity above the defined threshold are then appended to the list of tuples, which is converted into a Pandas DataFrame at the end of the process and saved as a '.csv' file. The following data is stored: a complaint index, a complaint text, a document sentence, a class index, and the similarity. The complaint index is represented as a unique complaint identifier required to identify the document in the original data set. The full complaint text is stored in the "complaint" column. The document sentence representing the class is stored in the "text_span" column. The "class_index" column contains the coding of the target classes and is represented as integers between 0 and 24. The similarities themselves are stored in the "similarity" column. If a document does not contain classified records, it is assigned the class index 9999. The unclassified documents are stored together with the classified data. A simplified version of the classification algorithm is shown as a pseudo-code in Figure 5.11.

```
data = []
for document in documents:
    for sentence in document:
    find_cosine_similarity_between_sentence_and_each_class_vector(
        sentence,
        class_vectors
    )
    data.append((complaint, sentence, class_index, cos_similarity))
# save data as a pandas DataFrame
pd.DataFrame(data).to_csv()
```

Figure 5.11.: A pseudo-code describing the procedure for classifying complaints.

5.9. Validation Procedure

In order to obtain measurable results and provide a quantifiable and comparable overview, the validation step was introduced. It is worth mentioning that the data annotated by vehicle owners are only used to obtain a distribution of vehicle components and are not considered as "ground truth", since the completion of the accident report may take place under stressful

conditions and the choice of labeling is therefore prone to errors. The validation process with domain experts was necessary to evaluate the performance of the proposed approach, to estimate the requirements for validation including the time required, and to obtain a complete picture of the process in order to propose further research steps, which are described in chapter 7.

In our thesis, the validation consists of four rounds in which the classification results of three different embedding models are given out to voters with experience in the automotive industry. We select a subset of classes for validation to balance the representativeness of the results and the scope of this thesis. We report validation results for five classes selected based on the following criteria:

- The meaning of the class must be easy to understand or even self-explanatory, even for people without extensive industry experience, since we want to reduce bias regarding the required expertise in the field and primarily assess how well the proposed algorithm performs in the first phase. Four out of five classes are selected according to this principle. However, the "Traction Control System" class was considered to require more expertise, and the results of this class were assigned only to non-student employees.
- We choose classes with a representative number of context rules, over 15 in our experiments, to obtain a well-described and diversified class vector.
- The classes are taken from the different sides of the topic distribution data set used for classification to investigate whether the proposed algorithm can well identify the documents with the expected high and a low number of documents.
- We also want to investigate the importance of specific and representative class descriptions. For example, the class "Seats" was selected for validation because it has a high semantic similarity to the classes "Child Seat" and "Seat Belt".

The selected classes, their definitions provided to the voters during validation, the number of context rules used for the class vector, and the distribution rank based on the expected number of documents in the original subset are presented in Table 5.4.

An important hyperparameter for determining thresholds for meaningful results was the similarity score. Given a unique combination of NLP methods applied to a particular domain, no prescribed similarity score for our task was found in the scientific literature to separate classified documents from those with low similarity to the class vector. Our solution was to experimentally determine the balance between the similarity score and the number of documents. In our thesis, two similarity score thresholds of 0.6 and 0.7 were selected for validation.

Further, the process of inviting industry experts to participate in the validation procedure is presented, as this was an important part of this thesis. The cooperating company provided a solid basis for access to people with expertise in the automotive sector. We mainly use a combination of convenience sampling [79] and snowball sampling [80] to find the participants. The author of this thesis attended two company social networking events in Frankfurt and

Table 5.4.: Selected classes for the validation procedure, their class descriptions for the voters, the number of context rules for each class, and their position in the distribution of a total of 26 classes, including unclassified documents, based on the number of documents in the corresponding class.

Class name	Class description given to the voters	Number of context rules	Position in distribution
Airbag	The "Airbag" class comprises issues regarding the airbag system, its functioning, deployment, and corresponding system warnings.	43	5-th
Engine	The "Engine" class describes complaints that affect the vehicle engine and its components, such as, but not limited to, the engine cylinder, starter, alternator, drive belt, and connecting rod. It also includes relevant system warnings.	33	1-st
Seats	The "Seats" class includes customer complaints and problems with the functionality of driver and passenger seats, including relevant system warnings. It does not include problems with seat belts or child seats.	19	17-th
Tires	The "Tires" class includes problems with vehicle tires, wheels, and rims, e.g. malfunctions, a worn wheel, or corresponding system warnings.	23	8-th
Traction control	The traction control system is designed to prevent wheel spin from causing loss of traction. This class covers complaints and problems related to the operation of a vehicle's traction control system, including related system warnings.	39	24-th

Ludwigsburg to establish face-to-face contact with colleagues. Participation in social formats such as speed dating allowed for initial contact with many colleagues. Communication with colleagues often took place in a friendly and relaxed environment and encouraged casual conversation about various topics such as work experience and current projects. LinkedIn profiles were shared to better memorize the person. The initial conversations with colleagues were helpful in getting to know teammates interested in Data Science and NLP-related conversations. In this way, the network quickly expanded, and during the aforementioned events, approximately 40 contacts were made with colleagues ranging from student to partner level. Additionally, an internal communication tool, Microsoft Teams, was used to make connections with other colleagues interested in data-related topics. Some relevant contacts were also made through networking at the company office in Munich.

When the algorithm was finalized and the data were prepared for validation, colleagues were contacted to ask for their help with this process. Since the company's employees are located in different cities, communication took place mainly through Microsoft Teams and email and was handled as follows:

1. **Initial contact:** The first message was sent via Microsoft Teams and included a personal reference to a shared experience during one of the social network events and a question asking if a person had time to participate in a 15-minute validation process. A brief description of the task was given.

2. **Assignment:** In case of a positive response, an email was sent with materials for the validation. It included a project and task description, introduced a vehicle component, and explained the validation requirements. One week was given to complete the validation.

3. **Reminder:** Two days before the deadline, a short friendly reminder to complete the validation questionnaire was sent via Microsoft Teams.

Information about the voters, their role in the company, response history, and deadlines was continuously updated in a dedicated Excel document. The results for each class of each embedding model were validated by three voters. This number of validators was intended to provide an optimal balance between the reliability of the results and the potentially reachable experts in the given time period. In our setting, 16 different samples were created for validation, requiring 48 experts to participate. We allowed the same person to participate in multiple rounds of validation. During the period of this study, 53 employees were invited to complete the questionnaire, 44 of whom participated. Thus, we can report an admittedly high response rate of 83%. The number of participants according to their role in the company is shown in Table 5.5.

The rounds of validation are described further. In general, results for the five previously mentioned classes were given to the voters, and each of them received a document containing 100 classified sentences with the task of reading each of the sentences and judging whether it belongs to the target class by giving it a 1 or otherwise assigning it a 0. We divide this procedure into six steps. In the first step, the raw complaints for each of the validation classes

Table 5.5.: Participants in the validation procedure by the company role.

Role in the company	Number
Student	9
Junior Consultant	10
Consultant	24
Senior Consultant	1
Grand Total	44

are extracted and saved as a ".csv" file. Then, the same number of complaints classified in other classes are extracted and added to the validation class complaints. A column containing either 1 or 0 is included to distinguish them. The complaints are then shuffled to be presented to voters in a random order. Excel documents are then created containing a column with the complaints and a blank column for the voters. These Excel documents are then sent to the validation participants, and the results are stored in such a way that the responses of the different participants can be distinguished. Finally, the responses are aggregated and prepared for assessment. An overview of the validation layers is given in Figure 5.12.

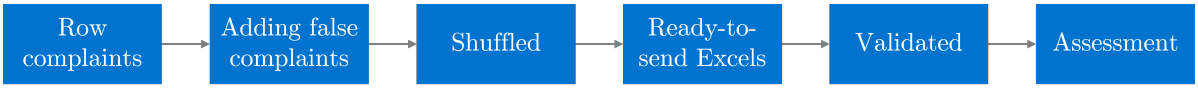


Figure 5.12.: Six layers of preparing the documents for validation.

A single embedding model was used in the first, second, and fourth validation rounds, and 50 predicted complaints and 50 complaints assigned to other classes were selected for each class. The falsely predicted complaints are not selected completely randomly from the data set. Instead, for each of the validation classes, their class vector is compared to other class vectors, and three classes with the greatest similarity are identified. Then, 50 complaints are randomly extracted from these classes and added to the predicted complaints, and shuffled. The idea behind this decision was to increase the difficulty of the validation by providing semantically related examples from other classes. The third validation round is a special case as it tests two embedding models in parallel. The idea was to investigate whether representative and interpretable results could be obtained with fewer documents for validation. In this case, voters were again given the 100-sentence files, but only 25 true and 25 false documents were selected for each model. For the fourth round of validation, we take the documents marked as "unclassified" from the data set to evaluate our approach with completely unknown data, as is the case in practice. For this last round, we report the results for one class. A summary of the validation procedure can be found in Table 5.6.

The validation outcomes are then aggregated in the assessment step according to the majority voting principle [81] to obtain the results that are considered true. In our case, the decision made by two validators for each sentence is considered to be true. These results are then used in combination with the results of the algorithm to produce classification reports. A

Table 5.6.: Overview of the validation procedure.

Criteria	Round 1	Round 2	Round 3	Round 4
Model name	all-MiniLM-L6-v2	all-MiniLM-L6-v2	all-distilroberta-v1 and tok2vec	all-MiniLM-L6-v2
Number of documents for classification	100,000	100,000	100,000	75,445
Number of classes for validation	5	5	5	1
Number of voters per class	3	3	3	3
Total number of voters	15	15	15	3
Similarity score threshold	0.7	0.6	0.7	0.7
Number of predicted complaints per class	50	50	25	50
Number of false complaints per class	50	50	25	50
Total number of complaints per voter	100	100	100	100

Fleiss' Kappa [82] value is also provided for each of the classes, indicating inter-rater reliability and assessing the reliability of agreement between voters. These results are presented in section 6.4

5.10. Application of Unsupervised Methods

In this thesis, we test the application of common unsupervised learning approaches to solve the underlying task. We consider the results from two perspectives. First, we want to investigate whether unsupervised learning methods are able to identify specific classes without the assistance of domain experts. Second, assuming that domain experts are available, we want to evaluate whether the results can facilitate any of the manual steps performed by domain experts for the proposed approach. For this purpose, we use k-means clustering [71] with 25 clusters and topic modeling using LDA [74] with 25 topics referring to the same number of target classes as defined for our task. Both approaches are applied to the same subset of 100,000 documents used for classification.

In the first case, we have unlabeled documents and the task of classifying them based on vehicle component names. The expertise is not available, which gives us the opportunity to evaluate whether the underlying task can be solved using unsupervised learning methods only, which is a reasonable decision considering no other options to address the underlying task. In this way, we can get a good initial assessment of the available topics. However, the industry perspective requires classification into specific topics, and specificity is not guaranteed by unsupervised learning methods. Our approach aims to bridge this gap by building on the experience of industry professionals who not only know the class requirements, but can also describe them through class titles.

In the second case, we have the same task and an expert who can define the classification topics and is eager to use the approach we propose. It includes three phases where the manual involvement of the expert is required. These phases are the creation of class descriptions, the evaluation of context windows, and the validation of the results. We assess the outcomes with respect to each of these phases to see if the manual effort can be reduced with help of unsupervised learning methods.

The results of using unsupervised learning methods are described in section 6.8.

6. Results

This chapter highlights the results of this thesis by answering the research questions. We start by describing the main components of the developed algorithm for obtaining training data for text classification. Then, we present the identified challenges throughout the pipeline involving the proposed approach. We then present the results of each of the processing steps. The validation results for each of the four validation rounds are further described in detail. Afterward, a summary of the results is given and the comparison of the time required between the common manual annotation and our approach is presented. Following this, a proposed validation process in practice is described. Finally, we also present the application results of the unsupervised learning methods.

6.1. Developed Algorithm

In this section, we present the developed algorithm for obtaining training data using the combination of expert knowledge and various NLP techniques. The algorithm can be accessed on the GitHub of the author of this thesis.[1]

The files of the created algorithm are further introduced:

- *main.py* initializes the algorithm.
- *component_dict.py* contains class dictionaries.
- *preprocessing.py* is used to concatenate the output data and get preprocessed data.
- *reading_preprocessed.py* reads the preprocessed data.
- *context_windows.py* is used to extract context windows for each class.
- *class_vectors.py* creates class vectors for each class from the corresponding context rules and saves them as ".npy" files.
- *embedding.py* contains the classification algorithm that uses class vectors and document corpora for classification.
- *validation.py* is used to extract true and false complaints for validation classes.
- *visualization_results.ipynb* is a Jupyter Notebook containing visualizations of the validation results.

[1] https://github.com/andreikreinhaus/master_thesis

- *unsupervised.ipynb* is a Jupyter notebook containing the results of applying unsupervised learning methods to the data set.

We also present the folders stored in the working directory and the files they contain:

- The *class_vectors* folder contains calculated class vectors for tested models. For each model, vectors are calculated for 25 classes of pre-defined vehicle components. The calculation of each class vector is done by averaging the embeddings of the context rules of each class. The number of context rules used is specified for each class vector.
- The *context_windows* The folder contains extracted and evaluated context windows. For each class, context windows containing a keyword from the component dictionary and surrounded by a context from a complaint are evaluated manually. The list of used keywords is given in component_dict.py
- The *nhtsa_complaints* folder contains the raw data of NHTSA complaints received from 1995 to 2022. The complaints are open source and can be downloaded from the NHTSA website.[2]
- The *preprocessed* folder contains NHTSA complaints after an initial preprocessing. The preprocessing algorithm is given in preprocessing.py
- The *similar_text_spans* folder contains the product of the classification algorithm for each of the models used. The similarity score for each sentence of each complaint is given.
- The *validation* folder contains information about all validation rounds and stores the questionnaires used, the voters' answers, and the aggregated assessment data.

6.2. Challenges

Following the order of the research questions, we start the description of our findings by presenting the challenges encountered during the development of the proposed algorithm. The identified challenges are associated with the respective methodology steps and are presented in Table 6.1 These insights are intended to form the basis for further research. A full description of each challenge can be found below in section 7.2.

Title	Challenge	Processing step
C1	Selecting preprocessing techniques and applying them in a sequence that ensures time efficiency	Data preprocessing
C2	Maintaining readability	Data preprocessing
C3	Choosing the number of target classes	Definition and description of target classes

[2]https://www.nhtsa.gov/nhtsa-datasets-and-apis

C4	Defining an optimal number of keywords per class	Definition and description of target classes
C5	Picking precise keywords for the class descriptions	Definition and description of target classes
C6	Creating hierarchical dependencies between classes	Definition and description of target classes
C7	Adding new classes or changing the definitions of target classes	Definition and description of target classes
C8	Being familiar with the language style and words used in the data set	Definition and description of target classes
C9	Using acronyms as keywords	Definition and description of target classes
C10	Searching for context windows word-wise if not enough context windows are found	Obtaining Context Rules from Context Windows
C11	Choosing the length and representation of context windows	Obtaining Context Rules from Context Windows
C12	Ambiguous meaning of context windows	Obtaining Context Rules from Context Windows
C13	Determining the optimal number of context rules to describe the class	Obtaining Context Rules from Context Windows
C14	Evaluating the results in terms of the ratio between the words used in a class dictionary and the number of keywords actually used for context rules	Obtaining Context Rules from Context Windows
C15	Reducing time for a manual evaluation of context windows	Obtaining Context Rules from Context Windows
C16	Exploring alternative methods for obtaining context rules	Obtaining Context Rules from Context Windows
C17	Selecting a pre-trained model for obtaining text embeddings	Calculating class vectors
C18	Exploring the possibilities of training a tailored model	Calculating class vectors
C19	Deciding on the method for calculating the class vectors using context rules	Calculating class vectors
C20	High similarities between class vectors	Calculating class vectors
C21	Exploring other approaches beyond computing the class vector	Calculating class vectors
C22	Selecting the approach to classify documents	Classification of documents
C23	Using time-saving code writing techniques for the algorithm	Classification of documents
C24	Determining a minimum threshold for saving classified documents	Classification of documents

C25	Adjusting a class threshold for each individual class	Classification of documents
C26	Exploring additional features of the data set to enhance the classification	Classification of documents
C27	Identifying multiple topics in a document	Classification of documents
C28	Examining false statements	Validation procedure
C29	Establishing the number of statements required for validation	Validation procedure
C30	Determining reasonable metrics for the validation sample	Validation procedure
C31	Improving the validation procedure	Validation procedure
C32	Exploring portability of results to other tasks	General
C33	Reducing labor time required for the suggested approach	General
C34	Training a text classifier	General
C35	Improving the model over time	General

Table 6.1.: Overview of the challenges along the classification process using suggested approach.

6.3. Classification Results

In this section, general results for individual steps of the proposed algorithm are presented.

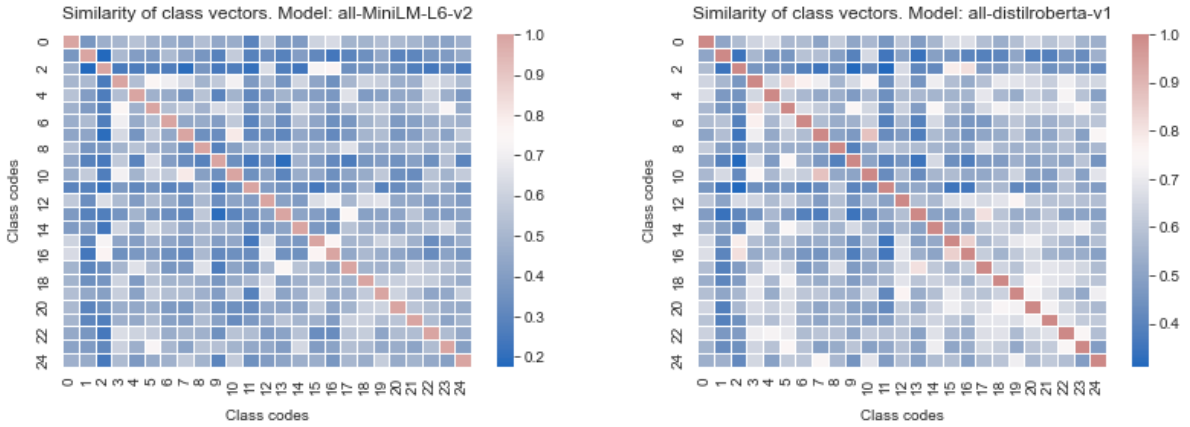
At the beginning of the classification process, class descriptions are created with a varying number of keywords. Only a subset of these keywords later appear in context windows and further in context rules, which means that some keywords from the class description are not utilized for the class vectors. This demonstrates that a small number of keywords can be sufficient to describe the class. The comparison between the number of keywords used to describe the validation classes and the number and frequency of keywords used in the context rules are shown in Table 6.2. From this table, we can see that although 18 keywords can be used for the class description, as in the case of the class "engine", only four keywords appear in the final context rules. We also see that for our validation classes, between two and five keywords end up being used for the context rules and the corresponding class vectors.

In one of the following steps, we convert the obtained context rules into 25 class vectors. We want to make sure that these class vectors describe different topics. For this purpose, we compute the cosine similarities between all 25 classes. The similarities between the class vectors are represented in the form of a heatmap and shown in Figure 6.1. The absence of red squares or hot spots indicates that the class vectors do not have high similarity to each other and describe different subjects.

In the next step, we perform the classification and report the number of classified documents with respect to two SBERT-based embedding models, two similarity score thresholds, and five validation classes. We find that small changes in the similarity score thresholds lead to

Table 6.2.: Statistics on the keywords used for the context rules of validation classes.

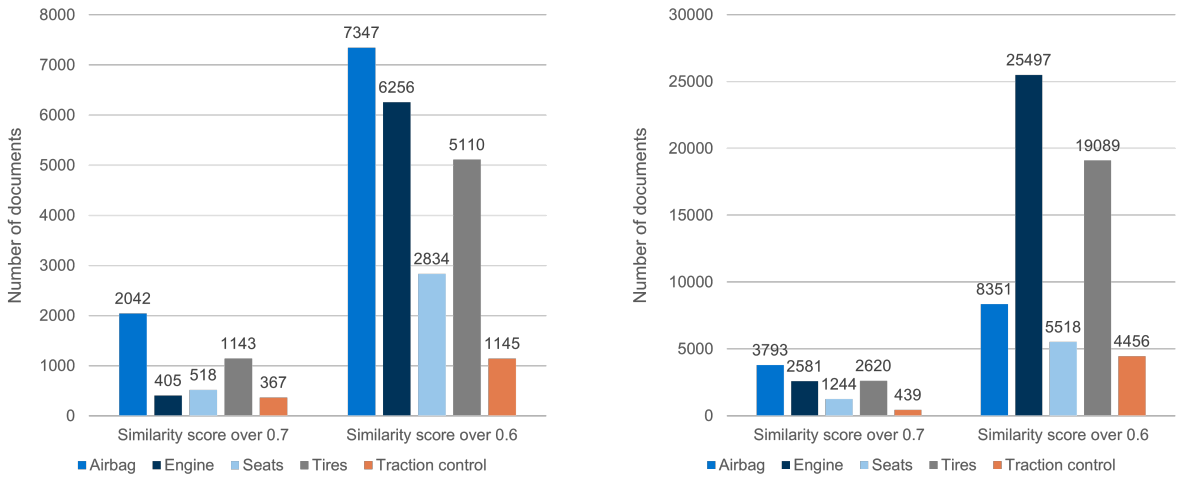
Class	Number of keywords in a class description	Keywords appearing in context rules	Respective number of context rules containing the keyword
Airbag	7	airbag, airbags, inflator, inflators, srs	25, 12, 1, 1, 4
Engine	18	alternator, engine, ignition, radiator	1, 25, 3, 4
Seats	7	seat, seats	17, 2
Tires	4	tire, tires, wheel, wheels	7, 7, 6, 3
Traction control	3	stabilitrak, traction	6, 33



(a) Class similarities heatmap for all-MiniLM-L6-v2.

(b) Class similarities heatmap for all-distilroberta-v1.

Figure 6.1.: Heatmap of class vector similarities for two SBERT-based embedding models used for result validation. Low similarity indicates that the class vectors describe different topics, while high similarity scores may represent the same concept. In the latter case, the class vectors may unintentionally describe the same subject, and it is recommended to reformulate their context rules. The absence of red cells in both figures, except for the comparison with the same class vector, shows that each class vector describes a separate topic.



(a) Model: all-MiniLM-L6-v2.

(b) Model: all-distilroberta-v1.

Figure 6.2.: The number of classified documents from the sample of 100,000 documents by model and similarity score threshold.

significant changes in the document counts. For example, using the *all-MiniLM-L6-v2* model and a threshold of 0.7 results in 2042 documents for the "airbag" class and 405 documents for the "engine" class. Lowering the threshold by 0.1 to 0.6 increases the number of documents to 7347 for the "airbag" class and 6256 for the "engine" class. The same principle applies to the *all-distilroberta-v1* model. A complete overview of the number of documents identified by each model for each class can be found in Figure 6.2.

Our approach allows specifying not only the number of identified documents for each class but also the number of classified sentences since we compute similarity scores for each sentence of each document. This means that multiple sentences of the same document can have a high class threshold and be recognized as part of the same class. While the classified documents are the final product of the classification, the sentences themselves can be used in a further step to train the text classifier. We observe significant differences between the number of sentences and the number of classes for each model and threshold. For example, using the model *all-MiniLM-L6-v2* and the threshold of 0.6, the number of sentences for the class "tires" is almost twice the number of documents. The corresponding results are shown in Figure 6.3.

Since we have a rough estimate of the classified documents from the data set, we also want to compare the obtained number of classified documents with the numbers from the data set. For the model *all-MiniLM-L6-v2* with the class threshold of 0.6, we see that the obtained number of documents for the classes "airbag" and "tires" is very close to the baseline. This is also the case for the class "airbag" obtained with the model *all-distilroberta-v1*. Another case that occurs frequently is that the estimated number of documents is between the threshold of 0.6 and 0.7. This is the case, for example, for the class "seats" in *all-MiniLM-L6-v2* or the classes "engine" and "tires" in *all-distilroberta-v1*. The third case is that the estimated number

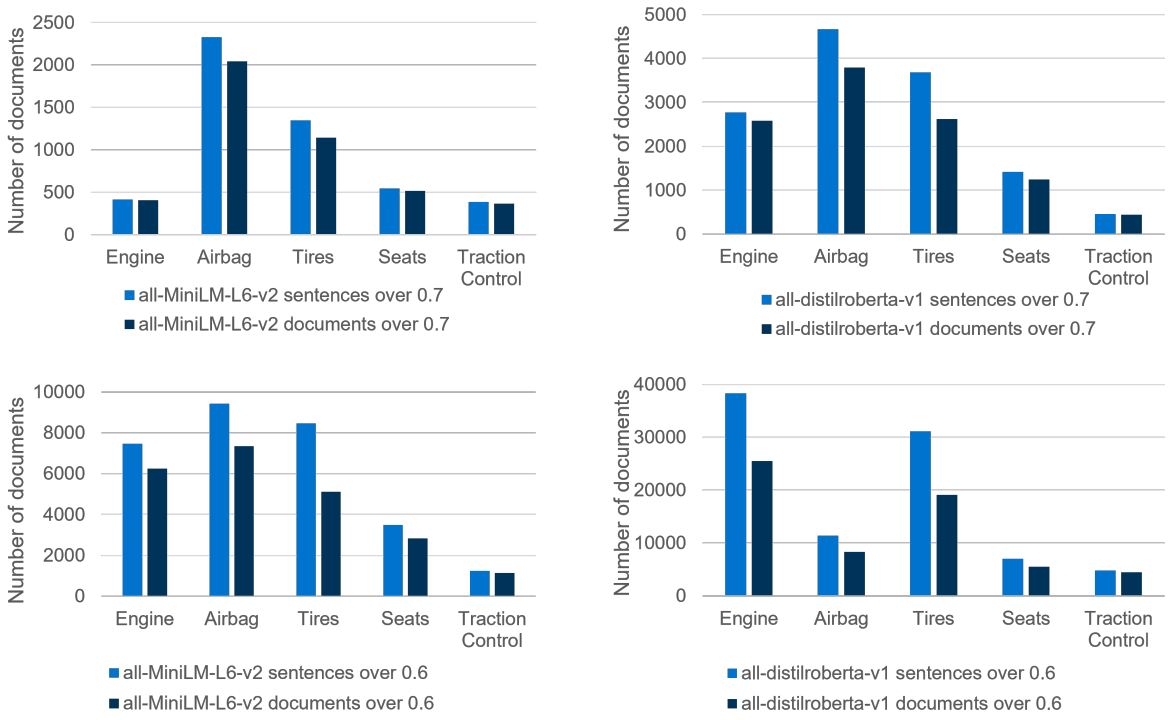
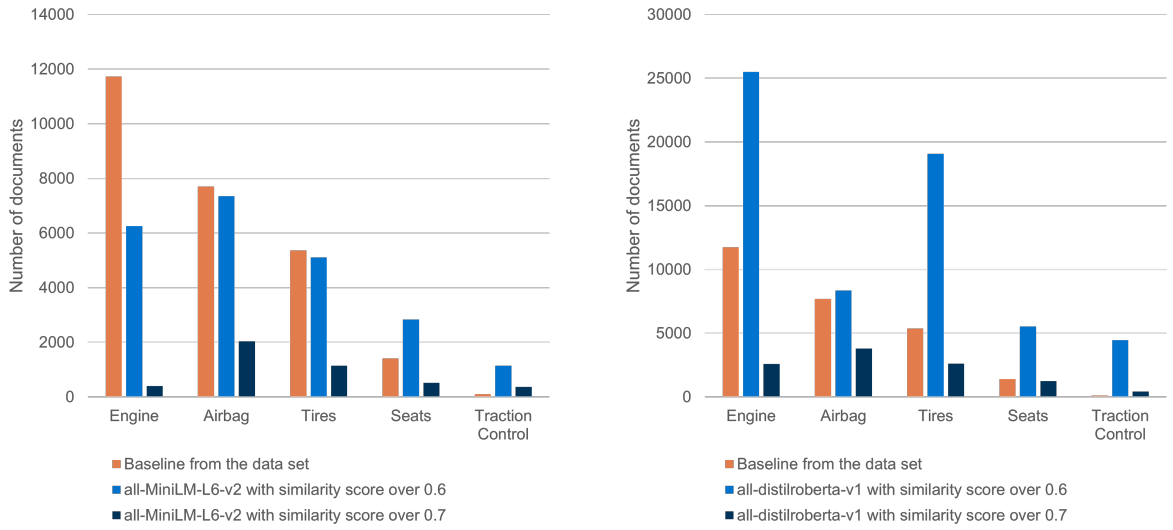


Figure 6.3.: Comparison of the number of sentences and documents obtained by two SBERT-based models for two thresholds.



(a) Model: all-MiniLM-L6-v2.

(b) Model: all-distilroberta-v1.

Figure 6.4.: The number of classified documents from the sample of 100,000 documents by model and similarity score threshold in comparison to the baseline.

of documents is much closer to the threshold of 0.7 than to 0.6. This is the case for the class "seats" with *all-distilroberta-v1*. The results with the number of classified documents for each class as a function of the class threshold are shown in Figure 6.4.

These considerations lead us to the idea that the proposed approach can be used for unsupervised classification without subsequent training of the text classifier. For this purpose, the calibration of the class threshold must be performed depending on the class and the embedding model used.

6.4. Validation Results

The validation process was performed on already classified 100,000 documents using the proposed approach and five validation classes representing vehicle components, which are "airbag", "engine", "seats", "tires", and "traction control". Each questionnaire contained 100 statements for each class, with half of the statements being true and the other half being false. Since each class was rated by three voters on a majority vote basis, the value of agreement among the voters, represented by Fleiss' Kappa, is given for all classes. Fleiss' Kappa values closer to 1 indicate high agreement among voters. In contrast, low or even negative values suggest disagreement between raters [83]. It is worth pointing out that no universal interpretation of Fleiss' Kappa exists and that it always depends on the number of voters and the underlying classes. We present the results in the form of classification reports for each validation class and embedding model and report the values for three metrics, which are precision, recall, and F1-score.

We select validation classes based on how easily their meaning can be understood and how similar they are semantically to other classes. The classes "airbag", "engine" and "tires" have self-explanatory titles and can be easily distinguished from other predefined classes. In contrast, the class "Seats" has a high semantic similarity to two target classes, which are "child seat" and "seat belt", making it more difficult to distinguish. The fifth class, "traction control," requires voters to have more experience in the domain to understand it. Our results suggest that self-explanatory classes with high semantic distinction from other classes achieve better classification results and higher voter agreement.

Four classification runs were required to validate the results. All computations were performed on a single laptop with 16 GB RAM and the Intel(R) Core(TM) i7-1065G7 processor with four cores, eight threads, a base frequency of 1.30 GHz, and a maximum turbo frequency of 3.90 GHz. We provide the time needed for the classification in Table 6.3.

Table 6.3.: Processing time required for the classification using the proposed approach.

Model	Number of documents	Processing time
all-MiniLM-L6-v2	100,000	9.1 hours
all-distilroberta-v1	100,000	13.4 hours
tok2vec	100,000	2.1 hours
all-MiniLM-L6-v2	75,445	7.2 hours

6.4.1. Validation Round 1

In the first round of validation, we use the SBERT-based *all-MiniLM-L6-v2* model and set the similarity score threshold to 0.7 for all validation classes. For validation, 50 true and 50 false statements are used for each class.

For the "airbag" class, 2042 documents are identified, and this class has the highest F1-scores of 0.93 for both true and false statements. These values are supported by a high Fleiss' Kappa of 0.83. The "engine" class of 405 documents also has high metrics for true and false statements, achieving F1 scores of 0.91 and 0.89, respectively. Although the results for the "seats" class are high enough, with 518 documents, they are the lowest in this round of validation. It achieves F1-scores of 0.76 and 0.83 for true and false predicted sentences, respectively, and a high Fleiss' Kappa score of 0.82. One of the reasons for this is that, as expected, the semantic similarity to two of the predefined classes makes it difficult for the validators to draw the line between these classes. Then, we also get solid results for the "tires" class with F1-scores of 0.88 and 0.9 for true and false statements, a high agreement rate of 0.82, and 1143 documents. The last class, "traction control", receives 367 documents and achieves impressive F1 values of 0.92 and 0.9 for the true and false documents and a moderate Fleiss' Kappa of 0.59.

The statistics on the number of classified documents can be found in Figure 6.2 and the results of the classification report in Figure 6.5.

Overall, the metrics of the classification reports, along with the Fleiss' Kappa values, suggest high-quality results for all classes in this round.

6.4.2. Validation Round 2

The second round was performed with the same classification results of the model *all-MiniLM-L6-v2* as in the first round. Again, 50 true and 50 sentences are used for each class. In this round, the similarity score threshold was lowered to 0.6, allowing more documents to be included in the validation classes.

The "airbag" class with its 7347 identified documents delivers strong results with its F1-score of 0.91 for both true and false data, and a significant inter-rater agreement of 0.79. For the "engine" class, 6256 documents are identified. Although the validation results for this class show substantial F1-scores of 0.7 and 0.67 for the true and false sentences, respectively, these results have weak support from Fleiss' Kappa of 0.21. At this point, it should be mentioned that the validation approach used in this work is sensitive to the assessments of the individual raters. This means that a low agreement between the raters either indicates irritating results or one or more raters had difficulties in understanding their task in the validation procedure. Next, the "seats" class obtains 2834 classified documents and moderate results of the classification report with the F1-scores of 0.59 for true and 0.76 for false data, and a fair agreement ratio of 0.38. Then, the results for the "tires" class of 5110 documents are supported by strong F1 scores of 0.89 for true and 0.91 for false sentences. It also has a substantial agreement rate of 0.63. Finally, the "traction control" class receives 1145 documents. Although the classification report results are more than convincing, with F1-scores of 0.91 for both true and false data, the Fleiss' Kappa results of 0.08 reveal a slight agreement among voters suggesting that the strong results may have arisen arbitrarily.

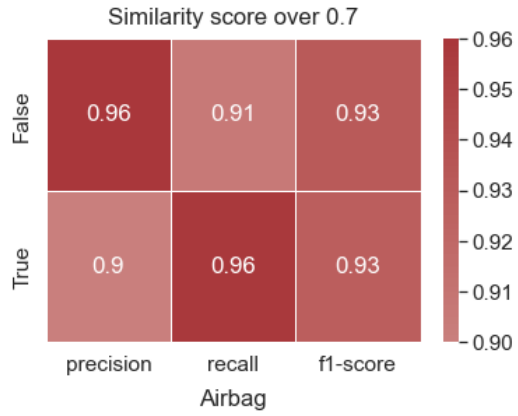
The number of documents for each class is given in Figure 6.2, and F1-score statistics and full classification reports can be found in Figure 6.6.

The results of this round of validation indicate that the "airbag" and "tires" classes received the documents with high metrics and a supportive rate of agreement among voters. The other three classes either have unconvincing results or are not supported by Fleiss' Kappa.

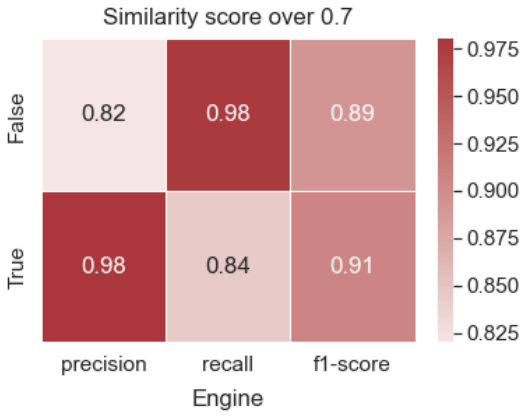
6.4.3. Validation Round 3

In this round, two embedding models are tested in parallel. These models are SBERT-based *all-distilroberta-v1* and *tok2vec*. For each of the models, 25 true and 25 false statements are given to voters for validation, so that each voter obtained 100 statements in total as in earlier rounds. The similarity score threshold is set to 0.7.

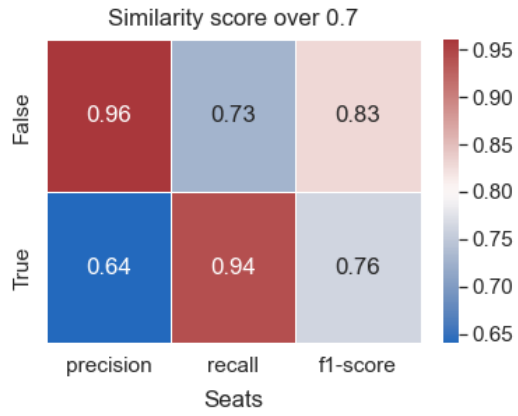
We start presenting the results for *all-distilroberta-v1* first. For the "airbag" class, 3793 documents are found, and it achieves maximum results reaching F1-scores of 1.0 for both true and false statements. These results are supported by a substantial inter-rater agreement score of 0.76. The "engine" class receives 2581 documents and performs well with an F1 score of 0.86 for both true and false statements. These results are supported by a moderate Fleiss' Kappa of 0.49. The "seats" class of 1244 identified documents achieves competitive F1-scores



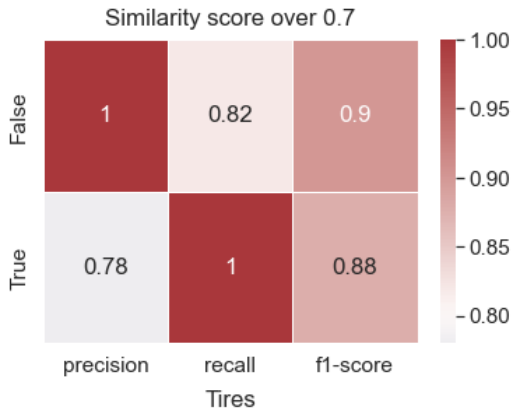
(a) Class: "airbag", Fleiss' Kappa: 0.83



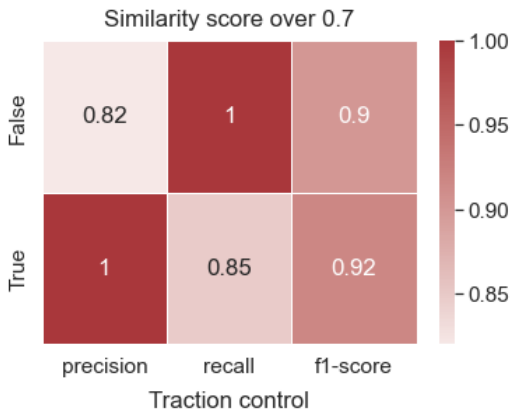
(b) Class: "engine"; Fleiss' Kappa: 0.7



(c) Class: "seats". Fleiss' Kappa: 0.82

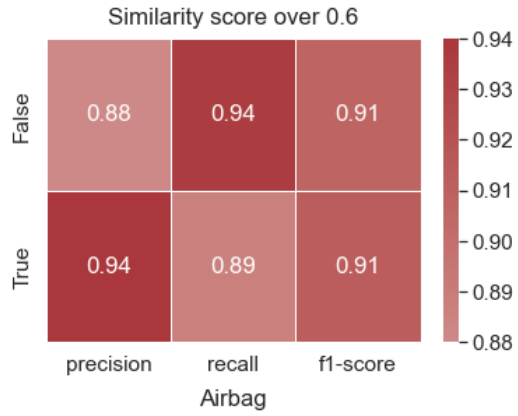


(d) Class: "tires". Fleiss' Kappa: 0.82

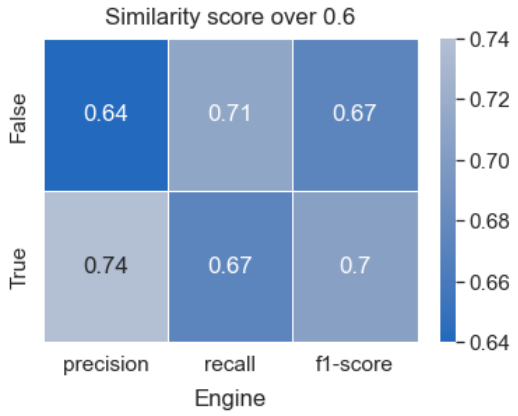


(e) Class: "traction control". Fleiss' Kappa: 0.59

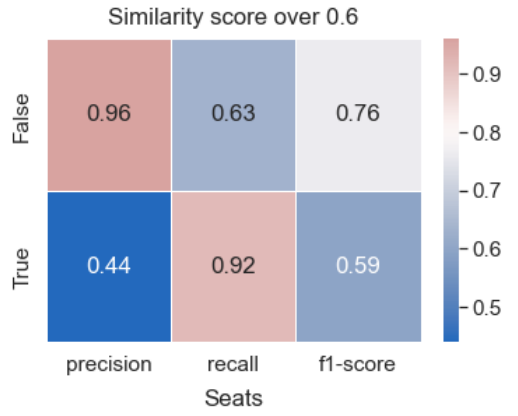
Figure 6.5.: Classification reports for the first validation round; model used: *all-MiniLM-L6-v2*.



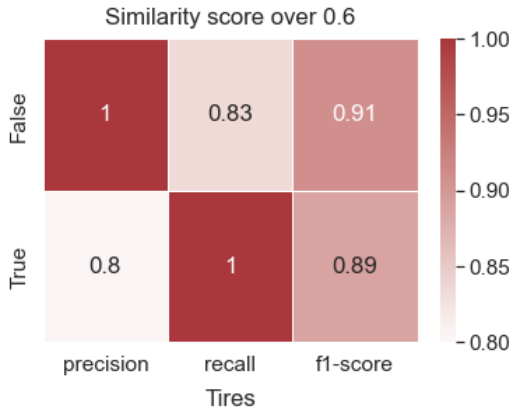
(a) Class: "airbag"; Fleiss' Kappa: 0.79



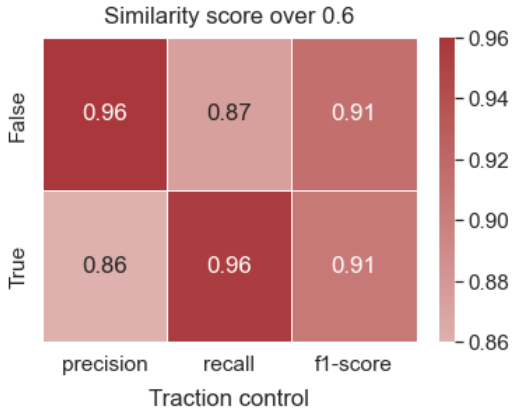
(b) Class: "engine"; Fleiss' Kappa: 0.21



(c) Class: "seats"; Fleiss' Kappa: 0.38



(d) Class: "tires"; Fleiss' Kappa: 0.63



(e) Class: "traction control"; Fleiss' Kappa: 0.08

Figure 6.6.: Classification reports for the second validation round; model used: *all-MiniLM-L6-v2*.

of 0.7 for true and 0.8 for false sentences, and a substantial agreement value of 0.7. For the "tires" class, 2620 documents are determined, and we observe high F1-scores of 0.89 for true and 0.91 for false statements. A reasonable Fleiss' Kappa of 0.65 is computed for this class. The final class, "traction control", receives 439 documents. Although it reveals almost perfect F1-scores of 0.89 for true and 0.91 for false sentences, its moderate rate of agreement of 0.44 makes the reliability of the results questionable.

The statistics containing the number of identified documents for each class are shown in Figure 6.2 and the classification reports can be found in Figure 6.7.

Summarizing the validation results for *all-distilroberta-v1*, we find that the "airbag", "seats", and "tires" classes have significantly high metrics supported by voter agreement, while the agreement rate for "engine" and "traction control" have moderate levels of voter uncertainty and require additional investigation of the results.

As for the validation results for the context-free *tok2vec* model, we report poor metrics for each of the validation classes. Both F1-scores and Fleiss' Kappa values are low for all classes and can be seen in Figure 6.8. Due to the low performance, the results of this model are not discussed in detail in this thesis.

6.4.4. Validation Round 4

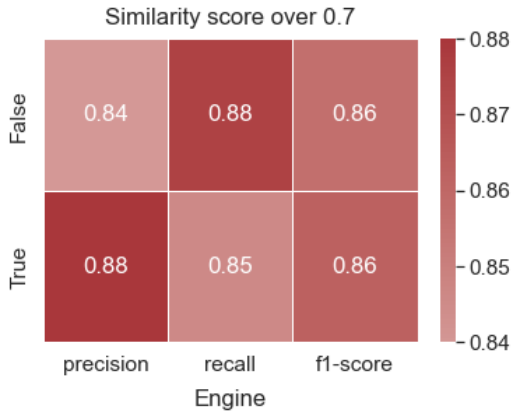
In the final validation round, we use 75,445 classified documents from the data set that are marked as "unknown", meaning that they have no associated vehicle component. We use *all-MiniLM-L6-v2* for classification and set the already proven similarity threshold of 0.7. Due to constraints on the availability of domain experts for this research, we report the validation results in this round only for the "airbag" class. We identify 769 documents and obtain high F1-scores of 0.98 for both true and false statements, supported by an almost perfect inter-reviewer agreement of 0.83.

This experiment was designed to simulate real-world conditions where all documents are unclassified and no baseline is available. Our results indicate that the proposed approach can be used in practice for classification tasks of unknown documents to achieve robust results.

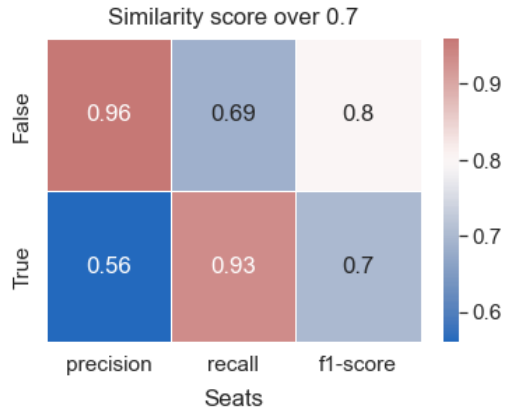
6.5. Results Summary

Previously, the results of four rounds of validation were described independently. In this section, we now compare the metrics obtained with different embedding models for the validation classes. A first look at the comparison results in Figure 6.10 shows that both SBERT-based embedding models provide strong results, while the results of the *tok2vec* model can be considered insufficient. We note that the results of the fourth round of validation cannot be directly compared to the other results because a different subset of data was used for this classification, but we include these results for completeness. In many cases, our approach shows convincing results, which are supported by the agreement between the voters. For example, we often see F1 values above 0.8 and even reach the maximum value of 1.0 for the "airbag" class. We also find that at higher similarity thresholds, the metrics are

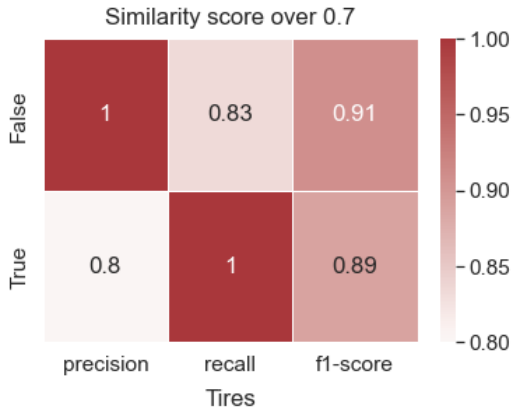
(a) Class "airbag"; Fleiss' Kappa: 0.76



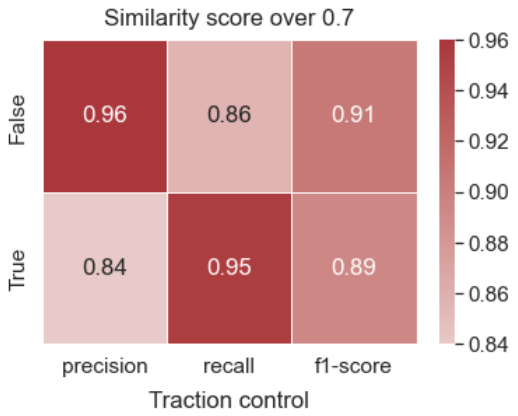
(b) Class: "engine"; Fleiss' Kappa: 0.49



(c) Class: "seats"; Fleiss' Kappa: 0.7

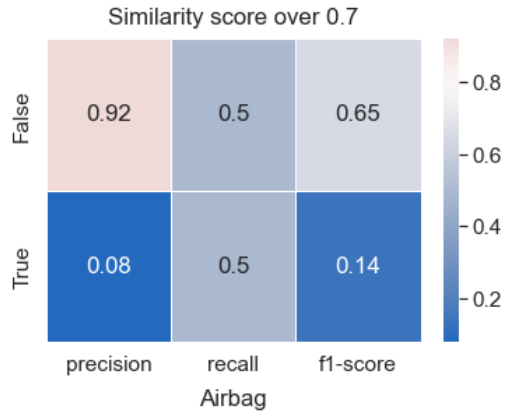


(d) Class: "tires"; Fleiss' Kappa: 0.65

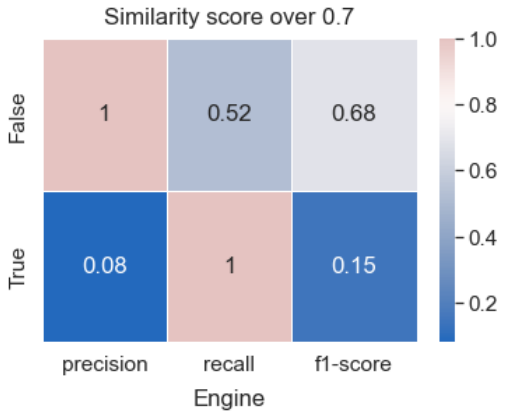


(e) Class: "traction control"; Fleiss' Kappa: 0.44

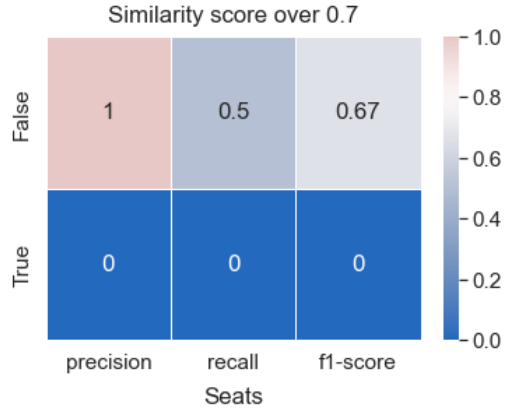
Figure 6.7.: Classification reports for the third validation round; model used: *all-distilroberta-v1*.



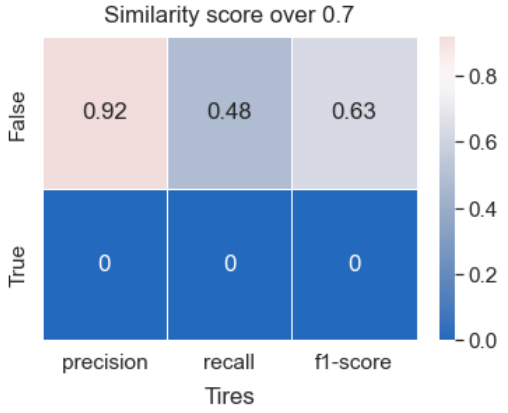
(a) Class: "airbag"; Fleiss' Kappa: 0.44



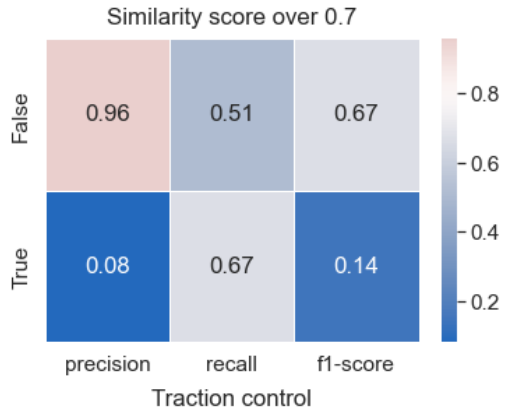
(b) Class: "engine"; Fleiss' Kappa: 0.24



(c) Class: "seats"; Fleiss' Kappa: -0.1

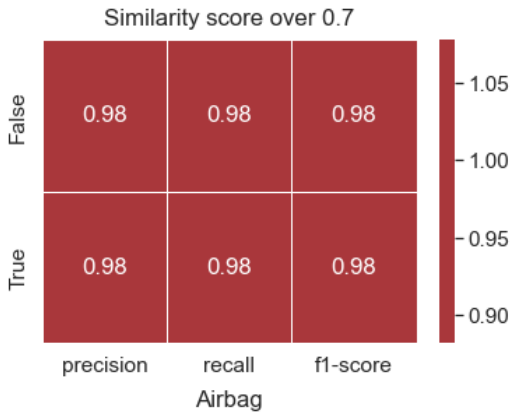


(d) Class: "tires"; Fleiss' Kappa: -0.1



(e) Class: "traction control", Fleiss' Kappa: 0.11

Figure 6.8.: Classification reports for the third validation round; model used: *tok2vec*.



(a) Class: "airbag"; Fleiss' Kappa: 0.83

Figure 6.9.: Classification report for the fourth validation round; model used: *all-MiniLM-L6-v2*.

either greater, as is the case for the "engine" class, or show higher inter-voter agreement, as shown by the results for the "traction control" class.

To identify the best model for each class, we combine the metrics from the classification report for true statements with the number of documents identified and present a full comparison of model outputs in Table 6.4. For the "airbag" class, the best model is *all-distilroberta-v1*, which scores highest on all three metrics and identifies the second-highest number of documents. The results are confirmed by a solid Fleiss' Kappa score. The best classification results for the "engine" class are obtained in the first validation round by *all-MiniLM-L6-v2*. Although it identifies a comparatively small number of documents, this is the only round in which the classification report results are supported by substantial agreement among the raters. Similarly, for the other classes, the best results are obtained in the first validation round with *all-MiniLM-L6-v2* and a similarity threshold above 0.7. For a complete overview of the results, see Table 6.4.

6.6. Comparison of Time Required for Manual Activities

One of the main motivations for this work was the fact that supervised learning methods used for classification require annotated data, which is scarce, and that manual label generation is time-consuming and expensive, especially in an industrial context. Our approach, therefore, aims to optimize the time spent on manual labor.

We further present the comparison of the time required between a common approach for obtaining training data through manual annotation and our proposed approach. For this purpose, we consider the same use case underlying this work, i.e., we have a data set of customer complaints and a domain expert who wants to classify the complaints into 25 different classes corresponding to vehicle components and containing problem descriptions. We show the computation of the required time for both cases.

Table 6.4.: Comparison of model outputs by validation class. The highlighted model has the best performance for the corresponding class.

Class	Validation round: Model	Identified documents	Precision	Recall	F1-score	Fleiss' Kappa
Airbag	1: all-MiniLM-L6-v2	2042	0.9	0.96	0.93	**0.83**
	2: all-MiniLM-L6-v2	7347	0.94	0.89	0.91	0.79
	3: all-distilroberta-v1	3793	**1.0**	**1.0**	**1.0**	0.76
	3: tok2vec	-	0.08	0.5	0.14	0.44
	4: all-MiniLM-L6-v2	769	0.98	0.98	0.98	0.83
Engine	1: all-MiniLM-L6-v2	405	**0.98**	0.84	**0.91**	**0.7**
	2: all-MiniLM-L6-v2	6256	0.74	0.67	0.7	0.21
	3: all-distilroberta-v1	2581	0.88	0.85	0.86	0.49
	3: tok2vec	-	0.08	**1.0**	0.15	0.24
Seats	1: all-MiniLM-L6-v2	518	**0.64**	**0.94**	**0.76**	**0.82**
	2: all-MiniLM-L6-v2	2834	0.44	0.92	0.59	0.38
	3: all-distilroberta-v1	1244	0.56	0.93	0.7	0.7
	3: tok2vec	-	0.0	0.0	0.0	-0.1
Tires	1: all-MiniLM-L6-v2	1143	0.78	**1.0**	0.88	**0.82**
	2: all-MiniLM-L6-v2	5110	**0.8**	**1.0**	0.89	0.63
	3: all-distilroberta-v1	2620	**0.8**	**1.0**	**0.98**	0.65
	3: tok2vec	-	0.0	0.0	0.0	-0.1
Traction control	1: all-MiniLM-L6-v2	367	**1.0**	0.85	**0.92**	**0.59**
	2: all-MiniLM-L6-v2	1145	0.86	**0.96**	0.91	0.08
	3: all-distilroberta-v1	439	0.84	0.95	0.89	0.44
	3: tok2vec	-	0.08	0.67	0.14	0.11

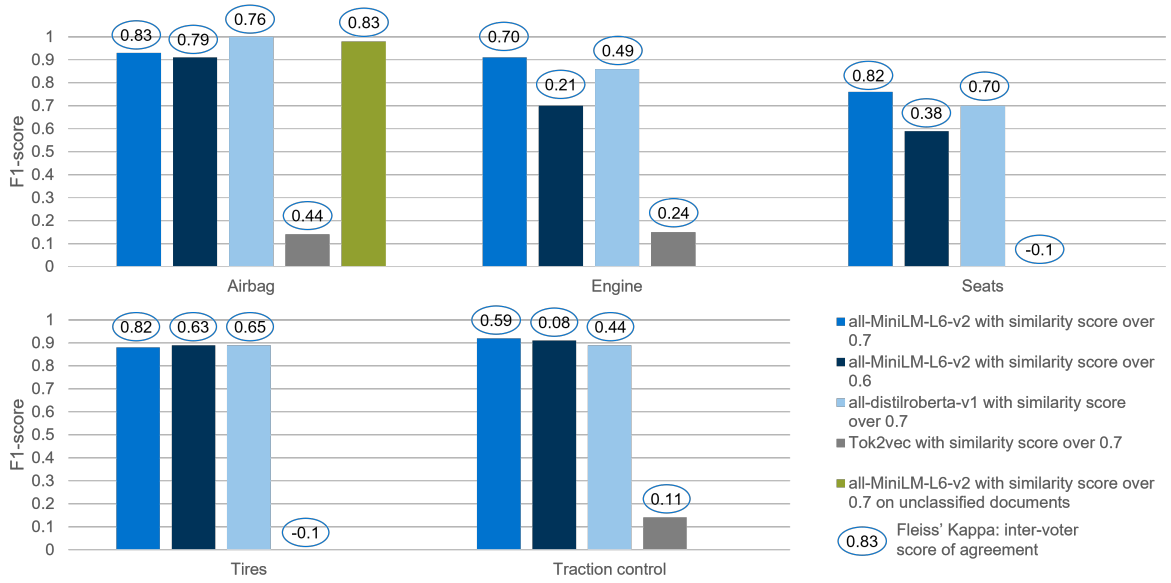


Figure 6.10.: F1-scores and corresponding values of the Fleiss' Kappa for each of the validation classes.

First, we describe the common approach of manual annotation. This method is based on reading and comprehending the individual complaints and assigning the documents to predefined classes. We assume that at least 100 documents per class are needed to train the text classifier [51]. This means that at least $25 * 100 = 2500$ annotated documents are needed for the underlying task. We also know from our data set that the average length of a complaint is about 100 words, so we assume that each document is that length. Then we need to know how fast a person can read and understand the text. For our calculations, we assume an average reading speed of 200 words per minute. This leads us to consider that the time required to read a single complaint is $100/200 = 0.5$ minute or 30 seconds. At this stage, however, a manual annotator has several problems. First, she must remember all 25 available classes when comprehending the complaint. Second, some complaints contain multiple topics, and in this case, it may be necessary to divide the document into several parts to ensure clear separation between topics. Third, a document may not contain any of the predefined topics and must be skipped. Fourth, complaints are received in a random order, and a uniform distribution of complaints across topics is unlikely, as is the case in our data set. These considerations show that extra time is needed to classify the complaints. For this reason, we assume that an additional 30 seconds are required for each complaint. Thus, we assume that one minute is needed to obtain each annotated document. This results in 2500 minutes of work, or over 41 hours, or about one work week, necessary to meet the minimum requirements for training the classifier.

Now that we have a benchmark for manual annotation, we proceed to describe the time required for our approach. It involves three steps where manual work is required. First, a class dictionary must be created that contains the descriptions for each class. Second, the

extracted context windows for each class must be manually evaluated to create context rules. Third, the results for each class must be validated. For the first step, our results described in section 6.4 indicate that two to five keywords may be sufficient to extract relevant context windows. We assume that it would take a domain expert five minutes to identify enough descriptive words for each class, resulting in $5 * 25 = 125$ minutes. The second step was performed by the author of this thesis, and the evaluation of 50 context windows presented as individual sentences required an average of eight minutes for each class. This results in a total effort of $25 * 8 = 200$ minutes for this step. The final step involves validation of the results, for which 50 sentences are again randomly extracted, adding $25 * 8 = 200$ minutes to the manual time required. In total, our approach requires $125 + 200 + 200 = 525$ minutes or 8.75 hours.

Thus, our approach requires 79% less time, reducing the effort from one work week to about one work day. The visualization of the required time can be found in Figure 6.11.

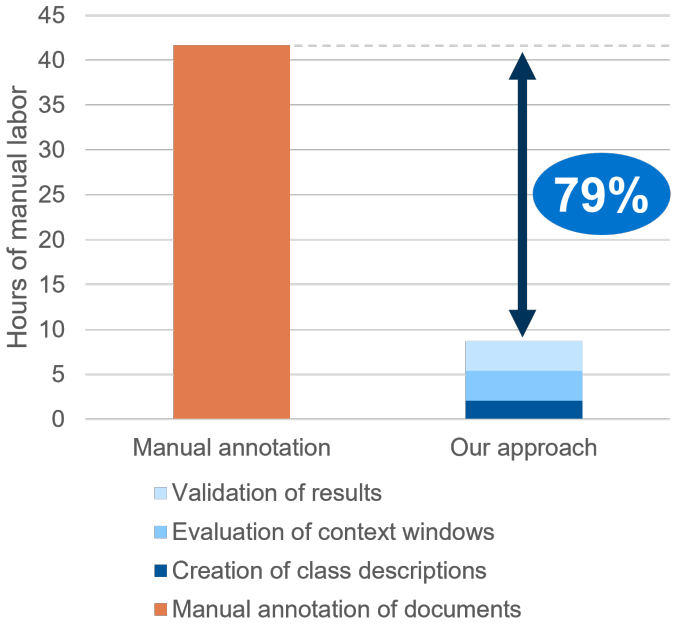


Figure 6.11.: Manual labor time comparison for creation of training data.

The considerable time saving is not the only result of this comparison. With reference to the first validation round, we would like to emphasize again the high quality of the obtained results for all classes, supported by a significant value for the inter-rater agreement. Comparing the number of documents for each class, we note that we expect 100 documents for each class in the manual annotation, while our approach yields 20 times more documents for "airbag", 4 times more for "engine", 5 times more for "seats", 11.5 times more for "tires" and 3.5 times more documents for the "traction control" class. A larger number of training documents should improve the process of training the classifier.

6.7. Validation Procedure in Practice

The proposed approach includes essential manual processing steps described in the previous section, as well as additional manual steps that include, for example, identification of target classes and calibration of the class threshold. The latter is described in this section. As we have shown above, the optimal number of identified documents depends on the class threshold with respect to the class specifics and the chosen embedding model. This fine-tuning is done by domain experts. We propose an algorithm for calibrating the similarity score threshold for each class, presented in Figure 6.12. The algorithm starts with the selection of an initial similarity threshold. We suggest starting with a lower threshold to get more classified documents. The next step is to validate a sample above the selected threshold. Depending on how satisfied one is with the results and the number of documents obtained, the threshold value should be either increased or decreased. The algorithm also suggests either training a text classifier or using the model as the final product for unsupervised text classification, depending on the expectation of the topics covered in the validation sample. We also consider a case where the threshold cannot be changed significantly without causing repetition, which implies that the problem should be looked for in other steps, e.g., documents for the expected class in the data set may be underrepresented.

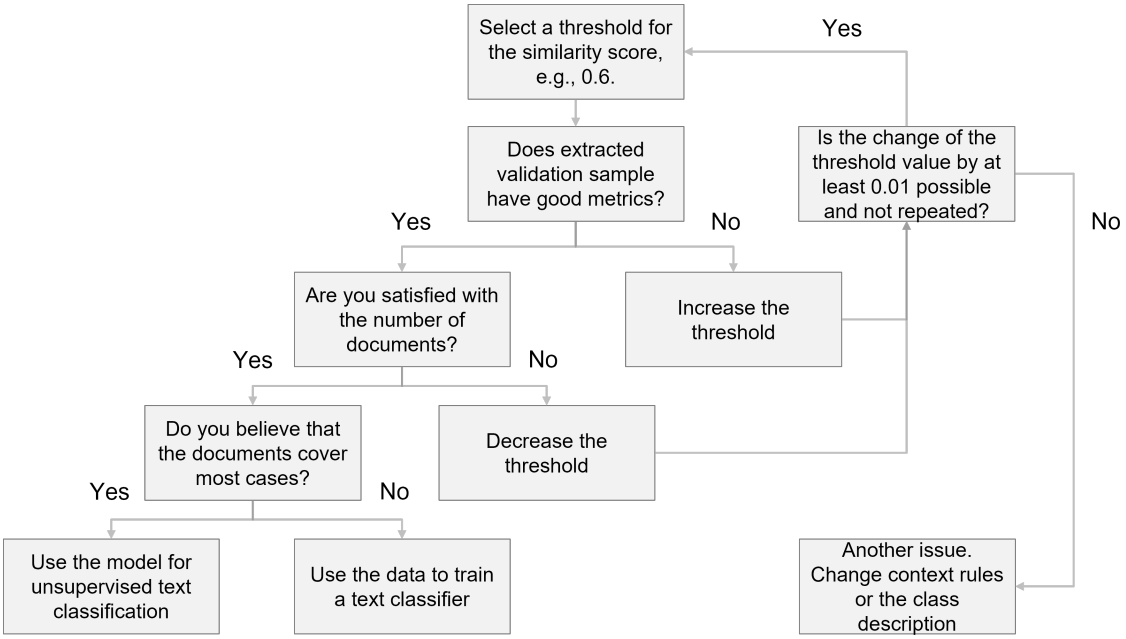


Figure 6.12.: The validation process in practice by fine-tuning the similarity score threshold for each class.

6.8. Topic Modeling and Clustering

This section presents the results of applying common unsupervised learning methods to the same subset of 100,000 documents used for the classification procedure. We evaluate the results from two perspectives. First, we assume that no domain expert is available and the unsupervised learning methods are able to classify the documents into specific classes. Second, we assume that a domain expert is available who would like to use our approach. We provide an assessment of whether these unsupervised methods can be helpful in facilitating manual activities.

For our purposes, we initialize an LDA and k-means clustering with 25 topics or clusters. For topic modeling, we use 15 keywords for each topic, while each cluster is described with 10 keywords. Then, we read through the results and give titles to the obtained topics or clusters. We see that the topics lack specificity, often do not reveal the context associated with the vehicle component problems, and their meaning is ambiguous. Therefore, these results cannot be used to classify the given documents. Some examples of the topic modeling and clustering results are given in Table 6.5 and Table 6.5, respectively.

Table 6.5.: Selected topics identified by LDA with assigned titles. The titles are assigned by the author of this thesis.

Topic number	Identified keywords	Given title
1	'warranty', 'sienna', '2009', 'new', '2007', 'safety', 'dealer', '2010', 'issue', 'bike', 'prius', 'camry', 'motorcycle', 'problem', 'toyota'	Vehicles
4	'wire', 'electrical', '552b6', 'foia', 'redacted', 'pursuant', 'freedom', 'headlight', 'act', 'xxx', 'harness', 'wiring', 'switch', 'information', 'honda'	Electric components
8	'flat', 'pressure', 'right', 'number', 'ak', 'new', 'replaced', 'firestone', 'dot', 'size', 'rear', 'tread', 'miles', 'tires', 'tire'	Tires
15	'dash', 'night', 'comes', 'time', 'come', 'headlights', 'times', 'safety', 'turn', 'driving', 'vehicle', 'problem', 'light', 'issue', 'lights'	Light
20	'fluid', 'automatic', 'problems', 'failed', 'dealer', 'gears', 'shifting', 'speed', 'problem', 'shift', 'replaced', 'vehicle', 'miles', 'gear', 'transmission'	Transmission

Since unsupervised learning methods are not able to capture specific topics required for our task, we can conclude that our approach prevails in this respect since domain expertise is added to the topic selection process.

Then, we want to evaluate the applicability of the results to facilitate the manual work of the domain expert that the proposed approach requires. Among the three manual steps, which are the creation of class descriptions, the evaluation of context windows, and the validation

Table 6.6.: Selected clusters identified by k-means clustering with assigned titles. The titles are assigned by the author of this thesis.

Cluster number	Identified keywords	Given title
2	contact, repair, manufacturer, recall, campaign, notification, parts, available, reasonable, received	Campaign
4	seat, belt, child, driver, belts, buckle, passenger, broke, safety, retract	Seat belt
8	car, driving, engine, start, problem, light, time, stop, drive, started	Engine
18	windshield, wipers, wiper, work, motor, intermittently, working, cracked, crack, visibility	Visibility
24	deploy, bags, air, did, vehicle, driver, impact, airbags, collision, deployed	Airbag

of the results, the results of topic modeling and clustering can be useful only for the first step, to identify the keywords for class descriptions. For this purpose, we read through the results and pick out the words that can be used to describe the predefined classes. The combined representation of the topic modeling and clustering results in terms of the target classes is given in Table 6.7. From this table, it can be noticed that the unsupervised methods can identify some relevant keywords for the description of the predefined classes.

We also identify keywords that can be helpful in describing task-specific contexts:

- Automotive context: vehicle, bike, motorcycle, car, driving, traffic, road, gm, dodge, truck, speed, mph, mileages
- Car manufacturers: vw, kia, honda, nissan, ford
- Daytime: night
- Issue description: noise, warranty, safety, issue, problem, failure, repair, recall, replace, fix, service, damage, accident, hit, leaking, leak, accident, cracked, rust, rusted, injuries, warning, broke

We conclude that the results of topic modeling and clustering cannot be used alone to classify documents into specific classes, but they can serve as a source of inspiration for creating class descriptions.

Table 6.7.: Potentially applicable keywords for the predefined classes determined by unsupervised learning methods.

Class	Keywords
Electrical system	plugs, plug, electrical, wire, wiring, harness, switch, battery, gauge, sensor
Airbags	airbag
Engine	cylinder, valve, motor, gasket, coolant, oil, ignition
Power train	clutch, transmission, shift, shifting
Steering	steering, column, turning, gear
Service brakes	braking, brake, brakes, rotor
Fuel/propulsion system	fuel, gas, pump
Tires	wheel, tire, tires, sidewall, firestone
Suspension	suspension, spring
Exterior lighting	headlight, headlights, light
Electronic stability control	abs
Seats	seats
Seat belts	buckle, seatbelt, belts
Visibility	window, windows, windshield, glass, shattered, wiper, wipers
Latches/locks/linkages	latch, locks, lock
Structure	door, doors, sunroof, frame
Equipment	tank
Child seat	child
Parking brake	park
Traction control system	sliding

7. Discussion

This chapter highlights the limitations of the proposed approach, discusses the identified challenges in detail and suggests directions for future research, evaluates the use of unsupervised learning methods, and summarizes the benefits of the proposed method.

7.1. Limitations

In this section, the limitations of our work and the suggested approach are presented:

- Domain expertise is required for the creation of class descriptions, evaluation of context windows, and validation of the results.
- A reasonable class threshold, which determines the number of documents belonging to the class, can only be identified experimentally.
- The selection of keywords is influenced by the person's past experience and therefore can be biased.
- The identification of documents in which the complaint and the vehicle component are described in different sentences is limited due to the design of the proposed approach. The next section suggests ways to overcome this limitation.
- In the validation procedure, we assume that all professional employees of the cooperating company are domain experts, but we cannot guarantee this. For this reason, we deliberately choose self-explanatory classes for the validation procedure to reduce bias.
- Our approach is intended to be used for topics that can be predicted in advance. In contrast, it is not supposed to be used for the identification of unknown or new topics.

7.2. Future Work

This section discusses identified challenges during the classification process summarized in section 6.2.
C1: A deliberate choice of text preprocessing techniques often depends on the content of the underlying data set, its structure, and its size. To achieve time-efficient results, the techniques must be applied in a meaningful sequence [77]. We suggest investigating the application of common preprocessing approaches:

- Stop word removal is a common text processing technique to eliminate the words that are unlikely to convey much information [77, 84]. They include articles such as "a" and "the", function words such as "and" and "he", and domain-specific words. Numerous stop word removal lists and algorithms exist [85]. Using the list of stop words provided by NLTK [86] or spaCy[1] would be a good start. However, the application of this method requires the preservation of the original documents, since reading a text with missing words can be difficult for a human. We recommend creating an additional column with the removed stop words and using it for computer processing while providing the human-readable texts for manual evaluation.

- Removing numbers is used to eliminate the numbers that do not carry useful information [77]. For example, data set entries may contain local identifiers that are not very informative. However, in other cases, numbers such as collision speed or paragraphs in the context of legal documentation are important for context understanding.

- Stemming is another standard text preprocessing technique often used to reduce the vocabulary and facilitate text processing [77, 84]. As with the stop words removal, the original documents must be preserved for further manual processing.

- N-gram inclusion can also be considered to treat the words not as single units, but as word combinations [77].

- Infrequently used terms can also be removed to reduce the vocabulary [77].

C2: Maintaining readability is necessary for the manually performed steps, i.e., the evaluation of the context windows and the validation of the results. For this reason, we propose to maintain two versions of the documents for classification: one version can be left in its original form and used for the manual steps, while the other version includes the necessary text processing steps such as lower-casing and noise removal and is processed by the computer. In this case, one of the challenges is to correctly retrieve the context windows from the original documents based on the preprocessed version, especially if a fixed context window size is used.
C3: The pre-defined classes should not be collectively exhaustive, but should only cover the topics that interest us. Our approach helps classify the documents that have similar semantics to the pre-defined classes while accepting that documents containing information outside our scope of interest will not be classified. To explore the topics of unclassified documents, unsupervised methods such as topic modeling can be used.
C4: When describing the class, a balance must be achieved between the number of keywords and the amount of time required. In our experiments, many words included in the class descriptions do not appear in the final context rules, which means that rare words are less likely to be identified as part of context windows. For domain experts, we recommend taking 3-7 minutes to describe each class, starting with 3-5 keywords. Further research can determine if increasing the number of keywords improves the explainability of the class.

[1] https://spacy.io/

C5: The availability of domain experts to produce meaningful class descriptions is one of the prerequisites of our approach. In practice, however, access to expertise may be limited, and employing these individuals can be costly. Once access to domain experts is given, they need to choose precise keywords for class descriptions. However, the definition of "precise keywords" is ambiguous. The keyword generation process can be further improved by defining what is precise and specific and what supporting tools such as Wikipedia, Google Images, or topic modeling can facilitate this process.
C6: The proposed approach suggests that the classes can be structured in a hierarchical way. However, in our thesis, we do not directly investigate the application of the proposed method to the classes with this structure. This gap should be addressed in future research. We propose to explore two options for creating hierarchical dependencies:

- The first option is that a child class shares some or all of the keywords of a parent class and receives additional specific keywords to distinguish its features. In this case, more non-specific and frequent keywords of the parent class are used and the specific words are overweighted.
- The second option assumes that a parent class and a child class are described by mutually exclusive keywords. This means that some keywords of the parent class can be excluded in favor of the newly created class.

C7: Both adding new classes and editing the definition of existing classes is easy with our approach. For this purpose, either a new class must be added to the class dictionary or the keywords of the existing classes must be changed. Based on a new definition, new context windows, context rules, and class vectors can be created. The resulting class vectors are then used to classify documents in the newly defined classes.
C8: Appropriate description of classes requires either time investment to become familiar with the problem and the data set, or the availability of industry experts who know their target audience and are familiar with the commonly used words in a data set. This is necessary because the process of extracting context windows is highly dependent on the exact string match of keywords and words in the data set. For this reason, it is important to be familiar with linguistic features such as formal or informal vocabulary.
C9: Acronyms are often used to describe several terms, e.g. ABS is commonly used instead of an anti-lock braking system. However, the embeddings of the acronyms are ambiguous and can confuse the model [87]. Further research should investigate methods to obtain contextually correct definitions of acronyms, such as those proposed by Charbonnier and Wartena [87]
C10: In our setting, we allow the use of word combinations as keywords for class descriptions. In some cases, e.g. when the keywords used are rare and not enough context windows can be identified, the word combinations can be split into separate words and then used as separate keywords for identifying context windows in a new search.
C11: In our thesis, we use single sentences containing keywords to represent context windows. Further research should investigate different representation options for the context windows:

- First, the context window may contain several consecutive sentences, since in our case it is common for people to describe the vehicle part in one sentence and continue with the problem description in the following sentences.
- Second, the context window can contain all the sentences in which the class keywords appear. For example, the word "engine" can be used in the first sentence and in the fifth sentence. In this case, the concatenation of these two sentences can be considered as the context window.
- Third, a context window can be represented using tokens rather than sentences. A simple approach to determine the length of a context window is using a fixed number of tokens to the left and right of the keyword. However, our experiments have demonstrated that this approach often yields ambiguous context windows, as they often contain the end of one sentence and the beginning of the following sentence, which presents an unclear context for manual evaluation.
- Fourth, methods for representing context windows using a dynamic number of tokens on either side of the keyword can be explored.

C12: The ambiguity of context windows arises from the fact that they capture only a small portion of the document and often do not reveal the full context of the document. For example, in our early experiments with representing context windows by a fixed number of tokens surrounding the keyword, they often contained truncated portions of multiple sentences, making them difficult for a human evaluator to understand. Using single sentences as context windows helps to exclude truncated sentences, but does not completely solve the problem for two reasons:

- First, the vehicle component and the problem itself are often described in different parts of the document. This means that our context window often consists only of a component description and does not include the problem itself. For example, the sentence "My engine was repaired within a week" assumes that a person had a problem with their engine, but does not address the problem.
- Second, due to the informal tone of the documents in the data set, many people do not use punctuation correctly, resulting in documents consisting of only one sentence with no logical separators. A context window includes in this case a lot of redundant information. We propose to consider context windows with a limited length to facilitate the evaluation. In our thesis, we set the maximum length of the context window to 20 tokens.

Context windows with ambiguous meanings were discarded during the evaluation phase. Further research should investigate how to reduce the number of ambiguous context windows. Defining methods that allow extraction of context windows surrounding both the vehicle component and the problem should lead to higher quality context windows.
C13: The number of context rules is a hyperparameter that can vary depending on the class.

In this thesis, we use a different number of context rules for each class. Further research can investigate whether a minimum or an optimal number of context rules can be identified, on which class features this number depends, and whether the dependency between the quality of the class vector and an additional context rule can be quantified.
C14: The number of different keywords appearing in context rules is a hyperparameter that can vary by class. We used a considerable number of keywords for each of the classes, but the added value of each additional keyword is not clear, since only a few keywords from the dictionary actually appeared in the context windows and, further, in the context rules that were de facto used for the calculation of the class vectors. We propose to investigate if the ratio between the number of keywords in a class description and the number of keywords used for context rules influences the results.
C15: A manual evaluation of the context windows to consider them as context rules is one of the essential steps of the proposed approach. However, it requires a considerable amount of time. In our case, it took the authors of this thesis six to nine minutes to the evaluation of 50 context windows per class. In a multi-class classification with several dozens or even hundreds of classes, this step can become a significant bottleneck. Further research should explore the possibility of reducing the time required by using fewer or shorter context windows or eliminating this step entirely.
C16: This challenge refers to the fact that context rules can be obtained in a different way than is the case in this thesis. We provide some thought-provoking ideas for further research on the identification of context rules:

- A simple approach to determine context rules without prior extraction of context windows can be performed by artificially creating the context rules from scratch, e.g. for the class "engine" context rules like "engine damage", "ignition problem" and "rough idle" can be used to determine the class vector.

- Another idea is to use extracted context windows as inspiration and manually correct them to meet the requirements of the task. For example, the sentence "the interior lighting worked well" can be manually replaced with "the interior lighting did not work well". This method allows the rejection of fewer context windows with only minor changes.

- A set of universal, class-independent word combinations can be created, which are then combined with the classes. For example, word combinations such as "... stopped working" and "problem with ..." can be used in combination with the class keywords to obtain context rules in the form "engine stopped working" and "problem with interior lighting".

- The last suggestion leads us to consider that a dictionary can be created not only for classes but also to capture task-related semantics. In our thesis, we identify issues related to vehicle components, so an additional dictionary component can be created to contain the issue-related keywords. It can contain the following keywords: failure, issue, problem, broke, malfunction, etc. These descriptions can be used as additional

conditions for context window identification. In this case, a keyword from the class and problem description must be found in the context of the document.

C17: Many pre-trained models are available for text embedding. In this thesis, we experiment with the embeddings of the context-free tok2vec model and two contextualized models based on the SBERT model. We show that contextualized embeddings are superior for our task and can provide high-quality results with reasonable time consumption. Further research can explore the application of other state-of-the-art models. We propose to explore sentence similarity libraries provided by Huggingface[2] and consider recent technological developments such as Token Attention Sentence-BERT [88], which addresses the problem of giving equal weight to all words in a sentence, and SimCSE [89], which represents the state-of-the-art for sentence embeddings at the time of writing this thesis.
C18: In our thesis we do not train a new model, but use pre-trained models, because the additional training step would go beyond the scope of our thesis. However, there are several ways to train an individual model, e.g. SBERT[3] can be used for this purpose.
C19: We use an intuitive average-pooling method to compute class vectors, analogous to the weighting of individual words in a sentence used for training SBERT. Other approaches such as min- or max-pooling can be explored, but the intuition behind them does not seem promising to us.
C20: High similarity scores for pairs of class vectors may indicate that they describe the same topic and their context rules are very similar. In this case, the same documents may be assigned multiple times to the classes with different titles. In this case, it is recommended that the context rules of these classes be investigated and possibly modified to achieve the expected differentiation. Further research should investigate the maximum acceptable similarity between class vectors.
C21: In contrast to aggregation of context rule embeddings, more sophisticated approaches based on supervised learning methods such as K-Nearest Neighbors (KNN) [90] and Support Vector Machine (SVM) [91] can be applied to context rule embeddings in further research. For example, KNN can be applied to document embeddings in the classification phase to evaluate whether they belong to one of the classes represented through multiple embeddings of context rules.
C22: In our thesis, we use individual sentences of documents to calculate the similarity score and classify the underlying document. The problem we encounter is that the vehicle component and the topic can be described in different parts of the document and single sentences are not representative enough to cover both aspects. The same is true for identifying multiple topics in one document, which is one of the essential problems for a given task. Similar to the context window extraction approaches discussed above, other techniques can be tested in further research to identify text segments of a document with the highest similarity to the target class:

- Multiple sentences or n-grams of a document can be used as a basis for comparison

[2]https://huggingface.co/tasks/sentence-similarity
[3]https://www.sbert.net/docs/training/overview.html

with the class vectors. For example, two sentences with the highest similarity score can be concatenated, and the value can be recalculated for that sentence combination.

- For the creation of training data, only documents that contain the keywords from the class dictionary can be used for similarity detection. Then, the similarity scores are computed only for sentences or n-grams of these documents. However, this approach is not recommended for pure unsupervised text classification, since we also want to capture the documents that contain words outside the predefined class descriptions.
- A sliding window can be used to identify customer complaints that relate to specific vehicle components. In this case, either a static or a dynamic size of the context window can be considered. A conventional fixed-size sliding window [37] containing n tokens is iteratively shifted from left to right starting from the first word, and for each iteration, the similarity score is computed, considering the window with the highest similarity score as the class representative. In contrast, a dynamic sliding window can determine the initial position of the text segment of interest [92].
- A combination of sentence similarity and *Question Answering* (QA)[4] can be further explored. In this case, a high similarity score from the similarity step is considered as a signal for a QA model to iterate over the document and determine the passage that can describe the problem better than a single sentence. It works as follows: when a sentence with a high class threshold is identified, the QA model takes the entire document as context and identifies the most appropriate passage that describes the problem with a targeted vehicle component.

C23: Although the proposed approach is not very time-consuming and can be performed even with a single laptop, we present some ways to further improve the algorithm:

- Tokenization can be performed once for the entire corpus instead of tokenizing the sentences at each iteration.
- An iterative *itertuples* method can be replaced by faster vector calculations.
- Writing some code in Cython[5] can be considered.
- Parallelization of processes with multiple workers can be considered.
- The similarity scores can be calculated at a time for each sentence using matrix multiplication instead of calculating the similarity score for each class vector separately.
- The data processing can be performed in the cloud.

C24: A minimum threshold is a hyperparameter that can be adjusted depending on the underlying task. It has no direct influence on the results but allows for a reduction in the amount of redundant data. For our experiments, we only stored the data with a similarity

[4]https://huggingface.co/tasks/question-answering
[5]https://cython.org/

score above 0.5. In the future, this value can be better specified depending on the underlying task.
C25: In this thesis, we use the same class threshold for all classes and perform validation for values above 0.6 and 0.7. Our experiments show that the number of classified documents is very sensitive to changes in the class threshold. It means that the calibration of the class threshold is one of the required procedures that must be conducted by domain experts who can assess if the obtained validation metrics are sufficient for the underlying task. Further research can investigate the procedure for adjusting the class threshold, which can include suggestions for a starting class threshold and how much to change it depending on the number of identified documents and metrics of the classification report.
C26: Our thesis is primarily concerned with the characteristics of the complaint and the component description in the data set. Other features such as the car make or the source of the complaint could potentially be taken into account to improve the model.
C27: To solve the problem of recognizing multiple topics from the same document, we divide each document into sentences and compute the sentence-class similarities for each sentence of each document, instead of taking the document embeddings for the calculation of similarities scores as is commonly used in other researches [93]. The suggested approach is capable of identifying multiple topics based on the semantics of individual sentences of the document.
C28: In our case, voters are given 50 true and 50 false statements. The false statements are taken from the classes that are similar to the target class. An additional step can be introduced to examine the similarity scores of the false sentences with the target class to prove that they do not belong to this validation class.
C29: At this stage, we assume that the results for each class must be validated by an expert, and we describe the validation procedure for five classes. However, this stage is comparatively time-consuming, and we encourage the investigation of methods that reduce the time required. For validation, we use either 25 or 50 true statements, and measurable results are obtained in both cases. Future research may specify a minimum number of statements for a class so that the results can be verified. Another idea is to study the hierarchical structure of classes and explore whether good results for a lower-level class can causally mean that the results for the less specific higher-level class are also good and that validation for that class is not required.
C30: We use precision, recall, and F1 score to validate our results. Further research can evaluate the results using other metrics for quantifiable results, describe the significance of each metric, and elaborate on the meaning of "good" or "acceptable" results.
C31: The validation procedure can be improved in further research. First, using the same approach with three voters per class, an additional round of discussion can be organized after the results are reported to explore the reasons for the voters' choices. This is particularly useful when agreement among voters is low. Second, additional raters for each validation class can be included in the validation process to reduce bias. Third, other assessment methods can be used in place of majority voting. Fourth, the interpretation of Fleiss's Kappa depends on the number of voters and classes and should therefore be treated with caution.
C32: In our experiments, we have obtained strong classification report results using the proposed approach. The core idea of our method is to identify those parts of the documents

that describe both the vehicle component and the corresponding problem. Considering that we use a data set of customer complaints from the automotive industry, it is expected that the documents describing a problem with the vehicle prevail. For this reason, we remain critical of our method's ability to capture the differences in sentiment well and extrapolate to the tasks where both positive and negative reviews are present. This gap should be addressed in further research.
C33: The proposed approach requires a human's attention to the essential and additional steps. Essential steps include creating class descriptions, evaluating context windows, and validating results. Additional steps consist of configuring several hyperparameters such as the class threshold. We present the estimation of the required time for essential steps in section 6.6. Further research can suggest approaches to reduce the time spent in each of the manually performed steps.
C34: In classification, documents are assigned to a particular class based on a high similarity score of one of the sentences contained in the document. In this way, both sentences and documents can potentially be tested as a basis for training classifiers in further research. On the one hand, we obtain more sentences than documents and thus more training data. On the other hand, it is questionable whether the sentences alone contain enough information for the classifier to correctly classify longer documents.
C35: We believe that optimization in processing steps and suggestions for the hyperparameters can improve the performance of our approach and reduce the time required both for manual activities and classification. A non-exhaustive list of potential improvements is summarised as follows:

- Selecting preprocessing techniques and applying them in a sequence that ensures time efficiency (C1).
- Defining an optimal number of keywords per class (C4).
- Choosing the length and representation of context windows (C11).
- Determining the optimal number of context rules to describe the class (C13)
- Reducing time for a manual evaluation of context windows (C15).
- Exploring alternative methods for obtaining context rules (C16).
- Exploring other approaches beyond computing the class vector (C21).
- Selecting the approach to classify documents (C22).
- Adjusting a class threshold for each individual class (C25).
- Establishing the number of statements required for validation (C29).
- Reducing labor time required for the suggested approach (C33).

7.3. Categorization of Identified Challenges

In this section, we structure our findings with respect to the methodology steps introduced in section 5.3 and identified challenges summarized in section 6.2 to further elaborate on the answer to the first research question. We follow Li et al. [10] in describing the challenges from the data, model, and performance perspectives. Since our approach relies heavily on input from domain experts, we also add a fourth category, the expert perspective. The data perspective describes the challenges associated with the available text corpora and includes two challenges. The model perspective, which includes eleven challenges, discusses the use of different NLP approaches and their combination. The expert perspective, which includes fourteen challenges, discusses the steps performed by domain experts. Finally, the performance perspective addresses eight challenges with potential algorithm improvements. An overview of the categorized challenges can be found in Figure 7.1.

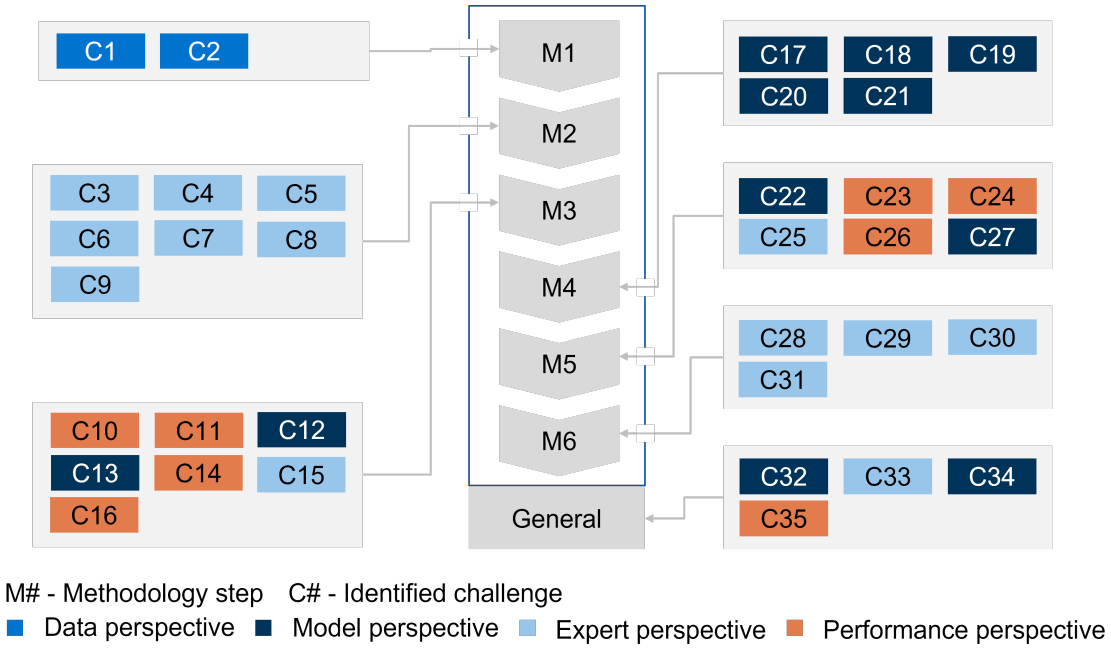


Figure 7.1.: Representation of identified challenges during the classification process using the proposed approach in four categories.

7.4. Application of Unsupervised Learning Methods

Previously, we demonstrated that common unsupervised learning methods such as clustering or topic modeling cannot be used alone to classify documents into predefined classes. However, they can be used as a complement to the proposed approach in the exploration phase to identify keywords either for class descriptions or for describing other task-specific semantics. It is worth mentioning that the use of these techniques is not necessary to obtain

representative results with our approach, but they can be used in combination with other supporting methods. We share some ideas on how to improve exploration with unsupervised learning techniques in the future:

- General and custom stop words can be removed from the data set.
- Lemmatisation or stemming can be applied to the data.
- A different number of topics or clusters can be tested to explore different keywords.
- The state-of-the-art approaches such as BERTopic [94] can be tested to obtain better results.

7.5. Advantages of the Proposed Approach

In this section we give a brief overview of the advantages of the suggested approach:

- As our calculations show, up to 79% of the manual labor time can be saved using our approach in comparison to the manual annotation while increasing the number of labeled documents.
- A document can be assigned to multiple classes.
- Both classified documents and sentences can potentially be used to train the classifier in the subsequent step.
- The proposed approach is robust to the changing requirements. Additional classes can be flexibly added, removed, or redefined.
- Our approach allows using hierarchical dependencies between classes.
- In this thesis we show that the proposed method has potential to be employed independently from the classifier training as a tool for unsupervised text classification.

8. Conclusion

In this exploratory work on obtaining training data from unlabeled text corpora, we investigated an approach that combines various NLP techniques with expert knowledge to classify documents into several pre-defined classes. The motivation behind this work was to help industry experts perform classification tasks in an environment with domain-specific target classes, missing annotated data, and frequently changing requirements.

The suggested approach relied on the use of pre-trained NLP models and the computation of cosine similarity between document and class embeddings. We conducted experiments with context-free and contextual transformer-based embedding models. The former is used as a benchmark and the latter to obtain state-of-the-art results. As a basis for our computations, we used an open-source data set of consumer complaints provided by NHTSA, which contains over 1.3 million samples collected over the last three decades.

Instead of time-consuming and costly manual annotation of documents, we used domain expertise to describe classes based only on their titles with some keywords. These class descriptions were then used to obtain context windows, which were then evaluated to become context rules and used to compute class vectors. For each pair of a class vector on one side and the embedding of each sentence of each document on the other side, the similarity score was computed. Depending on the similarity score and, more importantly, the class threshold, the documents were assigned to either one, several, or none of the target classes. Result validation was then performed with the cooperating company's domain experts, classification reports for each class were generated, and agreement between raters was measured.

Our results show that contextually enriched SBERT-based models significantly outperform a context-free *tok2vec* model. We present validation results for four rounds of validation with three embedding models, five validation classes, and two similarity thresholds. In our experiments, the SBERT-based models achieved the F1 score up to the maximum value of 1.0, supported by significant inter-rater agreement, while the classification results of the *tok2vec* model can be described as insufficient.

The proposed approach not only provides convincing classification results, but also solves other problems encountered in the industry, which are described in the introductory chapter. First, our calculations show that the time required to obtain training data for our task can be reduced by 79% compared to the manual annotation method. Second, due to the fact that target classes can be easily removed, added, or redefined, our approach is robust to frequent changes in requirements. Third, the developed approach allows industry experts to define only the classes of interest, thus filtering out irrelevant information. Fourth, this method is able to identify not only typical class features such as vehicle components, but also other semantic features such as negative sentiments. Fifth, the proposed approach can identify multiple topics in the same document.

We would like to additionally highlight the novelty of the proposed approach. First, instead of traditional manual data annotation, we relied on the expertise of people who are knowledgeable in their domain, understand the underlying task, and can validate the results. Second, we introduced concepts such as context windows and context rules and gave definitions to them. Third, we developed a framework that generalizes to classification tasks in other industries. Fourth, we showed that our approach, although intended only for identifying training data, has significant potential to be used as a tool for unsupervised text classification independent of the subsequent training of a text classifier.

An important goal of this work was also to identify challenges in the algorithm pipeline. These are intended to provide thought-provoking ideas on the development potential and to encourage further research. We identified the limitations of the proposed approach and formulated 35 challenges divided into four categories.

We also explored the application of topic modeling and clustering for the classification of automotive complaints. We showed that these approaches are not able to identify specific classes needed for our task, but they can be used for exploratory purposes to get an overview of commonly used keywords in the data set. Some of these keywords can be used by industry experts to create class descriptions.

In conclusion, we believe that further investigation of the proposed approach can achieve improvements in the classification process and allow this method to be used in a completely unsupervised manner to obtain a labeled data set without additional training of a supervised text classifier.

A. General Addenda

Table A.1.: Class codes used for classification

Class code	Class name
0	airbags
1	back over prevention
2	child seat
3	electrical system
4	electronic stability control
5	engine
6	equipment
7	exterior lighting
8	forward collision avoidance
9	fuel propulsion system
10	interior lighting
11	lane departure
12	latches locks linkages
13	parking brake
14	power train
15	seats
16	seat belts
17	service brakes
18	steering
19	structure
20	suspension
21	tiers
22	traction control system
23	vehicle speed control
24	visibility

```
master_components_dictionary = {
'electrical system': ['electrical', 'electricity', 'energy', 'cablecord', 'body control', 'seat heater', 'outlet', 'jack', 'port', 'usb', 'fuel level sensor', 'hill descent', 'driver assistance', 'fuel gauge', 'hud', 'display', 'odometer', 'chime', 'parking assist', 'park assist', 'starter', 'relay', 'hand heater', 'hill start', 'instrument panel', 'horn', 'fuse', 'circuit', 'driver monitoring', 'fuel cell', 'charge', 'camera', 'battery'],

'air bags': ['airbag', 'air bag', 'knee bolster', 'inflator', 'clock spring', 'srs', 'supplemental restraint'],
'engine': ['engine', 'ignition', 'screen filter', 'pressure sensor', 'generator', 'alternator', 'drive belt', 'chain belt', 'drain plug', 'radiator', 'emission', 'catalytic convertor', 'solenoid', 'seals gasket'],
'power train': ['power train', 'powertrain', 'torque converter', 'column shift', 'tcm', 'pcm', 'park start', 'neutral start', 'floor shift', 'floorshift', 'axle shaft', 'axle assembly', 'axle hub', 'shift pattern indicator', 'transmission'],
'steering': ['steering', 'tie rod', 'gear box', 'gearbox', 'gear stick', 'mounting bracket', 'shaft pitman', 'power assist', 'knuckle', 'idler', 'handle bar', 'column locking', 'pinion shaft', 'shaft sector', 'yaw rate sensor'],
'vehicle speed control': ['speedometer', 'accelerator pedal', 'speed sensor', 'speed control', 'throttle', 'cruise control'],
'service brakes': ['brake', 'low pressure warning', 'governor', 'quick release valve'],
'fuel/propulsion system': ['fuel', 'propulsion', 'gasoline', 'refuel', 'gas', 'diesel', 'petrol'],
'tires': ['tire', 'wheel', 'tread wear', 'flat spot'],
'suspension': ['suspension', 'coil spring', 'damper', 'steering pull', 'bumpy ride'],
'exterior lighting': ['exterior light', 'running light', 'beam dimmer', 'tail light', 'reverse light', 'fog light', 'light control', 'brake light', 'headlight', 'daytime light', 'flasher unit', 'turn signal', 'turn light'],
'electronic stability control': ['esc', 'electronic stability', 'esp', 'dsc', 'dynamic stability', 'antilock brake','anti lock brake'],
'seats': ['seat', 'carseat', 'headrest', 'slide adjuster', 'adjuster rod', 'cushion'],
'seat belts': ['seat belt', 'seatbelt', 'shoulder harness', 'shoulder strap', 'buckle'],
'structure': ['structure', 'roof', 'pillar', 'body', 'escape exit', 'egress exit', 'sun visor', 'underbody', 'door', 'frame', 'ceiling', 'bumper', 'interior panel', 'dashboard'],
'equipment': ['equipment', 'antitheft', 'anti theft', 'water heater', 'storage tank', 'radio', 'tape', 'cd', 'fire suppression', 'dongle', 'gps', 'navigation', 'helmet', 'bluetooth', 'wifi', 'touchscreen', 'monitor', 'tv', 'speaker', 'air conditioner', 'air con'],
'visibility': ['vision', 'visibility', 'wiper', 'sun visor', 'windshield', 'glass', 'rearview', 'condensor', 'evaporator', 'rear window', 'defroster', 'defogger', 'hvac'],
'latches/locks/linkages': ['latch', 'lock', 'linkage'],
'child seat': ['child', 'infant', 'baby'],
'parking brake': ['parking', 'brake', 'emergency'],
'interior lighting': ['interior', 'light', 'instrument panel', 'dome', 'dash cluster display', 'dash', 'overhead'],
'traction control system': ['traction', 'stabilitrak', 'vsc light'],
'forward collision avoidance': ['collision', 'lidar', 'adaptive lighting', 'automatic emergency braking', 'automatic braking', 'pedestrian recognition', 'cyclist recognition', 'crash sensor', 'brake assist', 'dynamic brake support'],
'back over prevention': ['backover', 'back over', 'back up', 'blind', 'backup'],
'lane departure': ['lane', 'blind', 'spot', 'assist'],
```

Figure A.1.: The class dictionary used for classification.

Instructions

Hello <name>,

First of all, I would like to thank you again for your willingness to participate in my survey.

Research Summary

This survey takes place in the context of my Master's thesis on "An Expert-Defined Rule-Based Approach for Generating Vector Representations to Classify Texts". This research was made possible through the cooperation of MHP and the Technical University of Munich. It aims to develop an algorithm for classifying text documents based on predefined rules. The open source data from the U.S. National Highway Traffic Safety Administration (NHTSA) with over one million datasets are used for this research.

The processing time takes approximately 15 minutes

Instructions

Please read these instructions carefully before you start!

This survey considers the **classification of customer complaints** with respect to **vehicle components**. In the attachment, you will find **one Excel document** that corresponds to one vehicle component. So, results for exactly one component are considered. The document has two columns. The first column is *"text_span"* and contains 100 randomly selected sentences. The second column is called *'validation'* and is empty. The task is to decide whether the sentence **describes a customer problem *and* is related to the given component.**

I would like you to read each sentence carefully and put a "1" or a "0" in the 'validation' column.

1 means: the sentence describes a problem **and** is related to the given vehicle component.

0 means: the sentence **does not** describe a problem **or** is **not** related to the given vehicle component.

The component for validation is <component_name>

A brief description of the component is given:
<Component description>

Please feel the column 'validation', save the document and send it back to me **by <date>.**

I would also like to ask you to avoid other changes in the document other than setting '1' or '0' in the 'validation' column.

Please don't worry about the 'correctness' of your validation as there is no right or wrong and feel free to rely on your 'gut feeling'

Have a great week!
Best
Andrei

P.S. Pro Tip (or Capitan Obvious Tip).

If you have a Numpad on your keyboard, the numbers 0, 1, and the enter button can be entered with three fingers only.

Figure A.2.: E-mail template used to approach domain experts for the validation procedure.

List of Figures

List of Tables

Acronyms

BERT Bidirectional Encoder Representations from Transformers

BoW Bag-of-Words

CBOW Continuous Bag-of-Words

CNN Convolutional Neural Network

ESA Explicit Semantic Analysis

GloVe Global Vectors for Word Representation

KE Keyword Enrichment

LDA Latent Dirichlet Allocation

LM Language Model

LSA Latent Semantic Analysis

LSTM Long Short-Term Memory

MLM Masked Language Modelling

NHTSA National Highway Traffic Safety Administration

NLG Natural Language Generation

NLP Natural Language Processing

NLU Natural Language Understanding

NRC National Research Council

NSP Next Sentence Prediction

POS Part of Speech

PTM Pre-Trained Models

RNN Recurrent Neural Network

RoBERTa Robustly Optimized BERT Pretraining Approach

SBERT Sentence-BERT

TF-IDF Term Frequency–Inverse Document Frequency

VSM Vector Space Model

Bibliography

[1] S. Grimes. *Unstructured data and the 80 percent rule*. 2008. URL: breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/ (visited on 10/19/2022).

[2] F. Sebastiani. "Machine learning in automated text categorization". In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[3] Z. Haj-Yahia, A. Sieg, and L. A. Deleris. "Towards unsupervised text classification leveraging experts and word embeddings". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 371–379.

[4] B. Liu, X. Li, W. S. Lee, and P. S. Yu. "Text classification by labeling words". In: *Aaai*. Vol. 4. 2004, pp. 425–430.

[5] V. Ha-Thuc and J.-M. Renders. "Large-scale hierarchical text classification without labelled data". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 685–694.

[6] X. Zhang, J. Zhao, and Y. LeCun. "Character-level convolutional networks for text classification". In: *Advances in neural information processing systems* 28 (2015).

[7] N. Shafiabady, L. H. Lee, R. Rajkumar, V. Kallimani, N. A. Akram, and D. Isa. "Using unsupervised clustering approach to train the Support Vector Machine for text classification". In: *Neurocomputing* 211 (2016), pp. 4–10.

[8] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han. "Text classification using label names only: A language model self-training approach". In: *arXiv preprint arXiv:2010.07245* (2020).

[9] T. Smith, B. Stiller, J. Guszcza, and T. Davenport. "Analytics and AI-driven enterprises thrive in the Age of With". In: *Deloitte Insights* (2019).

[10] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He. "A Survey on Text Classification: From Traditional to Deep Learning". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.2 (2022), pp. 1–41.

[11] B. Romera-Paredes and P. Torr. "An embarrassingly simple approach to zero-shot learning". In: *International conference on machine learning*. PMLR. 2015, pp. 2152–2161.

[12] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo. "How to train good word embeddings for biomedical NLP". In: *Proceedings of the 15th workshop on biomedical natural language processing*. 2016, pp. 166–174.

[13] W. McKinney et al. "pandas: a foundational Python library for data analysis and statistics". In: *Python for high performance and scientific computing* 14.9 (2011), pp. 1–9.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[15] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019).

[16] U. Kamath, J. Liu, and J. Whitaker. *Deep learning for NLP and speech recognition*. Vol. 84. Springer, 2019.

[17] W. J. Hutchins. "The Georgetown-IBM experiment demonstrated in January 1954". In: *Conference of the Association for Machine Translation in the Americas*. Springer. 2004, pp. 102–114.

[18] N. Chomsky. "Syntactic structures". In: *Syntactic Structures*. De Gruyter Mouton, 2009.

[19] J. R. Pierce and J. B. Carroll. *Language and machines: Computers in translation and linguistics*. 1966.

[20] R. C. Schank and L. Tesler. "A conceptual dependency parser for natural language". In: *International Conference on Computational Linguistics COLING 1969: Preprint No. 2*. 1969.

[21] H. Somers. "An introduction to machine translation". In: (1992).

[22] D. M. Christopher and S. Hinrich. "Foundations of statistical natural language processing". In: (1999).

[23] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer. "Class-based n-gram models of natural language". In: *Computational linguistics* 18.4 (1992), pp. 467–480.

[24] Y. Bengio, R. Ducharme, and P. Vincent. "A neural probabilistic language model". In: *Advances in neural information processing systems* 13 (2000).

[25] R. Collobert and J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[26] 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* 26 (2013).

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

[28] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[29] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[30] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap. "Relational recurrent neural networks". In: *Advances in neural information processing systems* 31 (2018).

[31] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim. "RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network". In: *IEEE Access* 10 (2022), pp. 21517–21525.

[32] S. Yu, S. R. Indurthi, S. Back, and H. Lee. "A multi-stage memory augmented neural network for machine reading comprehension". In: *Proceedings of the workshop on machine reading for question answering*. 2018, pp. 21–30.

[33] G. Wiese, D. Weissenborn, and M. Neves. "Neural domain adaptation for biomedical question answering". In: *arXiv preprint arXiv:1706.03610* (2017).

[34] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. "Addressing the rare word problem in neural machine translation". In: *arXiv preprint arXiv:1410.8206* (2014).

[35] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[38] N. Reimers and I. Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[39] S. Anderson. *Languages: A very short introduction*. Vol. 320. Oxford University Press, 2012.

[40] Z. S. Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[41] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. "Enriching word vectors with subword information". In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[42] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. "code2vec: Learning distributed representations of code". In: *Proceedings of the ACM on Programming Languages* 3.POPL (2019), pp. 1–29.

[43] G. Paltoglou and M. Thelwall. "More than bag-of-words: Sentence-based document representation for sentiment analysis". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. 2013, pp. 546–552.

[44] J. Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 1. Citeseer. 2003, pp. 29–48.

[45] G. Salton, A. Wong, and C.-S. Yang. "A vector space model for automatic indexing". In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[46] J. Han, J. Pei, and H. Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[47] C. D. Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[48] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar. "Importance of Semantic Representation: Dataless Classification." In: *Aaai*. Vol. 2. 2008, pp. 830–835.

[49] E. Gabrilovich and S. Markovitch. "Feature generation for text categorization using world knowledge". In: *IJCAI*. Vol. 5. 2005, pp. 1048–1053.

[50] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[51] Y. Song and D. Roth. "On dataless hierarchical text classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[52] D. D. Lewis. "A sequential algorithm for training text classifiers: Corrigendum and additional data". In: *Acm Sigir Forum*. Vol. 29. 2. ACM New York, NY, USA. 1995, pp. 13–19.

[53] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[54] R. Johnson and T. Zhang. "Semi-supervised convolutional neural networks for text categorization via region embedding". In: *Advances in neural information processing systems* 28 (2015).

[55] D. Chai, W. Wu, Q. Han, F. Wu, and J. Li. "Description based text classification with reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1371–1382.

[56] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. "Pre-trained models for natural language processing: A survey". In: *Science China Technological Sciences* 63.10 (2020), pp. 1872–1897.

[57] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. "Why does unsupervised pre-training help deep learning?" In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 201–208.

[58] J. Pennington, R. Socher, and C. D. 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.

[59] A. M. Dai and Q. V. Le. "Semi-supervised sequence learning". In: *Advances in neural information processing systems* 28 (2015).

[60] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[61] J. Sarzynska-Wawer, A. Wawer, A. Pawlak, J. Szymanowska, I. Stefaniak, M. Jarkiewicz, and L. Okruszek. "Detecting formal thought disorder by deep contextualized word representations". In: *Psychiatry Research* 304 (2021), p. 114135.

[62] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. "Improving language understanding by generative pre-training". In: (2018).

[63] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[64] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[65] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[66] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems* 6 (1993).

[67] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. "A large annotated corpus for learning natural language inference". In: *arXiv preprint arXiv:1508.05326* (2015).

[68] A. Williams, N. Nangia, and S. R. Bowman. "A broad-coverage challenge corpus for sentence understanding through inference". In: *arXiv preprint arXiv:1704.05426* (2017).

[69] *Model Distillation*. `https://www.sbert.net/examples/training/distillation/README.html`. Accessed: 2022-11-11.

[70] C. C. Aggarwal and C. Zhai. "A survey of text clustering algorithms". In: *Mining text data*. Springer, 2012, pp. 77–128.

[71] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[72] J. Boyd-Graber, Y. Hu, D. Mimno, et al. "Applications of topic models". In: *Foundations and Trends® in Information Retrieval* 11.2-3 (2017), pp. 143–296.

[73] D. D. Lee and H. S. Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (1999), pp. 788–791.

[74] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[75] C. B. Asmussen and C. Møller. "Smart literature review: a practical topic modelling approach to exploratory literature review". In: *Journal of Big Data* 6.1 (2019), pp. 1–18.

[76] T. Schopf, D. Braun, and F. Matthes. "Lbl2Vec: an embedding-based approach for unsupervised document retrieval on predefined topics". In: *arXiv preprint arXiv:2210.06023* (2022).

[77] M. J. Denny and A. Spirling. "Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it". In: *Political Analysis* 26.2 (2018), pp. 168–189.

[78] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. "Universal sentence encoder". In: *arXiv preprint arXiv:1803.11175* (2018).

[79] G. T. Henry. *Practical sampling*. Vol. 21. Sage, 1990.

[80] L. A. Goodman. "Snowball sampling". In: *The annals of mathematical statistics* (1961), pp. 148–170.

[81] L. S. Penrose. "The elementary statistics of majority voting". In: *Journal of the Royal Statistical Society* 109.1 (1946), pp. 53–57.

[82] J. Cohen. "A coefficient of agreement for nominal scales". In: *Educational and psychological measurement* 20.1 (1960), pp. 37–46.

[83] J. R. Landis and G. G. Koch. "An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers". In: *Biometrics* (1977), pp. 363–374.

[84] S. Vijayarani, M. J. Ilamathi, M. Nithya, et al. "Preprocessing techniques for text mining-an overview". In: *International Journal of Computer Science & Communication Networks* 5.1 (2015), pp. 7–16.

[85] J. Kaur and P. K. Buttar. "A systematic review on stopword removal algorithms". In: *International Journal on Future Revolution in Computer Science & Communication Engineering* 4.4 (2018), pp. 207–210.

[86] E. Loper and S. Bird. "Nltk: The natural language toolkit". In: *arXiv preprint cs/0205028* (2002).

[87] J. Charbonnier and C. Wartena. "Using word embeddings for unsupervised acronym disambiguation". In: *Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, August 20-26, 2018*. 2018, pp. 2610–2619.

[88] J. Seo, S. Lee, L. Liu, and W. Choi. "TA-SBERT: Token Attention Sentence-BERT for Improving Sentence Representation". In: *IEEE Access* 10 (2022), pp. 39119–39128.

[89] T. Gao, X. Yao, and D. Chen. "Simcse: Simple contrastive learning of sentence embeddings". In: *arXiv preprint arXiv:2104.08821* (2021).

[90] L. E. Peterson. "K-nearest neighbor". In: *Scholarpedia* 4.2 (2009), p. 1883.

[91] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[92] Z. Liu and N. F. Chen. "Dynamic sliding window for meeting summarization". In: *arXiv preprint arXiv:2108.13629* (2021).

[93] J. Shen, W. Qiu, Y. Meng, J. Shang, X. Ren, and J. Han. "TaxoClass: Hierarchical multi-label text classification using only class names". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 4239–4249.

[94] M. Grootendorst. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure". In: *arXiv preprint arXiv:2203.05794* (2022).