How big is the real estate property? Using zero-shot vs. rule-based classification for area categorization in real estate contracts.

Masterarbeit

Zur Erlangung des Mastergrades
Master of Science (MSc)

an der Fakultät für Digitale und Analytische Wissenschaften
der Paris-Lodron-Universität Salzburg

Eingereicht von:	Julia Angerer, BSc. MSc.
Gutachter:	Univ.-Prof. Dipl.-Inform. Dr.-Ing. Christian Borgelt
Fachbereich:	Artificial Intelligence and Human Interfaces (AIHI)

Salzburg, April, 2024

Abstract

Due to the massive amount of real-estate related text documents, the necessity to automatically process the data is evident. Especially purchase contracts contain valuable transaction and property description information, like living area. In this research project, a natural language processing (NLP) approach using open-source transformer-based models was investigated.

The potential of pre-trained language models for zero-shot classification is highlighted, especially in cases where no training data is available. This approach is particularly relevant for analyzing purchase contracts in the legal domain, where it can be challenging to manually extract the information or to build comprehensive regular expression rules manually.

A data set consisting of 668 classified contract sentence parts, each containing one area and context information, was created manually for model comparison purposes. The experiments conducted in this study demonstrate that pre-trained language models can accurately classify sentence parts containing an area, with varying levels of performance across different models (the highest macro averaged F1-score was 0.87). The results suggest that pre-trained language models can be effective tools for processing textual data in the real estate and legal domains, and can provide valuable insights into the underlying structures and patterns in such data.

Overall, this research contributes to the understanding of the capabilities of pre-trained language models in NLP and highlights their potential for practical applications in real-world settings, particularly in the legal domain where there is a large volume of textual data and annotated training data is usually not available.

Keywords: *natural language processing, contract documents, information extraction, natural language inference, zero-shot classification*

Contents

Prelude

The present thesis has been written in corporation with the company DataScience Service GmbH which is offering and selling automated real estate valuation and comprehensive digital market information in the field of real estate. Real estate valuation refers to the process of determining the monetary value of a property at a certain point of time (Frew and Jud, 2003). The real estate text data used in the analysis was provided by DataScience Service GmbH in order to facilitate research in general but also to use state of the art methods in product components.

1 Introduction and Business Understanding

The automated and computerized processing of written language is of concern to many research fields. In the field of real estate, these methods can be applied to extract information out of transaction documents.

The immense volume of transaction data necessitates automated processing methods. In Austria alone, over 100,000 real estate transactions take place each year. Manually reviewing such a vast number of contracts to extract key information, like areas, prices, and contracting parties, is impractical for an individual. Hence, there is a clear need for automated text processing techniques to efficiently extract relevant content and categorize these documents.

In the thesis the steps from a contract PDF to the extracted and categorized information will be described and performed on a sample of documents. The main concern lies in the description of the process but also the comparison of two attempts to categorize the information. This section contains domain specific real estate information to ease understanding of the underlying data and business need.

During the life cycle of a property, the value, which is the price an owner can offer and a potential buyer could accept on the market, is essential information for multiple persons and institutions like banks, brokers, insurance companies, and homeowners to make informed decisions (Frew and Jud, 2003; Yazdani and Raissi, 2023). The *title transfer* in real estate refers to the moment when ownership of a property transitions from one person to another (Globalnegotiator, 2017). It marks the formal transfer of legal rights and responsibilities associated with the property. In the context of real estate transactions, this event is typically described and documented in contracts, also known as transaction documents.

All transactions of a real estate from one owner to another involve a transaction document. Even though most transaction documents are purchase contracts and therefore contain a purchase price as a form of monetary value expression, not all kinds of real estate transactions are purchases. There are different types of real estate transaction documents, depending on the type of payment. In *purchase contracts* the transaction of money is the usual case whereas in *donation contracts* the transaction involves gifting without any payment. Apart from that, there exist more rare types of transaction documents like *exchange contracts* where two properties are swapped. Since only purchase contracts are investigated in the current study, other types of contracts will not be discussed any further.

The purchase price of the property should always be present[1] in a real estate purchase contract and can be extracted from the documents e.g. by a human. The price depends on a variety of features of the real estate, is agreed on by both the seller and the buyer and is usually close to the actual market value that can be determined with real estate valuation. The main categories of residential real estate[2] are apartments, family houses, and unbuilt plots of land. The average price of these types of property is monitored in Austria in the form of average price per square meter by various companies. One of these companies is Statistics Austria, which is the governmental institution responsible for European and Federal Statistics produced within Austria (Der Standard, 2021; Statistik Austria, 2023). Monitoring of average prices takes areas and other property features into account to ensure plausible time trends over the years (Statistik Austria, 2023).

Main components that affect the price of built as well as unbuilt properties on the market are the region (e.g. in which province), the location (e.g. district, floor), the infrastructure, and distance to points of interest (e.g. distance to highway, railway station, and family doctor). In case of built properties also the usable or living area, age and upcoming renovation costs matter (Der Standard, 2021).

Besides the price, usable or living area is another variable of interest that can be found in transaction documents of built real estate. The areas like living area, usable area but also plot area, can be used to put the price of the transaction into perspective. Different types of area can occur in the documents, depending on the type of property. For example when

[1]Formally, the price needs to be defined to ensure a valid purchase contract and title transfer, but the price can be unknown to the reader of the actual purchase contract. Cases of scanned purchase contracts having no price have been observed by the author, but are rare. A reason for an unknown price can be a missing page in the scanned contract document. Another reason can be that the price is defined in a different document, which the purchase contract refers to, but that document is not appended.

[2]*Residential* real estate is real estate where people actually live, like single or multi-family houses. On the other side, there is *commercial* real estate, which are e.g. shops, offices, and other types of properties not used for permanent living of natural persons. Both types are usually built properties, but can also be unbuilt plots of land with the dedication of using them in a residential or commercial way.

a property is unbuilt, the plot area is the size of interest, whereas for a built plot the living area or usable area are more important compared to the plot area. In case of unbuilt plots of land it is rather easy to determine the area of the property, since additional resources like the Austrian cadastre[3] can be used to retrieve this information with the parcel identification number, but for built plots it is difficult to determine the area of interest. Consequently, the price per square meter of living or usable area is unknown if not extracted from the purchase contract, since there is no publicly open register of living or usable area. Hence, reading out the information of the area from purchase contract documents is necessary.

Since the description of the object of purchase, including price and area, are important for usability of the transaction data for real estate valuation, the investigation and continuous improvement of area extraction methods is highly important for the industry.

One approach, that lays at hand when automatizing processes that humans can perform, is to mimic human behaviour. In respect of extracting area information from a contract, a human already knows many things about texts as well as understanding and extracting information – but a human needs a lot of time for that.

To illustrate the amount of time needed for manual extraction of information, assume an experienced person is manually checking a contract, which has the fictive length of roughly 40 pages,[4] for containing any information about area and extracts it in 10 minutes on average. With this assumed speed, six contracts would be processed in one hour and 48 contracts on an 8 hours working day. To extract the area information, and nothing else, of 1,000 contracts, the person would work 20.8 days, which is 4.2 working weeks[5]. Furthermore, the manual extraction of areas by one individual from the yearly amount of 100,000 real estate contracts would take 1.8 years without vacation or sick leave[6].

One of the initial steps in mimicking human behaviour in information extraction is the *recognition of characters* from a picture containing text. A human sees the letters and if they belong to a language the human speaks, the letters are recognized and combined to words. Machines can comprehend text from images as well, with a procedure called *optical character recognition* (OCR). During OCR, the software categorizes white parts of the text image as background and black parts as (potential) text. Moreover, letters and symbols of known fonts are matched to the black areas in the image and the best matching letter for every image

[3]The Austrian *cadastre* is an information system about reference (parcel identifier, address, and location), area, borders, and use of real estate, having the purpose of real estate protection and is accessible to the general public (BEV, 2023).

[4]A more detailed description and example of a purchase contract are given in Section 3. The amount of pages purchase contracts analyzed in the data set here is reported in Section 4.2.

[5]Assuming a working week has 5 days.

[6]The estimation of time needed is calculated as: 16,666.7 hours = 2083.3 working days (à 8h) = 416.7 weeks (à 5 working days) = 96.2 months (à 4.33 weeks) = 1.8 years (à 52.1 weeks).

part is kept in order to create a textual representation of the image (Chaudhuri et al., 2017). After OCR procedure is done, the text document can be further processed – it can be stored, modified, searched, and analyzed.

The text document is the baseline both humans and machines can work on when extracting area information. As a core step in analyzing textual information, a human would read briefly over the text and try to detect the relevant parts that contain any areas. Usually this would be digits followed by a quantity like square meters. If any of the two patterns, either digits or square meter would appear in a text, a human would most certainly read slower and "scan" the text around that in more detail to find out if the text part is relevant or not and if some information from there should be noted down. An obvious method to mimic the human behaviour with a machine is to also analyze the text for digits and square meter occurrences and cut out the text parts containing such information to analyze them in more detail. It is rather easy to extract the digits followed by square meter, but it is less trivial to assess what the area refers to.

An example sentence from a purchase contract might help to illustrate how important and precarious the detection of the reference of the areas to the other words in the sentence is:

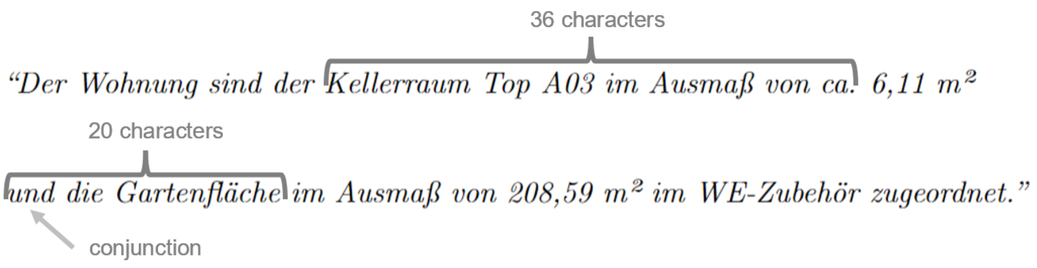
> *"Der Wohnung sind der Kellerraum Top A03 im Ausmaß von ca. 6,11 m^2 und die Gartenfläche im Ausmaß von 208,59 m^2 im WE-Zubehör zugeordnet."*

An English translation maintaining the word order (and neglecting the grammar) is:

> *"(To) the apartment are a cellar room Top A03 in the extent of approx. 6.11 m^2 and the garden area in the extent of 208.59 m^2 assigned in the condominium ownership accessories."*

The sentence is from the legal domain and more difficult to read than a non-legal text (like e.g. a newspaper article) because of the sentence structure as well as domain specific abbreviations or terms, like in this case the "WE" which means German "Wohnungseigentum" (engl. condominium ownership). The text part originates from a typical paragraph of a purchase contract, that is describing several areas of an object of purchase. The full paragraph is shown in Figure 15 and discussed in more detail in Section 3.

Where does the area of 6.11 m^2 belong to? A human understands the text and knows it belongs to the cellar. If the text is complicated or in a language not spoken fluently, a human could still try to determine the reference, e.g. by assuming the area is closer (less characters away, see Figure 1) to the noun it refers to compared to other nouns. In case of the "6,11 m²" this would be a wrong approach because the word cellar, where the area actually refers to, is farther away than the word garden, which is closely after the "6,11 m²". Obviously the sentence has two sub-sentences connected with the conjunction "und" (engl. and).



"Der Wohnung sind der Kellerraum Top A03 im Ausmaß von ca. 6,11 m² und die Gartenfläche im Ausmaß von 208,59 m² im WE-Zubehör zugeordnet."

Figure 1: German contract sentence with character counts of sentence parts containing cellar (ger. Keller) and garden (ger. Garten).

Moreover, humans can clearly state that this is a sentence but for machines it is more difficult to understand where a sentence starts and ends because e.g. a dot like in "ca." (engl. approx.) does not mark the end of a sentence but rather a common abbreviation.

Of course, not all humans read text in the same way and the assumptions and explanations in regard of this single sentence do only scratch the surface of language processing and language understanding – but the example highlights that human language processing behaviour can be described in detailed steps. Those steps, and many other steps, are used in contract text processing in the current thesis. The workflow starting with the contract text documents (retrieved by OCR) to the extracted and classified areas, is shown in Figure 2. The process of classification serves the purpose of organizing sentence parts, and the areas in the sentence parts, into pre-defined categories like living or usable area, balcony, garden, and other categories.

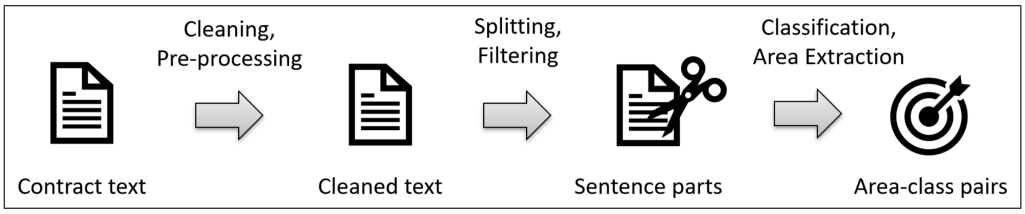


Figure 2: Processing workflow from text document to areas with related class.

The aim of the current thesis is to investigate a workflow to extract areas and classify them accordingly. The workflow starts with the contract texts obtained through OCR, which need text cleaning in order to remove mistakes caused via OCR and to harmonize the text basis. Moreover, the approach contains detection of text parts containing an area and extracting those sentence parts. Like in the example sentence stated above (Figure 1), it is assumed that the right amount of context information around the area, but not containing conjunctions like *and* helps to identify the correct category of the area. Therefore, text parts, like the example sentence, are split at the conjunction (and other parts if present, further details are given later, see Section 4.4) and the sentence parts are further passed to a classifier that determines the category the sentence part belongs to.

Even tough the wording within contracts can be different, the most prominent types of areas can be subsumed with certain words or category names. Usually a contract about an apartment mentions a living or usable area, some balcony, garden, or terrace as well as parking area (if any is present). Therefore, these and a few other categories were set to be the *labels* or *classes* the classification approaches assigns the sequences to (the full list of class labels used is reported in Section 3.3).

Despite some classes are prominent, others are more rare, or occur more often in just a single region or time frame. Certain categories or information is more important compared to other information. The living area or usable area are in general more important compared to cellar or garden area. In the present study, areas and their respective classes are analyzed. Before analyzing the documents and extracting the desired information, the *business need* needs to be stated. The need of the cooperation company is the extraction and classification of real estate areas, especially living area and usable area. Other areas like e.g. the balcony area are regarded as features to the property. Their extraction is desired as a secondary goal.

Within the cooperation company, there are two use cases the extracted and classified areas are needed for. On the one hand, there is a demand for extraction and classification of areas, which comes from the team that builds and maintains the automated valuation model (AVM, model to assess the monetary value of a property), where the areas can be input to the real estate valuation model. On the other hand, there is a demand for areas of real estate from customer side. On the customer side, the real estate data is desired information for brokers, banks, appraisers, and other institutions working in the domain. The usage of area information both inside the company products but also as part of a data product makes the accurate extraction and classification so important.

Even though it is already clear in advance, that especially living area and usable area are most important to be determined from the real estate contracts, the desired categories can change and other categories can become important as well in the future, e.g. due to projects or new customers for data and software products. Therefore, the amount of categories the sentence parts containing an area (and areas themselves) can be assigned to is not fixed at all. Hence, a classification approach was researched where categories can be added or removed easily and that does not need any kind of training[7] but is rather flexible. The predominant classification approach applied in this thesis has been inspired by Yin et al. (2019), who claimed first to use Large Language Models (LLMs) trained on assessing the connection between sentence pairs for classification purpose. Yin et al. (2019) utilized the classification approach without any training, called zero-shot classification, via a method called *natural language inference.* LLMs are trained beforehand on a massive amount of text, have representations of one or more human languages stored and are afterwards trained additionally on a task to perform – for instance natural language inference. This enables the models to infer from a given text part (e.g. "Gartenfläche im Ausmaß von 208,59 m²") and a statement (e.g. "This is about living area."), if this is a logical inference from the given text. The combination of text parts and statements is performed multiple times in order to find out the categories that matches the given text part best (i.e. has the highest entailment). The combination of text parts and statements are called premise-hypothesis pairs and will be discussed in more detail in the Related Work section (see Section 2.8). Moreover, LLM will be described in general in more detail in the next section.

Besides that, the next section will cover additional information about the processing of human languages in general. The term Natural Language Processing (NLP) describes a computerized approach of analyzing text using a range of computational techniques to understand and represent naturally occurring texts (Liddy, 2001). NLP, which is a field located at the intersection of Linguistics and Artificial Intelligence (Khurana et al., 2022), is supporting us in tasks such as reading, summarizing, classifying, and comprehending text. Apart from that, NLP is a research field with a very fast development in the last years and with yearly increasing amount of publications regarding new methods and models to process text data (Chen et al., 2022).

Summing up, the approach pursued and described in the current thesis is aiming to extract and classify different kinds of areas from real estate purchase contracts. Therefore, relevant sentence parts are isolated, areas are extracted, and the sentence parts as a whole are classified with the assumption that the sentence is best describing the area in there. The novel method of classification used for this approach is zero-shot classification with Large Language Models, which is rather flexible regarding including or excluding categories in the classification approach. Common areas occurring frequently in contracts serve as categories and will be applied in the initial classification approach described in the current work.

Moreover, in order to have data for evaluation purpose, a manual ground truth will be created. This ensures that any metrics describing the quality of the model results can be reported.

[7]Training is sometimes also called learning and involves the stepwise adaption of parameters in a model to improve the model performance. The performance is good when the difference between the models estimation and the labelled result is low (Kruse et al., 2016).

Furthermore, it will be investigated if the zero-shot classification is better compared to only matching the category names as patterns to the sentences. This alternative approach to zero-shot classification is further called *rule-based* approach because the category names are passed to the machine as *rule* to search for in the text.

Like already mentioned, zero-shot classification with LLMs is applied because it is rather flexible regarding *which* and *how many* categories are used and it does not need training data. Training data is usually data where categories or other information about the texts and the task are already defined and known beforehand. Generation of any kind of training data by humans is time consuming and therefore costly. Thus, the amount of contracts published with categories or any other other kind of explicit form of meaning and content (annotation) is sparse (Wang et al., 2023) and no data like that was available for the current thesis.

In this context, the present thesis is exploring the practicality of employing language models, that have been pre-trained on diverse domains, are assumed to have an internal representation of natural language and were fine-tuned on a classification task, in the realm of real estate purchase contracts. The aim is to apply these models to effectively classify sentence parts related to usable area, living area, and other kinds of area. The huge advantage of those models is that they do not need any kind of beforehand classified (annotated) contracts to perform the classification tasks.

To the best of the author's knowledge, no comparative analysis between several different zero-shot models as well as an alternative rule-based approach in the context of real estate contracts has so far been published in scientific literature. Further, this thesis is assumed to be the first work that reports an end-to-end purchase contracts text processing workflow, supported by LLMs-based classifiers, aiming to automate the extraction and classification of living area and usable area.

In the last decades, the invention of new and more capable methods for processing of huge amounts of textual data has been the aim of many researchers. An overview of classical and more recent methods for text analysis with a focus on information extraction is presented in Section 2. After that, Section 3 contains a more detailed description of real estate purchase contracts because the data focused on in the later sections originates from that domain. The following sections will cover descriptions of the data set features (see Section 4.2), methods of data selection and processing (see Section 4), model selection and descriptions (see Section 5) and assessment of the model's results (see Section 6) as well as the discussion with overall conclusion of the current project (see Section 7).

2 Related Work

This section covers an overview of methods and research that is relevant to understand the methods used in this thesis. Some methods are described superficially, as the description serves to place them in a theoretical and scientific context. Other methods, especially those that are applied later, are explained in greater detail. First, the core concepts and terms needed to understand the architecture of different models and methods processing numeric data is explained (Section 2.1), followed by methods to transform and process text data (starting from Section 2.3). Thereafter, the history of text classification methods is outlined and the emergence of Natural Language Processing (NLP) is covered briefly (Section 2.6), followed by a more detailed explanation of specific methods, including the transformer architecture (Section 2.7) and an application of the transformer architecture that assesses the connection or relationship between two sentences (Sections 2.8, 2.9). In the end, related research in the legal domain (Section 2.10) is presented since purchase contracts are documents from the legal domain.

2.1 Neural Networks and Learning

Whenever utilizing approaches and tasks humans perform, having a look at *how* humans perform them lays at hand. Artificial neural networks (ANNs) are an information processing system inspired by neurons in the brains of humans and animals (Kruse et al., 2016). An artificial neural network can be thought of as a digital brain made up of interconnected nodes known as *neurons* or *units*. These single computational units are the building blocks of the network and take a vector of real-valued numbers as inputs, process them and returns an output (Jurafsky and Martin, 2023g). The functionalities of the ANN described below are derived from the textbook authored by Kruse et al. (2016) and the chapter on ANNS in the textbook by Jurafsky and Martin (2023g), both providing even more detailed explanations of ANNs. Since only artificial neural networks, and not any natural neural networks like the human brain, will be further explained in this thesis, the term *neural networks* will specifically refer to artificial neural networks (ANN = NN) below.

Units in a neural network have associated *parameters* called *weights* and *bias*, which specify how they respond to incoming information. These weights function as the strengths of connections between neurons. Every connection from the incoming information to the neuron as well as to other neurons has weights. More precisely, when giving the neural unit a vector x with length n, there have to be n corresponding values in the weight vector w as well, plus the additional scalar bias value b. The weighted sum z can be represented using vector notation with a dot product of the vectors, which is visible in the equation below on the right side:

$$z = b + \sum_i w_i x_i = \mathbf{w} \cdot \mathbf{x} + b$$

The variable z is a linear function of $\mathbf{x}$, but neural networks apply a non-linear function called f to z. This is an important step to enable neural networks to model complex real word data that has non-linear relationships. When using a linear function, the units would only be able to model linear relationships between input and output (McQuillan, 2022).

The output y is the *activation* as well as in this case the output of the unit:[8]

$$y = a = f(z) = f(\mathbf{w} \cdot \mathbf{x} + b)$$

One of the most popular activation functions f is the sigmoid function (usually expressed by σ), which maps z to the range 0 to 1. For further details on activation functions and neural networks in general see text books and chapters about neural networks (e.g. Jurafsky and Martin, 2023g; Kruse et al., 2016). For more specific details which activation functions are usually used in Large Language Models, see e.g. the blog post by McQuillan (2022).

Figure 3 shows the schematic of such a neural unit. In this example, the input vector as well as the weight vector have length three and the values of the input vector are multiplied by the values of the weight vector. The bias value b is added to the sum and passed through the activation function σ.

A neural network usually consist of more than one unit and the units are connected with each other. In general, the units are organized with a hierarchy, where the neurons of the same hierarchy belong to the same *layer*. A neural unit can belong to one of these types of neurons:

- *Input neurons* perceive input from the environment and take the raw input from the domain.
- *Hidden neurons* perceive input from one neuron and output to another one. They do not have contact with the "outside" and are therefore called hidden.
- *Output neurons* are generating the desired output based on the information processed in the hidden layers.

[8]With this notation, a is the output of the *activation* function and y is the output of the complete *neural network*. The variables a and y are in this case the same because the "network" consists of only one neuron.

Jurafsky and Martin (2023g), who have written a book chapter about neural networks in their book about language processing, uses this notation. The equations can be found with additional explanations in the respective chapter.

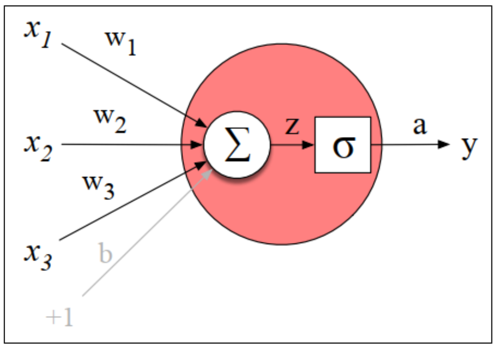


Figure 3: Illustration of a neural unit with input $\mathbf{x}$, weight w, bias b. The output of the activation function is a, which is in this case equal to y. The original illustration can be found in Jurafsky and Martin (2023g, p. 135).

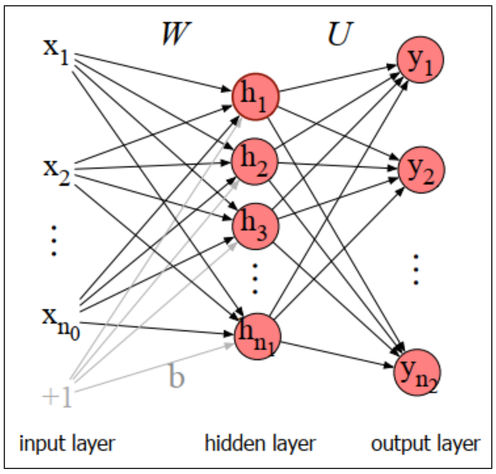


Figure 4: Illustration of a feed forward network consisting of neural units with input layer, hidden layer, and output layer. The original illustration can be found in Jurafsky and Martin (2023g, p. 140).

A simple form of a neural network consisting of more than one unit is a *feed forward network* (Jurafsky and Martin, 2023g). Such a network has only connections from one node to another, without any cycles. This can be in the shape of a layered network, that has connections from one layer to layer to the next one, which means the outputs of neural units in one layer are passed to units in the next higher layer. Nevertheless, not all layered neural networks are feed forward networks. The feed forward network shown in Figure 4 has 2 layers (the input layer is usually not counted): a hidden layer and a output layer.

The activation function in the hidden layer is applied to vector h and therefore allowed to be applied to a vector element vise. An activation function that is common in simple feed forward networks for the output layer is the softmax function, which normalizes z to a probability distribution.

Computations in a feed forward network can be done more efficiently with matrix operations than with single vector computations. Therefore, for every layer the weight vectors and bias values are combined to matrices and vectors respectively. Like explained above, the parameters of a single hidden unit are a weight vector and a bias value. The weight matrix W (see also Figure 4) represents the parameters of all the units in the hidden layer and every element W_{ij} represents the weight from the i^{th} input to the j^{th} hidden unit. Since every hidden unit can have a different bias value, the dimensionality of the hidden layer as well as the bias vector is n_1. The number of inputs as well as the length of the input vector are n_0. Therefore, the dimensionality of the weight matrix is $W \in \mathbb{R}^{n1 \times n0}$.

The output layer has a weight matrix similar to the one of the hidden layer(s) and is called U in Figure 4. The weight matrix U has the dimensionality $U \in \mathbb{R}^{n2 \times n1}$ because the length of the output layer is n_2 and there are n_1 units in the hidden layer. After the intermediate output z is obtained via matrix multiplication, z needs to be normalized with a function in order to obtain probabilities instead of real-valued numbers, e.g. with the softmax function.

Moreover, Figure 4 shows a *fully-connected network*, which means that every unit from the previous layer outputs to each unit of the next layer. This is illustrated in the figure by black arrows pointing from each unit in one layer to every unit in the next layer, e.g. the connections from the elements x_1, x_2, ..., x_{n_0} to all the units h_1, h_2, ..., h_{n_1} of the hidden layer. The standard architecture is a fully-connected network, but it does not necessarily have to be the case.

When a neural net, like a feed forward network, is trained – the true and desired y has to be known. The output of the network is just an estimation $\hat{y}$ of the true output value. During training the network adjusts its weights and the bias value to ensure the estimation $\hat{y}$ matches the true y, or comes as close to it as it can. The distance between the networks output and the true output is usually modelled by a so called *error function.* The higher the difference between $\hat{y}$ and y is, the higher is the error.

Training a neural network involves solving an optimization problem where the objective is to minimize the error. To achieve this, the partial derivative of the error function with respect to each parameter in the neural network must be computed. This process, often referred to as “computing the gradient”, is a crucial step in the overall optimization algorithm known as *gradient descent.* In the context of small feed forward networks, such as the one illustrated in Figure 4 with only one hidden and one output layer, manual computation of gradients is feasible (refer to Jurafsky and Martin (2023g), p. 150ff for equations and examples).

However, for larger neural networks with multiple hidden layers, this approach is more complicated to realize per hand. The reason is that the gradients calculated through manual computation only facilitate the update of weights in the *last* hidden layer[9]. In order to update weights in preceding hidden layers, a technique called *error backpropagation* (Rumelhart et al., 1986) is used. Since the error values of a layer can be computed from the error of the previous layer, the errors can be traced from the output layer over all hidden layers back to the first hidden layer. Error backpropagation involves propagating errors backward through the network, including numerous intermediate results and a nested structure of function evaluations, allowing the optimization algorithm to effectively adjust all parameters. For further explanations see textbooks like Kruse et al. (2016).

Furthermore, in the context of training, *loss functions* are regularly mentioned. The distinction between a loss function and an error function lies in that the loss function quantifies the severity of the error, whereas the error function describes the deviation between the true and observed values. Depending on the objective, the loss function can be asymmetric. For example, wrong values of a certain direction might be "punished" more severely (e.g., false positives, negative values, ...). A common loss function is the mean squared error (Ben, 2020). For classifications, the cross-entropy loss is a common loss function. Especially in combination with the softmax function, the cross-entropy loss is commonly used to measure how well the predicted probability distribution fits the true distribution of class labels (Wang et al., 2022).

Essentially, a neural network is a computational model that learns and adapts by adjusting its internal parameters. Summarizing the preceding paragraphs, it can be stated that during training, the network adjusts its parameters to approach the true output as close as possible. The network learns from examples and progressively improves its abilities using an error function, minimizing the error with the help of derivatives.

Adapting internal parameters enhances the network's performance on a specific task, such as classification of images, texts, or other kinds of data. Even though the overall architecture of a neural network processing text is different to the architecture of a neural network processing images, the small neural units are the elementary building blocks in both kinds and the principle of how training works is the same as well. An example for language processing is e.g. a neural network learns to translate between languages by adjusting its weights based on numerous translated sentences until it can accurately translate new text.

2.2 Example of a Feed Forward Neural Network Processing Text

Hidden values are some sort of representations of the input (Jurafsky and Martin, 2023g). The output layer is a special layer that takes the output of the last hidden layer as input and

[9]To be more precise, this concerns the weights of connections from the last hidden layer to the output layer (Kruse et al., 2016).

computes the final output. Depending on the purpose of the network, the final output differs. Even in case of a classification task, the output can have different shapes. For example for a classification between two classes one value as an output can be sufficient, whereas for a classification with several classes, the probability of every class could be an element in the output vector (Jurafsky and Martin, 2023g).

A simple example helps to illustrate how a feed forward network like the one shown above in Figure 4 can process a sequence of words. The example involves text classification, which is assigning text pieces to pre-defined categories. A text classification task with 2-3 possible outcome classes is *sentiment analysis*, where a positive, negative, or (optional) neutral sentiment[10] is assigned to a text piece.

Suppose the sequence of words "view was great" should be classified with a 2-layer feed forward network regarding the sentiment. One approach illustrated by Jurafsky and Martin (2023g, p. 145) is to design scalar features that characterize the words in the sequence and use those features in a vector $\mathbf{x} = [x_1, x_2, ..., x_n]$ as input to the neural network. One of those features could be the count of words in the sequence (x_1), another could be the count of words that are assessed to be generally positive according to a lexicon (x_2) and a third one could be either 1 if "no" occurs in the sequence or 0 if not to indicate occurrence of a negation (x_3). These three features form the *feature representation* of the input observation $\mathbf{x}$. When the sentiment can be one of the three categories *positive* $+$, *negative* $-$ and *neutral*, there are three output neurons $\hat{y_1}, \hat{y_2}, \hat{y_3}$ referring to the probability of each of the categories respectively ($+, -$, neut.). In case of the sequence of words "view was great" the probability of $\hat{y_1}$ ($+$) should be higher than the other two probabilities because the sentiment of this sequence is positive.

The computations of the 2-layer feed forward network can be summarized with the equations:

$$\begin{aligned}
\mathbf{x} &= [x_1, x_2, ..., x_n] && \text{(n = 3 features in this example.)} \\
\mathbf{h} &= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \\
\mathbf{z} &= \mathbf{U}\mathbf{h} \\
\hat{\mathbf{y}} &= \text{softmax}(\mathbf{z})
\end{aligned}$$

The architecture for a neural network processing this example is sketched in Figure 5. The example shows the sentence part (also called sequence) "view was great", which is basically a string of letters and white space. In the processing of textual data in general, such sequences

[10]Moreover, sentiment can be defined as a positive/negative evaluation expressed with language. It is like an affective meaning, but only positive or negative and optionally neither nor, which is called neutral (Jurafsky and Martin, 2023f).

are usually split at some place that is determined to be the word separator, like e.g. the white space. It does not necessarily have to be a white space, especially in English, where terms can consist of more than one word, are concatenated and each word alone has a different meaning then the combination of words (e.g. white space, single-family house). The process of separating and identifying words out of a running text or sequence is called *tokenization* and the words or word parts generated by that are called *tokens*. Depending how the tokenization is done, the tokens can be words or not. Therefore, in later explanations, tokens and words are sometimes used as synonym.

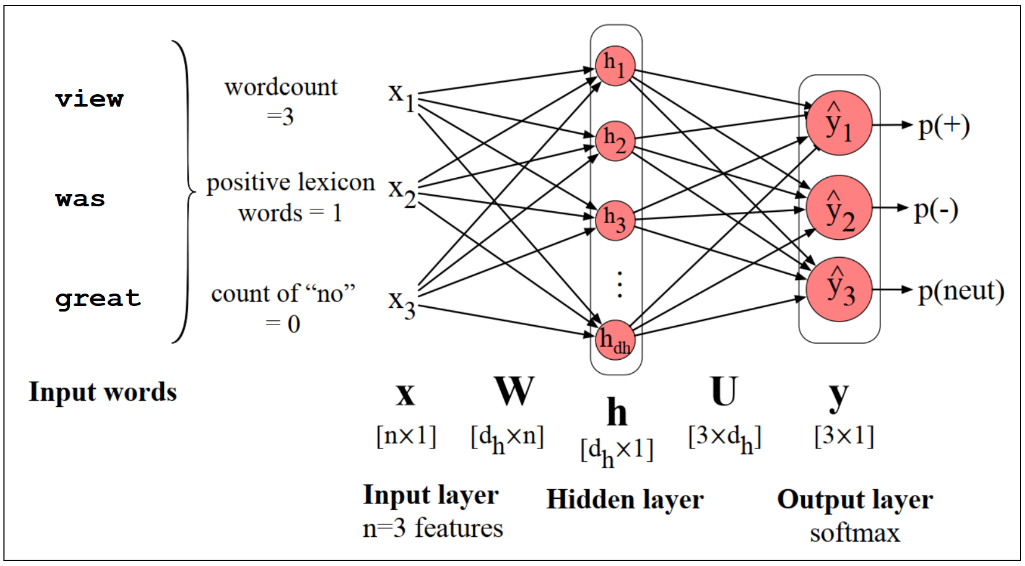


Figure 5: Illustration of a 2-layer feed forward network for sentiment classification. The network consists of several neural units in the input, hidden and output layer and takes three features of a text sequence as input. The original illustration can be found in Jurafsky and Martin (2023g, p. 145) and has been slightly modified. $p()$ refers to the probability of the sentiment.

The features in this example (Figure 5) can be called *hand designed* or *human-engineered* because they did not occur naturally in this form but are rather manually designed and a method to transform the words to the numeric input which is needed in the neural network. In most neural networks for text processing, the features are not hand designed – the networks are trained to learn the relevant features themselves. In this training process, the data is represented as embeddings, which are vectors representing words (Jurafsky and Martin, 2023f). Before embeddings are explained in more detail, the common methods and models for representation of words are explained next.

2.3 Representing Words as Vectors

In order to analyze textual data effectively, a fundamental question is how to represent text in order to make it available to models. Depending on the kind of model that is applied, the kind of representation used varies. One way is to represent words as pure letter string, another one is to represent words with identification numbers and process the number (Jurafsky and Martin, 2023f). These methods are rarely used in modern NLP models. One reason for not using words directly is that a classifier working with the words would need to be trained on the same words that are going to occur afterwards during classification (Jurafsky and Martin, 2023f). Therefore, a representation of words as vectors, also taking into account *similarity* of words, is used for text processing in NLP.

To comprehend that words can be represented as vectors, it needs to be described that words can have relationships to each other (Jurafsky and Martin, 2023f). Words can be *similar*, indicating that they share certain features or usage to some extent, as exemplified by *window* and *door*. Both separate the outside and inside of a building, can be opened and closed, and have handles, among other similarities. Additionally, words can be *related* to each other, implying an association within a similar context or domain (also known as a semantic field), even though they have different features. Due to health and safety reasons, it is not advisable to have a tiled fireplace for heating in a home without any form of chimney or other mechanism to remove the smoke – thus, *tiled fireplace* and *chimney* are related words. However, a tiled fireplace and a chimney are certainly not similar.

Vector semantics is a way to represent word meaning with the help of its use in language (Jurafsky and Martin, 2023f). Similar words are assumed to be used in similar environments i.e. are surrounded by similar neighboring words. Words can be represented as points in a multidimensional semantic space where the space is derived from word neighbors. Therefore, vector semantics is the representation of the meaning of words as vectors in an n-dimensional space.

One method of vector representations of words and documents, which is often used in NLP, is based on *co-occurrence matrices* that provide insight into word relationships within a collection of texts (Jurafsky and Martin, 2023f).

Vector models are primarily constructed from two types of matrices: *term-document matrices* and *term-term matrices* (Jurafsky and Martin, 2023f). In a term-document matrix, each *row* corresponds to a *word* in the vocabulary, and each *column* represents a *document* within a collection[11]. Each cell in this matrix denotes the number of times a specific word (defined by

[11]Document refers to a text document. For example one purchase contract could be a document in the collection of all purchase contracts investigated in a study.

the row) appears in a particular document (defined by the column). Consequently, a document is represented as a count vector, which is essentially a column in the matrix. Table 1 shows a shortened term-document matrix consisting of four documents and four words.

Table 1: A term-document matrix for four words in four real estate transaction documents. Each cell contains a fictive number of times the word (term) occurs in the document.

		Document			
		Purchase contr. #1	Purchase contr. #2	Donation contr. #1	Donation contr. #2
Term	seller	58	65	2	0
	price	20	40	5	0
	gift	0	3	70	30
	donate	0	2	55	28

Conceptually, a vector is essentially an array of numbers. For example, in Table 1 the document “Purchase contract #1” can be represented as the list of the numbers $[58, 20, 0, 0]$, “Purchase contract #2” can be represented as the list $[65, 40, 3, 2]$ and so on. When considering a collection of documents, a vector space is established, where all the vectors have usually the length of the amount of words in all the documents (which is the vocabulary, $|V|$). The example document vectors (columns) in Table 1 have been shortened to vector dimension of 4 instead of the vocabulary size in order to fit the example on a page and give a vague impression of how vectors in a term-document matrix can look like.

In the context of term-document matrices, each dimension of these vectors corresponds to the frequency of a specific word, allowing for comparisons between dimensions (Jurafsky and Martin, 2023f). For example in Table 1, the vectors for the two documents “Purchase contract #1” and “Purchase contract #2” have similar values (58 and 65, respectively) for the first dimension, representing the word “seller”.

These document vectors can be thought of as points in an $|V|$-dimensional space, where $|V|$ corresponds to the vocabulary size (Jurafsky and Martin, 2023f). To make this concept more easy to comprehend, a 2-dimensional visualization is in Figure 6, with the two dimensions “price” and “gift” as selected terms for the dimensions. The purchase contract documents have high values for the “price” dimension and low values for the “gift” dimension whereas the opposite is the case for the donation contracts.

One use case for term-document matrices is to find similar documents in document classification. In practical scenarios, the term-document matrix is much larger, typically having rows equal to the vocabulary size and columns equal to the number of documents in the collection, which can be very large, consisting of hundreds and thousands of documents (Jurafsky and Martin, 2023f).

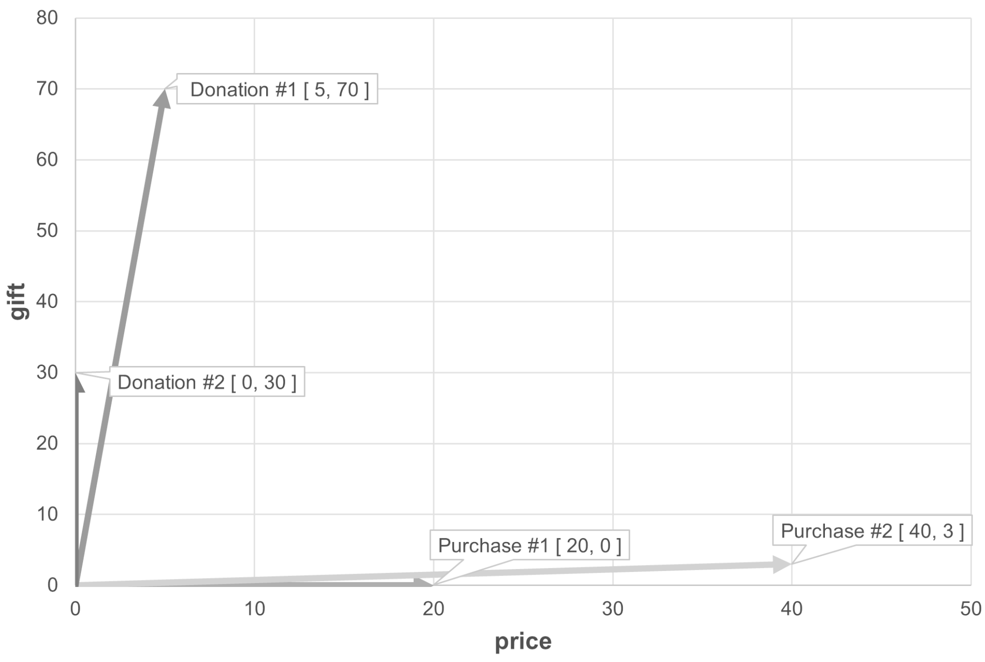


Figure 6: Spatial visualization of four document vectors for two types of documents (purchase vs. donation contracts from Table 1), showing just two of the dimensions, corresponding to the words “price” and “gift”. The figure has been inspired by the illustration of Jurafsky and Martin (2023f, p. 110).

In addition to representing documents, vector semantics can also be used to represent word meanings and detect similar words (Jurafsky and Martin, 2023f). This is achieved by associating each word with a word vector, which is a row vector. Similar words usually have similar vectors because they tend to appear in similar documents. Therefore, the term-document matrix enables the representation of a word’s meaning based on the documents it frequently appears in. For example in Table 1 the first row vector is the vector for “seller” $[58, 65, 2, 0]$ which is fairly similar to the second row vector “price” $[20, 40, 5, 0]$, but less similar to the third one, which is “gift” $[0, 3, 70, 30]$. The precise similarity between two vectors from a term-document matrix can be computed with the cosine of the angle between the vectors[12].

Alternatively, instead of using the term-document matrix to represent words as vectors of document counts, the *term-term matrix*, also known as word-word matrix, can be described. This matrix has dimensions $|V| \times |V|$, with each cell indicating how often a target word and a context word co-occur, which is why those matrices are also called co-occurrence matrices. Contexts can be set as desired and therefore vary but often involve a small window around the word. For instance, a context might be defined as the words within a 4-word window of

[12] As known from Linear Algebra, the cosine similarity metric between two vectors $\mathbf{v}$ and $\mathbf{w}$: $cosine(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|}$. The metric can be understood as a form of normalized dot product.

the target word. For examples on how such matrices can look like, see text books about that topic, e.g. Jurafsky and Martin (2023f, p. 111ff).

Since co-occurrence matrices display raw frequencies and those frequencies can be sewed, methods to weight co-occurrence matrices were invented. One of those method is the TF-IDF weighing.

2.3.1 Exploring Weighted Word Frequencies with TF-IDF

TF-IDF, which stands for *Term Frequency-Inverse Document Frequency*, is a widely used technique in Natural Language Processing to evaluate the importance of words within a collection of documents. It balances the significance of words based on their frequency within a document and their uniqueness across the entire document collection. The TF-IDF weighing is a method to weigh co-occurrence matrices (Jurafsky and Martin, 2023f).

The first component, *Term Frequency* (TF), focuses on how often a specific word t occurs within a single document d[13]. Initially, TF uses the raw count of word occurrences, which is referred to as $\mathrm{tf}_{t,d} = \mathrm{count}(t, d)$ where t represents the word, and d represents the document. However, in practice, it is common to apply a logarithmic transformation, such as the base-10 logarithm, to the raw frequency (Jurafsky and Martin, 2023f). This transformation helps in reducing the influence of words that appear very frequently within a document. For instance, if a word occurs 100 times, applying a logarithm stops it from being considered 100 times more relevant to the document's meaning. To avoid issues with taking the logarithm of zero, a constant (typically 1) is added to the word count. The mathematical notation of the logarithmic transformation, which was retrieved from the text book by Jurafsky and Martin (2023f, p. 114), usually is:

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t, d) + 1)$$

With log weighting, the Term Frequency (TF) becomes more nuanced. For example, if a term occurs 0 times, its TF is 0; if it occurs 10 times, its TF is approximately 1.04, and if it occurs 1000 times, its TF is approximately 3.

The second component, *Inverse Document Frequency* (IDF), addresses the importance of words based on their uniqueness across all the available documents (which is the entire document *collection*). It calculates the *Document Frequency* (DF) of a term, which represents the number of documents in which the term appears. The fewer documents in which a term

[13]Document refers to a text document. For example one Wikipedia entry would be a document in the collection of all Wikipedia articles.

occurs, the higher its IDF weight. Terms that appear in only a few documents are considered discriminative, as they help differentiate those documents from the rest of the collection (Jurafsky and Martin, 2023f).

Given the usually large number of documents in collections, IDF values are typically adjusted using a logarithmic function. The resulting definition for Inverse Document Frequency (IDF) is a logarithmic transformation of the inverse document frequency, which is (according to Jurafsky and Martin (2023f), p. 115) usually expressed in mathematical notation as

$$\text{idf}_t = \log_{10}(\frac{N}{df_t})$$

The TF and IDF are combined to the TF-IDF weighted value for for word t in document d as:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

In summary, TF-IDF is a technique for evaluating word importance within documents. It considers how often a word appears in a document and how unique it is across all the documents. This helps to find a balance between common words that occur frequently and rare, distinctive terms.

This method plays a big role in various NLP tasks, such as text classification, information retrieval, and document ranking (Jurafsky and Martin, 2023f). Nevertheless, TF-IDF calculates document similarity within the word-count space, which can become slow when the documents are rich of vocabulary. Moreover, it operates under the assumption that the counts of various words offer independent indicators of similarity and it does not take into account semantic similarities or relationships between words (Liu et al., 2018).

2.3.2 Embeddings in NLP

The vectors in the co-occurrence matrices, no matter if weighted with TF-IDF or not, are very long, because of having the same size as the vocabulary $|V|$ or amount of documents D in the collection, but also sparse, having a lot of zeros. There is an alternative representation of words as short and dense vectors, which are called *embeddings*. In contrast to vectors from co-occurrence matrices, the vectors of embeddings have dimension d of 50-1000 and not full vocabulary size (Jurafsky and Martin, 2023f). The values in embeddings are real-valued, can be negative and usually do not have many zeros. An advantage of using shorter and more dense vectors to represent words is that the neural network performing a task like classification, needs to learn fewer parameters during training in comparison to a network processing longer vectors. Moreover, dense vectors are assumed to capture better which words are synonyms because in sparse vectors the dimensions of synonyms are distinct and not related but in dense vectors this might not be the case (Jurafsky and Martin, 2023f).

2.3.3 Exploring word2vec as an Embedding Technique

Word2vec is a common term occurring in the context of NLP. It describes a set of algorithms and model architectures to learn word embeddings efficiently from large documents.

The *skip-gram* algorithm (Mikolov et al., 2013), which is the most famous word2vec model, is able to generate dense embeddings for words. It uses a logistic regression classifier to determine the likelihood that two words will occur close to each other in text, calculated through the dot product between their embeddings. Basically, the skip-gram model trains a probabilistic classifier that assigns probabilities based on the similarity of a context window to a target word (Jurafsky and Martin, 2023f).

The training objective of the skip-gram algorithm is to learn suitable word embeddings by training a probabilistic classifier. Even though there is a classification task involved in training the parameters, the skip-gram algorithm does not mainly focus on predicting the correct class. Its overall goal is to learn the correct weights through the classification task. The learned weights are then the resulting word embeddings (Jurafsky and Martin, 2023f). Therefore, the model is trained on a large corpus of text[14] and has the task of predicting the context words given a target word (or vice versa).

For this purpose, the model uses *positive and negative examples.* For each training instance, the algorithm considers a pair of a target word w and a neighboring context word c as a positive example: (w, c). For instance, a sentence representing a context window of $L = 4$ words (± 2 words context) could be:

...	high	quality	*oak*	flooring	in ...
	c1	c2	w	c3	c4

Considering this example, if the target word w is "oak" and the context word is "flooring" this pair is a positive example. The probabilistic classifier is trained in a way such that it returns the probability that the context word (e.g. c is *flooring*) is a real context word of the target word w: $P(+|w, c)$. The probability of a word not being a context word is $1 - P(+|w, c)$.

To train a binary classifier, negative examples are also required. Skip-gram generates these negative examples by combining the target word w with so called *noise words* that are selected

[14]A collection of linguistic data (texts) is called *corpus* (Li et al., 2021).

randomly from the vocabulary. Noise words are words that are not genuinely in the context of the target word. In the example of “oak” being the target word, a noise word could be “transfer”. The probability is obtained as the logistic function of the dot product of the dense vectors for w and c. For further equations please see text books like e.g. Jurafsky and Martin (2023f, p. 122ff).

Skip-gram represents each word using two embeddings: the target embedding (input embedding) and the context embedding (output embedding). These embeddings are stored in matrices, usually denoted as $\mathbf{W}$ (for target embeddings) and $\mathbf{C}$ (for context embeddings)[15]. One can represent a word by combining its target and context embeddings, such as $w_i + c_i$, or simply use the target embedding (w_i) alone.

The initialization of the skip-gram algorithm involves assigning random numbers to the embedding vectors of all the words in the vocabulary, During training, the embeddings of each word w is adapted so the vectors are shifted to be more similar to words that occur nearby a target word. The training algorithm iteratively adjusts the embeddings to achieve two objectives: Maximize the similarity between target and context word pairs drawn from the positive examples and minimize the similarity between the target word and noise words from the negative examples.

To achieve these objectives, skip-gram uses gradient descent optimization (which was explained before in Section 2.1). It starts with randomly initialized matrices $\mathbf{W}$ and $\mathbf{C}$ and then iterates through the training corpus. During each iteration, the algorithm calculates the gradients of the loss function and updates the embeddings accordingly to move them closer to the desired values. An intuition of this training and embedding adaption is given in Figure 7.

As for the TF-IDF, also for the skip-gram embeddings the size of the context window matters. Experiments often include tuning this parameter to find the optimal window size for a specific task or corpus. Overall, the skip-gram algorithm, which is sometimes just called word2vec like the umbrella term, is a technique for training word embeddings by treating the relationship between target and context words as a classification problem. It uses gradient descent optimization to adjust embeddings to maximize similarity with actual context words and minimize similarity with noise words. By learning these embeddings, it captures the semantic relationships between words, making them useful in various Natural Language Processing tasks.

[15]For further details on the matrix shape and content please see text books like e.g. Jurafsky and Martin (2023f, p. 122ff).

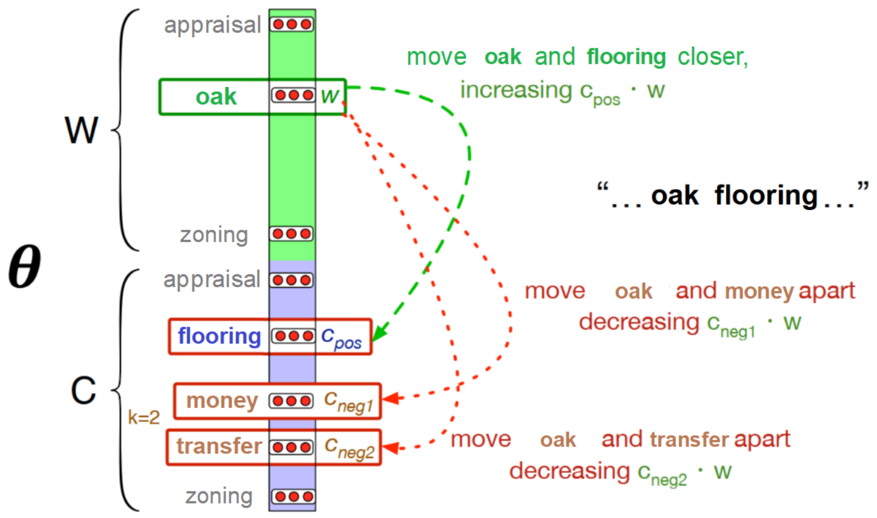

W
θ
C
k=2
appraisal
oak
w
zoning
appraisal
flooring
c_{pos}
money
c_{neg1}
transfer
c_{neg2}
zoning
move oak and flooring closer,
increasing $c_{pos} \cdot w$
"... oak flooring ..."
move oak and money apart
decreasing $c_{neg1} \cdot w$
move oak and transfer apart
decreasing $c_{neg2} \cdot w$

Figure 7: Schematic illustration of a single iteration in gradient descent in the skip-gram model: The goal is to adjust embeddings in such a way that the embeddings associated with the target word "oak" move closer to (have stronger dot product with) embeddings of nearby context words, like "flooring". Here, "oak" is supposed to simultaneously move farther away from (have weaker dot product with) embeddings of context words that are unrelated and distant (such as "money" and "transfer"). The figure is a modification of an illustration by Jurafsky and Martin (2023f, p. 124).

2.4 Feed Forward Neural Language Models

Manually creating representations of characteristics of the data is called *feature engineering*. Features are the characteristics of the data that help the models applied to the data obtain the desired output (Sharma, 2020).

With neural networks it is usually avoided to create features by hand (like previously done for sentiment classification in the example in Figure 5), but rather to let the model learn relevant features on its own. In the word2vec model, for example, the model learns the embeddings through the context words, so the features are learned from the data rather than defined by a human directly. Also the schematic example network from Figure 5 can be adapted to use embeddings instead of hand-crafted features since there are many different ways to represent input text. Jurafsky and Martin (2023g) describe that it is possible to apply a pooling function (not further specified) to the embeddings of words in the input. This means for a text with n words (as input) the embeddings are created as $\mathbf{e}(w_1), ..., \mathbf{e}(w_n)$, where each embedding $\mathbf{e}(w_i)$ has dimensionality d, achieved by taking the mean of the vectors. The input to the feed forward network is then, instead of a handcrafted $\mathbf{x}$-vector, the vector representing the means of the word embeddings: $\mathbf{x} = mean(\mathbf{e}(w_1), \mathbf{e}(w_2), ..., \mathbf{e}(w_n))$. The other equations of such a neural network would remain the same as described on page 14. Basically, the model architecture just described can be imagined to be adapted in such a way that the representation of the words as embeddings and the pooling process is *before* the rest of the feed forward network shown previously in Figure 5[16].

The feed forward networks considered so far contained examples for text classification. Another common task in NLP is next word prediction. Below, a model capable of this task is explained with the corresponding architecture, including embeddings.

A feed forward neural *language model* (LM) is a type of artificial neural network designed for natural language processing tasks. In this model, at each time step t, it takes as input a representation of a sequence of preceding words, typically including words from time $t-1$, $t-2$, and so on. The key objective of this model is to predict the likelihood of the next word that should follow the input sequence (Jurafsky and Martin, 2023g).

The network processes the input information through a series of hidden layers, which may involve various nonlinear transformations, and eventually produces an output layer that represents a probability distribution over a set of potential next words. In other words, it estimates the probabilities of various words that are most likely to follow in the given context (Jurafsky and Martin, 2023g). The approximation of the probability of a word given the prior context is $P(w_t|w_{1:t-1})$.

Moreover, the words in language models are not represented as words directly, but rather as embeddings (see Section 2.3.2 above). The advantage of using embeddings instead of words for a next word prediction task can be illustrated with an example:

Imagine the sentence *"I have to make sure that the* ***window*** *gets closed"* has been used, among other sentences, for training word embeddings. During training, the "gets closed" has been seen after "window", but not after e.g. "door". Due to example sentences in a fictional training set that allows the model to learn the similarity between the two words "window" and "door", the embeddings are similar. This implies that they have a relatively small distance between their higher-dimensional, dense embedding vectors. Hence, a language model that uses the learned embeddings is able to generalize from the "window" to the "door" and would be able to comprehend that the sentence *"I have to make sure that the* ***door*** *... "* should end with *"gets closed"* through assigning high enough probabilities. This is a rather simple example of what *generalization* in language models is but illustrates very much that embeddings are essential for text understanding[17].

[16]The Figure including pooled embedding is printed in the text book of Jurafsky and Martin (2023g) on p. 146. Refer to this text book chapter for further details.

[17]Due to the capability of language models to generalize form seen information to new information the models are sometimes assigned the capability to *understand* natural language.

Forward inference, in the context of a neural language modelling, involves executing a forward pass in the neural network. An input is given, the network is generating a probability distribution regarding potential outputs and specifically, the subsequent words. To understand a forward inference in a feed forward neural language model, an example (based on a similar example by Jurafsky and Martin (2023g)) is given among the next paragraphs. Therefore, another example sentence, "spyhole in the wooden entrance **door**", is used. In this example sentence the word "door" should be predicted based on the preceding words.

To initiate the process, each of the previous N words is represented as a *one-hot vector*, which is a vector that has all elements set to zero except one element that is set to 1. The one-hot-vector has length $|V|$ (one element for each word in the vocabulary) and the one element that is set to 1, corresponds to the word's index in the vocabulary. For example, in the one-hot representation for the word "entrance" (indexed as V_4), the element $x_4 = 1$ and all the other x_i would be set to 0. This looks like a sparse vector:

$$\begin{array}{cccccccccc} [\,0 & 0 & 0 & 1 & 0 & 0 & ... & 0\;0 & 0\,] \\ 1 & 2 & 3 & 4 & 5 & 6 & ... & ... & |V| \end{array}$$

In the feed forward neural language model including embeddings (like in Figure 8), a window captures the previous N words of the input text. In case of e.g., $N = 3$ the previous words are denoted as w_{t-1}, w_{t-2}, and w_{t-3}, where each word is represented as a one-hot vector. One-hot encoders for $N = 3$ previous words in the sentence "spyhole in the wooden entrance **door**", where "door" is the target word, would have a "1" at the respective vector position for the words *the*, *wooden*, and *entrance*.

In order to select the correct embedding vector for a word for further processing, an embedding matrix is multiplied with the one-hot vector with the 1 at the index N. The embedding matrix $\mathbf{E}$ is a matrix with the dimensions $d \times |V|$, containing column vectors for each word (dimension of column vectors is d). The matrix $\mathbf{E}$ is a weight matrix. This multiplication process with a one-hot vector selects the specific column vector for the corresponding word, resulting in the word's embedding (Jurafsky and Martin, 2023g).

An embedding layer can be created with concatenating embedding vectors, like e.g. in case of $N = 3$ preceding words, 3 embeddings vectors would be concatenated to $\mathbf{e}$. As also shown in the smaller feed forward networks previously, after the input follows a hidden layer, and in the end an output layer with softmax to produce probabilities. In this example, there are several output nodes. Every output node shows the determined probability of the next word w_t being exactly this word. For example, $\hat{y}_{41}$, the output node corresponding to the vocabulary word $V_i = V_{41}$, indicates the probability of the next word w_t being 41^{st} word in the vocabulary. In our case V_{41} is "door", which should get the highest output probability in our example since the example sentence used in our example (see Figure 8) was constructed like that.

The algorithm for the described example that is shown in Figure 8 follows these main steps:

1. **Select N embeddings from the embedding matrix**: Because $N = 3$, the three prior word embeddings are retrieved by creating their respective one-hot vectors and multiplying each by the embedding matrix $\mathbf{E}$. This process results in the concatenation of these embeddings to form the embedding layer $\mathbf{E}$.
2. **Multiply by the weight matrix**: The resulting layer is multiplied by $\mathbf{W}$ and, after adding $\mathbf{b}$, goes through an activation function to produce the hidden layer h.
3. **Multiply by U**: The hidden layer h undergoes multiplication by another weight matrix that is denoted as $\mathbf{U}$.
4. **Softmax**: The output layer uses softmax to estimate the probability of each node i representing the next word: $P(w_t = i|w_{t-1}, w_{-2}, w_{t-3})$.

Overall, the equations for a neural language model with a window size of $N = 3$, and using one-hot vectors for each context word, can be represented as (retrieved by Jurafsky and Martin (2023g), p. 149):

$$\mathbf{e} = [\mathbf{Ex_{t-3}}; \mathbf{Ex_{t-2}}; \mathbf{Ex_{t-1}}] \qquad \text{(N = 3 prev. words considered here, ; indicates vector concatenation)}$$

$$\mathbf{h} = \sigma(\mathbf{We} + \mathbf{b})$$

$$\mathbf{z} = \mathbf{Uh}$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$$

It can be observed that the equations are still rather similar to the ones given previously on page 14 in the context of a 2-layer feed forward network. Just the x changed from the previously hand-engineered feature vector to the embedding layer $\mathbf{e}$.

Figure 8 illustrates the forward inference in a feed forward neural language model processing our example sentence "spyhole in the wooden entrance **door**". At each time step t, the model generates a d-dimensional embedding for each context word by multiplying a one-hot vector with the embedding matrix $\mathbf{E}$. These resulting embeddings are concatenated to form the embedding layer $\mathbf{E}$. Subsequently, $\mathbf{E}$ is multiplied by weight matrix $\mathbf{W}$, activating an element-wise activation function, leading to the formation of the hidden layer h. This hidden layer is further multiplied by another weight matrix $\mathbf{U}$. Finally, the output layer, employing softmax, predicts the probability of the next word w_t being the vocabulary word V_i at each node i.

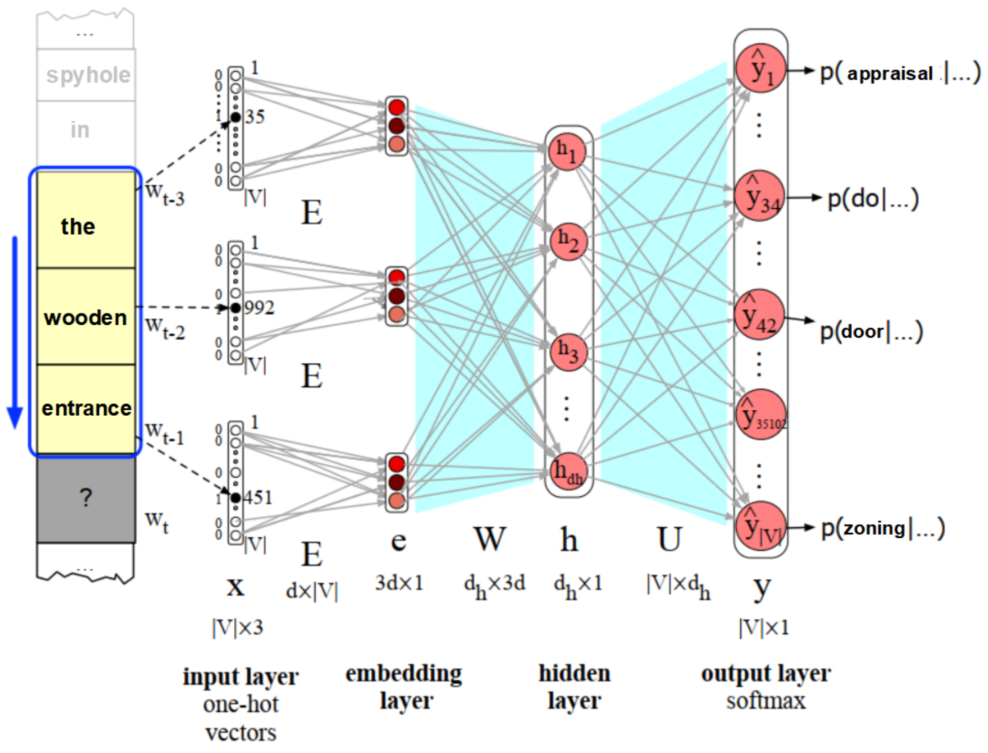


Figure 8: Schematic illustration of a neural feed forward language model. The embeddings are obtained via multiplication of one-hot vectors end the embedding matrix **E**. Three embeddings are concatenated to the embedding layer **e**, which serves as further input to the feed forward network layers. The figure is a modification of an illustration by Jurafsky and Martin (2023f, p. 149).

Training

After setting up an architecture for a neural language model, the model can be trained via adjusting its parameters to predict upcoming words. These parameters are in text books often denoted as θ. In case of a neural feed forward network include the embedding matrix $\mathbf{E}$, weight matrices $\mathbf{W}$ and $\mathbf{U}$, and biases $\mathbf{b}$.

In some instances, freezing the embedding layer $\mathbf{E}$ with pre-computed word2vec or similar pre-training values is possible. Freezing refers to keeping the embedding matrix $\mathbf{E}$ constant while modifying the other parameters $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ during language model training. However, it depends on the task if it is beneficial to simultaneously learn the embeddings along with training the entire network or not.

The complete training process, including updating all parameters $\theta = E, W, U, b$, uses gradient descent optimization via error backpropagation (see Section 2.1). This training phase not only adjusts the weights $\mathbf{W}$ and $\mathbf{U}$ but also refines the embeddings $\mathbf{E}$ for each word to best predict upcoming words (if the embedding matrix is not frozen).

During training, text is taken as input, and the process begins with random weights. During training, the model is moving with a window through the text, and predicts each word w_t. At each word w_t, the loss can be applied. The parameter update for gradient descent is applied to minimize this loss, iterating from step s to $s + 1$. This gradient is computed across the parameters $\theta = E, W, U, b$. For further details on the gradients see textbooks like e.g. Jurafsky and Martin (2023g, p. 151ff).

By minimizing the error, the model is not only a word predictor but also creates a new set of embeddings $\mathbf{E}$. Since $\mathbf{E}$ are learned word representations, those can also be used for other tasks (Jurafsky and Martin, 2023g).

2.5 Note on Various Neural Network Architectures

An architecture that is common in neural networks that process languages is the *transformer architecture* (more on that in Section 2.7) whereas a common architecture for image processing is the convolutional neural network (CNN). The latter are organized grid-like because images have pixels and the way to process the images is inspired by the human retina which can focus specific parts of an image at a time. For further information about how images are processed by neural networks see text books (e.g. Kruse et al., 2016, p. 88f) or blog posts (e.g. Mishra, 2020). In order to describe how the *transformer architecture* works, a few more terms and network types still need to be covered beforehand.

Networks processing textual data often use *embeddings*, which are vectors representing words. In Section 2.3.2, embeddings created by an word2vec algorithm have been explained. LLMs based on the transformer architecture (see Section 2.7) usually use contextual embeddings instead of static embeddings.

In the word2vec algorithms words so far are mapped to a *fixed* vector and the resulting embeddings are therefore also called *static* embeddings. In contrast to static embeddings, there are *contextual embeddings* which serve as dynamic representations of word meanings and adapt to the specific context in which a word appears (Jurafsky and Martin, 2023c). While *static* embeddings capture the general meaning of word types (entries) in the vocabulary, *contextual* embeddings describe the meaning of individual word *tokens*. A word *token* is a word and the total number of word tokens is the total number of words. On the other hand a *word type* is a distinct word in a text corpus.[18] The number of word types in a corpus represents the vocabulary size. The capturing of word tokens enables contextual embeddings to consider instances of a word type within their unique contexts. This dynamic behaviour allows contextual embeddings to perform good in tasks that make semantic similarity between two words within specific contexts necessary. With contextual embeddings the inability of word2vec to represent a word that has the same spelling but a different meaning is overcome (Mei, 2020).

A collection of linguistic data is also called *corpus* (Li et al., 2021). An un-annotated corpus is simply raw, plain text whereas an annotated corpus has implicit information marked with explicit annotations[19]. Especially in pre-training of transformer models (see Section 2.7.6),

[18]An example for a word with meaning depending on the context is *gift*: a) a present, b) a poison. The word *gift* is a word type in the vocabulary, whereas *gift* and *gift* are two tokens.

[19]Annotation is the process of making implicit linguistic information explicit. Since annotated corpora are not used in this research in any way, the term is not described further. For additional information about annotation techniques and utilization of annotated corpora see further resources (like e.g. Marcus et al., 1993; Musi et al., 2018; Pustejovsky and Stubbs, 2012).

corpora consisting of plain text are needed to learn a language representation (Jurafsky and Martin, 2023b).

Deep learning is a type of machine learning that uses large artificial neural networks to learn and make predictions from large data sets. ANNs are usually referred to as Deep Neural Networks (DNNs) whenever they have more than one hidden layer, even though there is no strict border for the amount of parameters when a neural network is called "deep" (jaygala260 Community Member, 2022). The models are characterized as being networks with many hidden layers, many parameters and therefore being deep (Jurafsky and Martin, 2023g).

Deep learning has the aim of avoiding that humans have to formally specify all knowledge the computer needs for task accomplishment (Goodfellow et al., 2016), which is the opposite approach compared to a approaches where humans generate the features and rules manually (like we saw e.g. in Figure 5 above). DNNs are famous for achieving good results in tasks like image recognition and language understanding.

2.6 Text Classification and Information Extraction

Text classification is the process of categorizing text data into predefined classes based on extracted features (Li et al., 2021). Automatic text classification serves the purpose of organizing the text into pre-defined categories based on features from the text. Automatic text processing including text classification is necessary for organization of large amounts of textual data, but is also part of many tasks like e.g. information extraction (Yang et al., 2023).

Two common example tasks for text classification with two or three possible outcome classes are *sentiment analysis*, where a positive or negative sentiment is assigned to a text piece like a customer review (simple example ANN for this task was given in Section 2.2), and *spam filtering*, where the application in e-mail software helps users to detect spam messages vs. not-spam messages (Jurafsky and Martin, 2023h). Sentiment analysis as well as spam detection are so called *sequence classification* tasks, where an entire text sequence is assigned to a category. A sequence classification task with more than three possible categories is *topic classification*, where the overall topic of the text sequence is selected from several category options. The text sequence in topic classification can be consisting of a few words up to a complete document (Jurafsky and Martin, 2023h; Wang and Manning, 2012).

In the beginning of text processing, *regular expressions* (regex) play a huge role in pre-processing and text normalization. Regex is a standardized language for specifying text search strings (Jurafsky and Martin, 2023a). It can be used for text normalization, which is the task of transforming text into a more convenient, harmonized form. A simple normalization task would be to standardize spacing (e.g. two or more spaces are transformed to one space). Further, regex can be used for normalization tasks like tokenization (separating out words from running text), lemmatization (mapping words to their root) and sentence segmentation (breaking up text into individual sentences).

Besides that, regex can also be used to specify strings that should be extracted from a piece of text (Jurafsky and Martin, 2023a) or for text classification (Liu et al., 2020). Regex is a popular technique providing an interpretable solution in text classification tasks using matching of predefined patterns (Liu et al., 2020). Due to the predefined patterns and rules, information extraction and text classification with regex can also be called *rule-based approach.*

Apart from rule-based classification, there are many more methods to approach the task of classification. A quite similar kind of classification can be conducted with several words describing a class or occurring in the context of a class. This approach is commonly called *dictionary-based* classification and conducted with a dictionary for every potential outcome class (Abel and Lantow, 2019). Like the rule-based classification, the dictionary-based classification is based on manually defined patterns and words. Therefore, one could subsume the dictionary-based classification as rule-based classification approach as well (Abel and Lantow, 2019). To implement rule-based classification, it is essential to have knowledge about the data and the domain. This understanding helps capture the specific characteristics of the text data within the respective classes (Oznalbant, 2023).

Although a rule-based approach can be employed for text classification, there are also language models developed specifically for text classification tasks. While rule-based and dictionary-based approaches for text classification depend on experts' domain knowledge and rules derived from it, deep learning models use the observed text data and generalizations for classification. Deep learning models require a large amount of text for training to effectively perform on unseen text (Li et al., 2021; Zhao et al., 2022). Deep learning models are trained on a large corpus and therefore have the power to generalize from the learned text to new text data, which is a process called *transfer learning* (Jurafsky and Martin, 2023c). However, it should be noted that deep learning models *need to be* trained on a large amount of data in order to estimate the parameters reliably because they have a large number of parameters. Small amounts of data would not suffice to train the parameters well enough.

Until the 2010s, traditional text classification primarily involved statistics-based algorithms, such as Naive Bayes (Maron, 1961; Pang et al., 2002), K-Nearest Neighbor (Cover and Hart, 1967), Random Forests (Breiman, 2001), and Support Vector Machines (Joachims, 1998). While accurate and stable, these methods require time-consuming feature engineering and lack consideration of the natural *sequential structure* in text data, making it challenging to capture semantic information (Li et al., 2021). In order to roughly understand how a traditional text classification method can be applied, Naive Bayes is explained below. For further information on the other methods mentioned, please refer to textbooks.

Note on Naive Bayes Algorithm

Ignoring the sequential structure of sentences basically means that the word order is ignored. For example *Naive Bayes* can be applied in text classification via using *Bag of Words* (BoW) as text representation. A bag of words can be imaged like a shopping bag containing various words in an unordered and unstructured way. When taking out the words like groceries from a bag, the count of the words is noted. Hence, the frequency of the words in a text is known, disregarding any order information. This methods makes it possible to count the number of words within contexts and calculate probabilities for words regarding the context (Jurafsky and Martin, 2023e).

For instance, when dealing with real estate descriptions, consider the phrases “A single-family house with a garden” and “A renovated house with swimming pool”. In the BoW representation, one simply counts the occurrences of each word, irrespective of their placement. Thus, the word “house” appears twice, “garden” once, and so forth.

The Naive Bayes classifier, is a probabilistic tool for document classification. Given a document d, it aims to determine the most probable class from a set of possible classes $c \in C$. The classifier returns $\hat{c}$, the estimate of the correct class, which exhibits the highest posterior probability given d:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

To keep the computational complexity manageable, the Naive Bayes classifier relies on two simplifying assumptions. Firstly, it uses the Bag of Words approach, implying that the position of words within the text does not influence the classification outcome. In the context of real estate descriptions, this means that whether the term “garden” appears at the beginning or end of a description, it carries the same importance in classification.

Secondly, the Naive Bayes classifier relies on the *Naive Bayes assumption*, which means that the probabilities of individual features (in this case, words) occurring within a document are independent, given we know the class c. The individual features $f_1, f_2, ..., f_n$ just encode the word identity, not the order. This assumption simplifies the calculations by allowing the probabilities $P(f_i|c)$ to be treated as independent entities, which can be “naively” multiplied together. For more information on Naive Bayes Classification see text books like e.g. Jurafsky and Martin (2023e).

Since the 2010s, the focus has shifted from traditional methods like the Naive Bayes towards deep learning models, particularly with the increasing popularity of transformer models (which are DNNs) after 2018 (Li et al., 2021). DNNs are important methods in the current text processing and classification research, offering automatic and semantically meaningful representations without the need for manual rule design. Large Language Models (LLM) are trained deep learning models with the purpose to process textual data produced by humans. The usage of contextual word embeddings (see Section 2.5) have significantly contributed to the field’s improvements. LLMs are assumed to understand and generate text with close to human performance (Brownlee, 2023).

Rule-based or dictionary-based approaches are typically easier to implement and interpret in comparison to machine learning or deep learning approaches, as they rely on explicit rules or lists of keywords for feature or attribute identification in text. They can be especially effective in situations where the text data is highly structured and/or consistent since they do not transfer any knowledge as deep learning models could do. Another advantage of rule-based methods is that they do not require large amounts of labeled data to perform well – but a person with domain knowledge needs to write down the rules.

In summary, text classification has developed over the decades from traditional statistics-based methods, via (less deep) neural methods to deep learning models, with LLM playing an important role in current research.

2.7 Large Language Models: The Transformer Architecture

Different types of language models can be distinguished in the field of NLP[20]. One of them are *pre-trained language models*, which can also be called *Large Language Models* (LLMs) (Wei and Zhou, 2023). While the LLMs have a different amount of layers and parameters, they all have a *transformer* architecture (Vaswani et al., 2017), which is the most common language modeling architecture (Jurafsky and Martin, 2023b) and is further explained in this section. Differences in the architecture of models used in this thesis will further be covered later (see Section 5.2).

2.7.1 Overview of the Transformer Model

Basically, transformers are performing a mapping of sequences of input and output vectors both having the same or a different length via an internal representation of the input (Jurafsky and Martin, 2023b,h). Transformers consist of several components and structures that form a deep neural network with many layers and include:

1. Encoder: This part takes in an input sequence, such as a sentence in one language. It processes each word in the sentence one by one, transforming the words into numerical representations. These vectors contain information about the meaning and context of the words in the sentence. This part of the model is specialized to create internal representations of the input.

2. Self-Attention Mechanism: The special characteristic why the transformer architecture is so powerful is the self-attention mechanism. It enables the model to capture the relationships between different words in a sentence. It works by allowing each word to “pay attention” to all the other words in the sentence. This helps detecting dependencies and relationships between words, understanding which words are more important for understanding a particular word.

[20]A recently published survey on language models history and current impact on science can be recommended for further details of models not covered in this thesis. Please see Zhao et al. (2023).

3. Multi-Head Attention: The self-attention mechanism is performed multiple times in parallel, each with a different so called *head* or perspective. This helps the model to learn different types of relationships or features in the text.

4. Positional Encoding: Transformers do not have a built-in sense of word order, so they use *positional encoding* to give the model information about the order of words in a sentence.

5. Decoder: Once the input sequence is processed by the encoder, the decoder takes the encoded information and generates an output sequence. For example, in translation tasks, it takes the encoded meaning of the input sentence and produces the translated words in another language. This part of the model is optimized to create the desired output.

Self-attention and positional encoding are important components found in both the encoder and decoder of transformer architectures (Kortschak, 2020). In the decoder, self-attention enables each position to focus on various positions in the encoded input sequence. This helps the model consider the relevant context for each step of output generation. In both the encoder and decoder, positional encoding is added to the input embeddings to help the model distinguish between the positions of different elements in the sequence.

To gain a deeper understanding of transformer models, the following subsections go into more detail regarding the functioning of these elementary components.

2.7.2 Self-Attention

The *self-attention* mechanism computes the interaction among words in a sequence. It determines the strength of relationships between words in an input sequence, generating a score that shows the attention the network should pay to a word concerning other elements in the sequence.

In the original transformer architecture by Vaswani et al. (2017), the self-attention mechanism works by creating three vectors for each word in a sentence: Query (**Q**), Key (**K**), and Value (**V**). These vectors are derived from the input embeddings of the words.

For a sequence of words (or tokens) in a sentence, one can denote the embeddings of the words as $X = [x_1, x_2, ..., x_n]$, where n is the length of the sequence.

The Query, Key, and Value vectors for each word are computed through linear transformations of the input embeddings:

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

Here, W^Q, W^K, and W^V are weight matrices used to project the input embeddings into the Query, Key, and Value spaces. Once the **Q**, **K**, and **V** vectors are obtained, the self-attention mechanism calculates attention scores between each word in the sequence.

The attention score A_{ij} between the i^{th} word (or token) and the j^{th} word is computed as follows:

$$A_{ij} = \text{softmax}\left(\frac{Q_i \cdot K_j}{\sqrt{d_k}}\right)$$

Here, Q_i and K_j represent the Query and Key vectors for the i^{th} and j^{th} words, and d_k is the dimension of the Key vectors. The softmax function normalizes the scores to represent the attention weights for word i attending to word j. Each word's vector interacts with all the other word vectors to find how much attention it should pay to each word. The attention is calculated by comparing the similarity between each word pair.

The weighted sum of the Value vectors according to these attention scores produces the output of the self-attention mechanism for each word:

$$\text{SelfAttention}_i = \sum_{j=1}^{n} A_{ij} \cdot V_j$$

This operation is applied independently to all words in the sequence, allowing each word to gather information from all other words based on their importance (determined by the attention scores).

In the transformer model, this self-attention operation is typically performed in parallel for all words in the sequence and across multiple heads to capture different aspects of relationships and contexts. This process helps the model understand the importance and relationships between different words in a sentence. It allows the model to learn which words are more relevant in the context of other words.

These weighted representations obtained from self-attention contribute to the contextual understanding of the words in the sequence and are implemented also in the later layers of the transformer model for different downstream tasks[21].

The self-attention layers of transformer models can be either be *left-to-right oriented*, which means they do not regard future text, or *bidirectional*, which means they regard text from before and after the current value. An illustration is given in Figure 9 where the self-attention layer that maps the input sequence $(x_1, ..., x_n)$ to the output sequence $(y_1, ..., y_n)$ is described for sequence length $n = 4$.

[21] A downstream task is the task the model should perform in the end. This depends on the use case. Examples of downstream tasks in LLMs: text translation, classification, or generation.

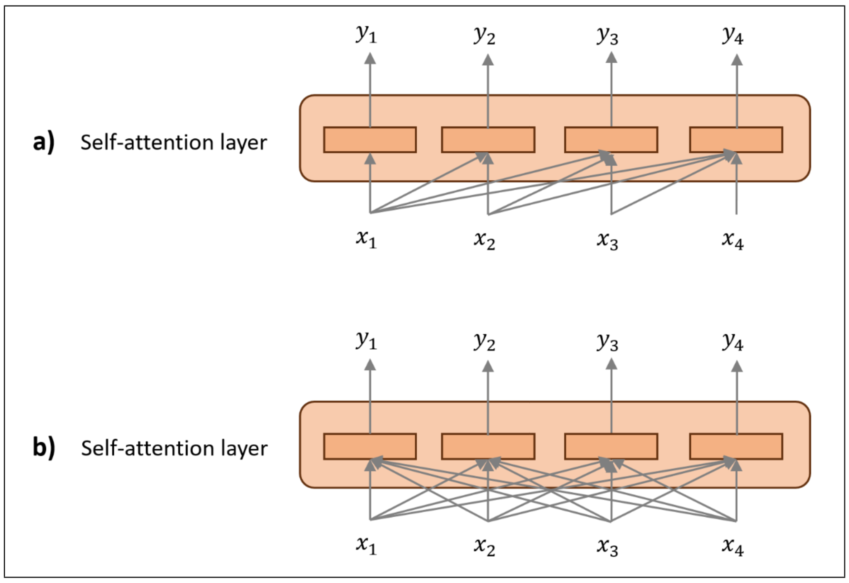


Figure 9: Self-attention layer information workflow with a) solely left-to-right approach and b) with bidirectional approach. In a) the attention is based solely on current and earlier input whereas in b) both directions before and after the current input are incorporated. The illustration can be found in Jurafsky and Martin (2023c, p. 229) and has been redrawn with a shorter input vector.

Transformer models have several layers of self-attention, and the original transformer paper describe bidirectional self-attention in the encoder and left-to-right oriented self-attention in the encoder (Vaswani et al., 2017).

The self-attention method is called in this case *multi-headed* because the self-attention for a sequence is executed multiple times with different weights which results in multiple vectors, called heads, that are combined in the end (Kortschak, 2020; Vaswani et al., 2017).

2.7.3 Positional Encoding

Whenever any model takes a sequence of text as input, the order of the words matters. When words are scrambled up, the text can get a different meaning or become meaningless. Therefore, positional information about words needs to be maintained (Kazemnejad, 2019).

If there was no additional, preceding component that maintains word order in the many hidden layers transformer models have, they would not be able to keep word order in the self-attention layers because they lack a built-in awareness of the relative or absolute positions of input tokens (Jurafsky and Martin, 2023b). So called *positional embedding* is used to forward word position information to the middle layers of the model (Kazemnejad, 2019; Jurafsky and Martin, 2023b).

One straightforward approach to process position information involves adjusting the input embeddings of the model by incorporating positional embeddings tailored to each position in the input sequence. The approach begins by generating randomly initialized embeddings corresponding to all possible input positions, up to a predefined maximum length (Jurafsky and Martin, 2023b). Like word embeddings, these positional embeddings are trainable parameters adjusted during the training process. The creation of an input embedding incorporating positional information involves adding the word embedding for each input to its corresponding positional embedding. Jurafsky and Martin (2023b) describe in their textbook about NLP that these embeddings are not concatenated – rather, they are added to produce a modified embedding vector of the same dimensionality. Figure 10 illustrates this concept. The traffic-light-like symbols represent the embedding of each token in the sequence of tokens “apartment is sold to person”. Here the embeddings are called *composite embeddings* because the consist of the input word embeddings as well as their corresponding position embeddings.

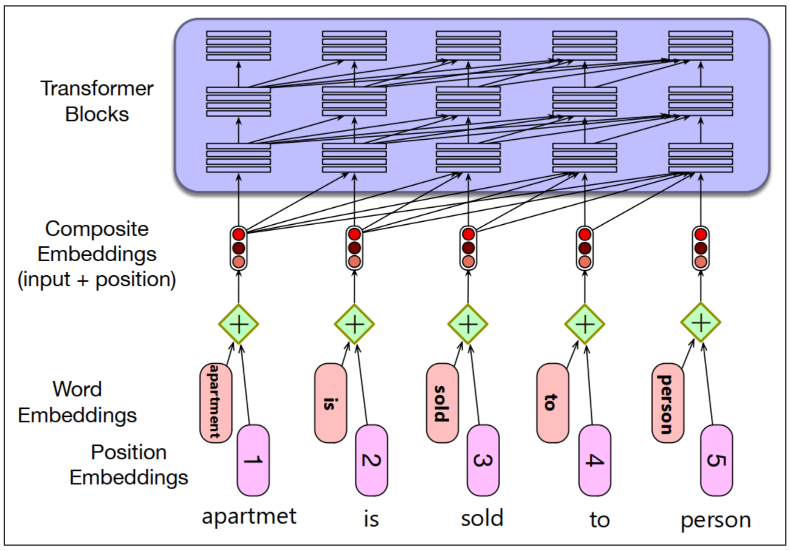


Figure 10: Illustration of positional embeddings. The original illustration can be found in Jurafsky and Martin (2023h, p. 207) and has been modified to have an input text sequence related to real estate.

However, a drawback of this approach to maintain word order is that the uneven distribution of training examples, with initial positions having more instances (e.g. because every sentence starts with the first word, but not every sentence has as many words as the other one) than positions near the length limits[22]. Consequently, embeddings for the latter positions may be inadequately trained, limiting generalization during testing and using of the transformer model

[22]Transformer models have length limits, which means they cannot process infinitely long input text. The limit varies across the models. For an example regarding one LLM that has a length out of 512 words, reasons for that and solutions see e.g. McQuillan (2021).

(Kazemnejad, 2019). An alternative approach to positional embeddings involves selecting a static function that maps integer inputs to real-valued vectors, capturing relationships among positions. For instance, recognizing that position 2 is more closely related to position 3 than to position 15. The original transformer paper (Vaswani et al., 2017) employed a combination of sine and cosine functions with varying frequencies for this purpose. The transformer based model preserves word order using a d-dimensional vector, containing information about a specific position in a text sequence. Also in the original paper, the positional embeddings are summed with the word embeddings (and not concatenated). This approach has the advantage of reducing parameters (Kazemnejad, 2019). For more details and proofs on positional embeddings, refer to Vaswani et al. (2017) or blog posts going into details, like Kazemnejad (2019).

2.7.4 Encoder-Decoder

Encoder and decoder are connected with each other and have a context vector in between (see Figure 11). The encoder processes an input sequence x_1^n and generates an internal representation h_1^n of the input sequence. The context vector c is a function of the encoders internal representation (hidden states). It is a concatenation of the hidden states with a linear transformation applied, and contains the essence of the input. The decoder receives c as input, generates sequences of hidden states h_1^m and a sequence of output states y_1^m in the desired format (Jurafsky and Martin, 2023h).

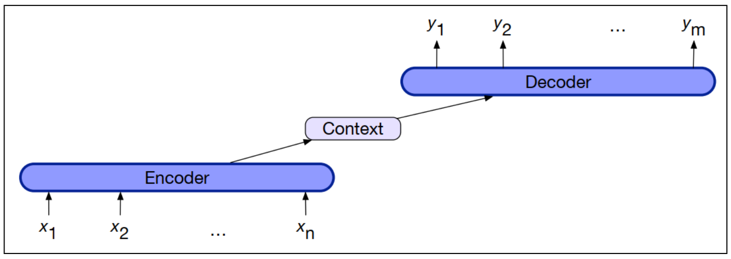


Figure 11: Illustration of encoder, decoder, and the representation of the input. The illustration has been retrieved from Jurafsky and Martin (2023h, p. 202).

Both the encoder and decoder each consist of at least one *transformer block* (Jurafsky and Martin, 2023b). A standard transformer block, like displayed in Figure 12, in general consists of two sub-layers: a self-attention layer and a fully-connected feed forward layer with layer normalization after each of these sub-layers (Jurafsky and Martin, 2023b,d). Feed forward networks were already described in detail in Section 2.1 and the self-attention concept was described in Section 2.7.2.

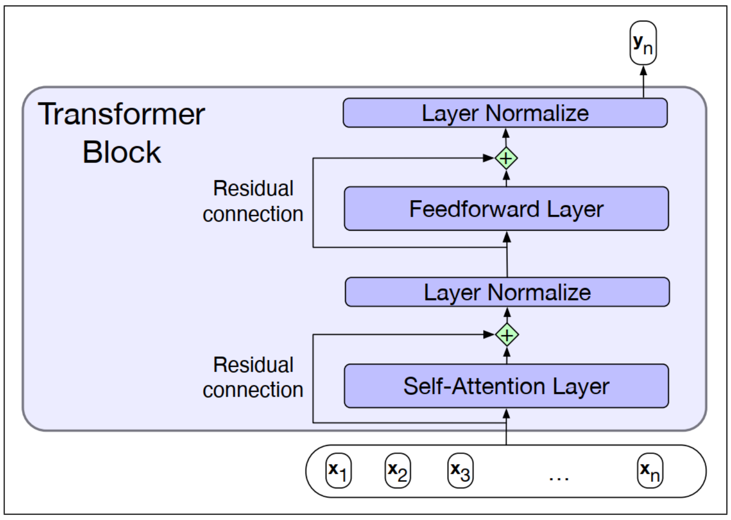


Figure 12: Illustration of a transformer block with all the common components. The original illustration can be found in Jurafsky and Martin (2023b), p. 216.

Layer normalization and residual connections in the transformer block both ease the information flow between the layers. The *layer normalization* includes summing up vector values and normalization to a certain distribution to avoid large fluctuation of values across different layers, whereas the *residual connections* enable the transfer of information directly from a lower layer to a higher layer, bypassing the intermediate layer (Jurafsky and Martin, 2023b).

The incorporation of residual connections is done via the addition of a layers input vector to its output vector before forwarding it through the network (illustrated through the green plus symbol in Figure 12). The integration of residual connections is applied after both the attention and feed forward sub-layers within the transformer block. Consequently, the resulting summed vectors undergo normalization through layer normalization layers to ensure appropriate scaling for subsequent processing steps in the transformer model (Jurafsky and Martin, 2023b; Ba et al., 2016).

In case the encoder and/or decoder each consist(s) of more than one transformer block, these are interconnected in the structure. Like already mentioned, the encoder and decoder are connected with the context vector as internal representation of the input in between.

In summary, the model architecture containing *encoder and decoder*, is a standard component in the architecture of (most) transformer models, and consists of two stacks of so called transformer blocks representing the encoder and decoder (Vaswani et al., 2017; Zhao et al., 2023).

2.7.5 Illustration of the Transformer Architecture

Figure 13 sketches the transformer architecture like originally proposed by Vaswani et al. (2017) and containing the components described above. The two grey blocks left and right show the first encoder and decoder blocks respectively. Both the encoder and decoder consist of transformer blocks (Section 2.7.4) having two sub-layers: multi-head self-attention and a fully-connected feed forward network. Additionally, the decoder has an encoder-decoder self-attention, which is named *multi-head attention*. In Figure 13 this is located after the decoder multi-head attention and before the feed forward network (so quite in the middle of the decoder part).

The encoder has stacked multi-head self-attention layers (see light orange box in Figure 13) to encode the input sequence into a vectorized, continuous representation. The self-attention method is called in this case "multi-headed" because the self-attention for a sequence is executed multiple times with different weights which results in multiple vectors, called heads, that are combined in the end (Kortschak, 2020; Vaswani et al., 2017).

Those continuous representations are then forwarded to the decoder, which performs cross-attention on these representations to generate an output sequence. The cross-attention layer behaves the same way as the multi-headed self-attention layer within the transformer blocks, but with the difference that parts of the input to the cross-attention layers come from the encoder (more precisely the *keys* and *values*) and others (the *queries*) from the previous layer of the decoder (keys, values, and queries were explained in Section 2.7.2). For further details on the mathematics of how the cross-attention layer is connected in language translation see textbooks like Jurafsky and Martin (2023d, p. 257ff).

Every step of the model is *auto-regressive*, which means that the previous representations are used additionally as input when generating the next representation (Vaswani et al., 2017).

The decoder also has *masked multi-head self-attention layers* (see dark orange box in Figure 13), which "remove" (mask) words of the input text so the decoder ignores some positions of the input and can learn to predict words without using all parts of the original sequence. Essentially, attention pairs are removed via the masked attention layers. For example in a transformer model that generates language, the encoder part does generate one token at a time. Masked attention is used there to ensure that during the generation of each token in the output sequence, the model only attends to the previously generated tokens and not future tokens – the future tokens are masked in that case. This is a case of left to right self-attention, which was covered above in Figure 9 (Jurafsky and Martin, 2023b).

Masked attention can be implemented before the softmax in the attention score computation

A_{ij} that was described in Section 2.7.2, via adding a matrix that has basically all 0s except for the places where the attention between tokens should be cut, where it is $-\inf$ (Jurafsky and Martin, 2023b; Kortschak, 2020).

The feed forward networks (blue boxes in Figure 13) are fully-connected networks having two linear transformations and serve as sub-layers in encoder and decoder respectively. They process the outputs of the self-attention layers before passing them from encoder to decoder or to the next stack of layers (Vaswani et al., 2017).

After the feed forward and self-attention layers, normalization steps are conducted (yellow boxes in Figure 13). The normalization includes summing up vector values and normalization to a certain distribution to avoid large fluctuation of values across different layers (see Section 2.7.4).

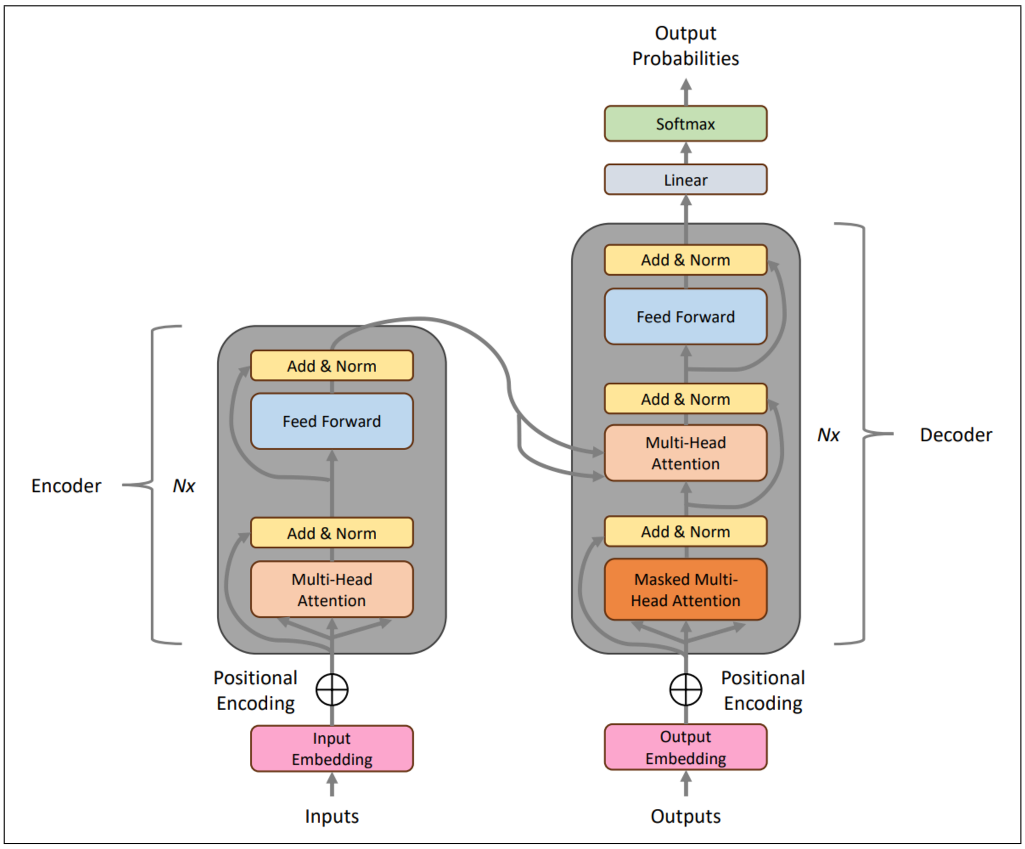


Figure 13: Transformer architecture with encoder-decoder for language modeling, first proposed by Vaswani et al. (2017). The figure is a redrawn illustration which can be found in the original paper on p. 3.

There is usually more than one transformer block in a LLM – and there are usually more than one encoder and decoder block as well. Encoder and decoder blocks are stacked n times, which is noted as Nx in Figure 13 and means that the output of the encoder and decoder of one level are used as input for the next encoder and decoder respectively (Kortschak, 2020). The last decoder outputs a vector which is transformed via a linear layer and softmax to output predictions. First, the fully-connected linear layer is processing the decoders output and outputs itself the logits vector[23] of the vocabulary. Next, the softmax function is applied to get the probabilities of each possible word and the word with the maximum probability (argmax) can be chosen afterwards. This is not the only occurrence of the softmax function – also the self-attention layers have a softmax in the end (Vaswani et al., 2017).

In transformer models, encoders can be either left-to-right oriented, but also bidirectional in the self-attention layers (see Figure 9). Two famous LLM, namely BART (Lewis et al., 2019) and BERT (Devlin et al., 2018), have bidirectional encoders in their model architecture. Taking information from before and after the current word of the sequence into account is important especially for downstream tasks like sequence classification and labelling[24], whereas left-to-right transformer models are useful for but also limited to tasks like contextual generation, for summarizing text, and machine translation (Jurafsky and Martin, 2023c). Both BART and BERT are LLMs using an encoder-decoder architecture as just described on the basis of the base transformer model and will be further described later (see Section 5.2).

The transformer architecture (Vaswani et al., 2017) has become the preferred model architecture for developing various LLMs due to its model characteristics that allow a scaling of language models to hundreds or thousands of billions of parameters with good parallelizability (Zhao et al., 2023). Mechanisms of the transformer architecture, including self-attention and positional encoding (see Section 2.7.3), help the model to learn word relations among each other even across a distance of several words (Jurafsky and Martin, 2023b; Vaswani et al., 2017).

[23]A logit is a raw model output value. The logit vector is a vector containing the raw model outputs for the tokens in the vocabulary, which makes the logit vector a rather large vector.

[24]Labelling is a task where the word is assigned to a category of word types, like e.g. a noun or a verb. One kind of labelling is part-of-speech tagging, where grammatical word types are assigned. For further information see reviews like e.g. Zewdu and Yitagesu (2022).

2.7.6 Pre-training Techniques for Large Language Models

The transformer model is trained using a process called *attention-based sequence-to-sequence learning* where it tries to predict the next word in a sequence. It learns by comparing its predictions to the actual output and adjusting its parameters to minimize the difference between the predicted and actual output.

The process of learning a representation of words can be called pre-training (Jurafsky and Martin, 2023b). Pre-training tasks in Natural Language Processing (NLP) can be broadly categorized into five methods for learning representations (Zhou et al., 2023): Masked Language Modeling (MLM), Denoising Autoencoder (DAE), Replaced Token Detection (RTD), Next Sentence Prediction (NSP), and Sentence Order Prediction (SOP).

Masked Language Modeling (MLM): MLM involves randomly masking some words (tokens) in the input sequence and then predicting these masked words during pre-training to reconstruct the original text (tokens). The most well-known example of an MLM-based model is BERT (Devlin et al., 2018).

Denoising Autoencoder (DAE): DAE introduces noise into the original document and then trains the model to reconstruct the original input from the noisy data. BART (Lewis et al., 2019) serves as a representative model based on this technique.

Next Sentence Prediction (NSP): NSP aims to help the model understand the relation between two sentences and determine if two sequences follow each other in the original text. This task involves taking two sentences from different documents and determining whether their order is correct (Zhou et al., 2023). BERT (Devlin et al., 2018) provides a typical example of a model using NSP.

Sentence Order Prediction (SOP): SOP is a task similar to NSP but with a different approach. It utilizes two contiguous fragments from a document as positive samples and considers the exchange order of the two fragments as negative samples. LLMs (Language Model Models) that utilize SOP, such as ALBERT (Lan et al., 2020), can effectively model the correlation between sentences (Zhou et al., 2023).

Replaced Token Detection (RTD): RTD is a discriminative task used to determine whether the language model has replaced the current token. The concept of RTD was first introduced in the ELECTRA model (Clark et al., 2020). In comparison to MLM, the tokens in the RTD method are not masked but replaced by plausible alternatives and in the discriminative model pre-training predicted if the token in the input was replaced or not. The method is also used in the DeBERTaV3 (He et al., 2023) because it is assumed to be more efficient in pre-training compared to the MLM that is used by the DeBERTa base model (Microsoft). More details on the DeBERTa base model are in Section 5.2.

It can be noted that RTD, SOP, and NSP are examples of contrastive learning methods. They operate under the assumption that the observed samples are more semantically similar than random samples. These pre-training techniques play an important role in advancing language models, which enables them to understand and generate text that is very similar to human language (Zhou et al., 2023).

In the pre-training phase, a LLM obtains rich knowledge about word meaning. This enables the model to be further fine-tuned either on a collection of text from a specific domain and either afterwards on a specific downstream task or directly on the specific downstream task (Hugging Face, 2023c). Like mentioned already in Section 2.7.2, a downstream task is a task that the LLM should perform in the end, e.g. next word prediction, question answering, text classification, or natural language inference (Jurafsky and Martin, 2023c).

Fine-tuning of LLM refers to the task of taking pre-trained models and further training them on data sets related to downstream tasks, that should be performed by the model after successful training. Application-specific parameters are usually added and in some cases additional layers can be added during fine-tuning to the model (Jurafsky and Martin, 2023c). During fine-tuning, labeled data is used to train these application-specific parameters.

A task that is not directly (without fine-tuning) grasped by language models is the relationship between sentences. This is important for downstream tasks as question answering and natural language inference (NLI) because they require reasoning about sentence pairs and their relationship with each other (Devlin et al., 2018). Through fine-tuning, the LLM infers knowledge about text already gained during pre-training also to the new downstream task. Since LLMs trained on the downstream task of NLI exist and the primary advantage of LLM is the ability to generalize via transfer learning, some authors like e.g.Yin et al. (2019) pointed out the efficiency of using them for classification purpose as well.

2.8 Natural Language Inference

One task of text classification is the classification of a pair of input sequences. An application of sequence classification is logical entailment. Recognizing textual entailment, also called natural language inference (NLI), has the goal of describing the relationship between two input sequences with either entailment, contradiction, or neutral (Jurafsky and Martin, 2023c).

The NLI ability is often tested after LLM pre-training in order to evaluate the natural language understanding capabilities of models (Joshi et al., 2020). A popular data set, in the field of NLI, to train models on sentence understanding is the Multi-Genre Natural Language Inference data set (MultiNLI; Williams et al., 2018). The data set contains more than 400k sequence pairs from 10 domains in English. The sequence pairs consist of a *premise* and a *hypothesis*. Every premise in MultiNLI consist of text parts from an pre-existing text whereas the hypotheses were generated by human annotators in order to produce sequence pairs, with three different hypotheses per premise. The novel sequences (hypotheses) were formulated in a way that: entailment, contradiction, or neutral (neither one of the other two conditions) is the logical inference. The label *entailment* means that the premise entails the hypothesis whereas the label *contradiction* means that the premise and hypothesis do not logically follow each other and both statements cannot be true at the same time[25]. In case of *neutral* neither one is true (Jurafsky and Martin, 2023c; Williams et al., 2018).

Apart from the MultiNLI data set, another well known data set in the field of NLI is the Stanford NLI Corpus (NLI; Bowman et al., 2015), which can be seen as the ancestor of MultiNLI and was the first data set to evaluate language models regarding natural language understanding ability. Due to the disadvantage of SNLI containing just one text genre, the MultiNLI data set was created (Williams et al., 2018).

An example of an NLI application are language models performing zero-shot classification. Those models usually have a state-of-the-art language model as baseline and are fine-tuned on an NLI data set (e.g. RoBERTa Liu et al., 2019, which has an NLI fine-tuned version by Facebook AI (2019)). The fine-tuning on the NLI task with models like e.g. BERT works in a way that the premise-hypothesis pairs are passed through a bidirectional encoder (see encoder explanation above, Section 2.7 or in Jurafsky and Martin (2023b)).

On a data set for common sense natural language inference (SWAG data set by Zellers et al., 2018), that contains sentences with logical inference describing situations that happen first and which situations are plausible to follow, the models back in 2018 performed much worse compared to humans, who reached an accuracy of 88%. Shortly after that, a famous LLM called BERT (Devlin et al., 2018) was introduced and already reached human-like performance with more than 86% in the inference tasks of that specific data set. Inference and extraction is not an easy task for humans as well. Depending on the task and data set, humans might also reach an accuracy above 90%, but usually not above 97% (e.g. Zellers et al., 2019).

Even though LLMs aim to perform as good as humans, it has to be noted that humans make mistakes in manual tasks like extraction or classification as well. To assess the complexity and error rates of specific tasks in text analysis, benchmark data sets used for evaluating

[25]An example of a contradictory premise-hypothesis pair can be described as follows: When the premise is describing a broker actively showing clients a house and the hypothesis is stating that the man is sleeping, the premise-hypothesis pair is clearly about a contradiction. Explanation: A person cannot be actively doing something and sleeping at the same time (Ruder, 2020).

Large Language Models incorporate human performance measurements. Comparative studies of human proficiency in natural language tasks have indicated that, in most cases, humans still outperform LLMs in terms of accuracy. However, it is worth noting that LLMs have shown consistent improvement, progressively narrowing the performance gap over time. Due to the pursuit of NLP to achieve human-like performance and the utilization for various purposes, NLP is considered as a discipline of Artificial Intelligence (Liddy, 2001).

2.9 Zero-shot Classification

Large Language Models (LLM) are able to transfer from learned to unknown tasks and domains (Wang et al., 2023). Since zero-shot classification is a key method used in the current thesis, it is explained further in this section.

Whenever no labelled data is available, zero-shot classification can become part of consideration since state-of-the-art LLM models are able to perform zero-shot tasks already with a comparably close quality as humans (Radford et al., 2018, 2019). Besides the advantage of not needing labelled training data, another advantage of the zero-shot ability of LLM is that the complex model fine-tuning and deployment is not needed (Wang et al., 2023).

In traditional classification methods the outcome labels are usually numbers and the underlying task does not have to be "understood", whereas in zero-shot classification some understanding of the labels and inference from the text is assumed (Yin et al., 2019). Besides the fact that no labelled training data is needed in zero-shot classification, another advantage is that the label space is not fixed (Yin et al., 2019). In conventional text classification the number of classes has to be fixed and annotated beforehand, but zero-shot classification does not have a boundary regarding the hypothesis space. Therefore, zero-shot classification approaches can be more flexible and adaptive, as they have the ability to generalize to new categories or labels that the model has not seen during training.

A overall definition of zero-shot classification goal was described by Yin et al. (2019, p. 2). When using this method, one:

> "aims at learning a classifier $f(\cdot) : X \to Y$, where classifier $f(\cdot)$ never sees Y-specific labeled data in its model development."

In the present thesis, pre-trained natural language inference models will be used as sequence classifiers, as first suggested to be successful by Yin et al. (2019) and further reported by other researchers to work effectively (e.g. Sainz and Rigau, 2021; Chalkidis et al., 2020a). The selected and applied models are further described later, in Section 5.2. Moreover, the concept of how zero-shot classification works without a predefined hypothesis space is described in Section 5.4, where the framework of algorithmic implementation is outlined.

2.10 Information Extraction in the Legal Domain

There has been a lot of research on text processing and classification in the last years, and even some research in the legal domain. This section will cover research of text processing in the legal domain with focus on research in NLI. The reason for covering research in the legal domain is that the data basis or the current study (further explained in Section 3) is from the legal domain.

Analysis of textual data in the legal domain was initially done with regular expression rules (see Section 2.6), but this has the disadvantage of being time consuming since many rules have to be defined by domain experts (Zhao et al., 2022). Besides that, classical models used in legal text classification include the Support Vector Machine (SVM; Joachims, 1998) based models on one hand (e.g. with the goal of court decision classification, Sulea et al., 2017; Aletras et al., 2016), logistic regression (e.g. Chalkidis et al., 2017) and neural frameworks like CNNs (e.g. Hammami et al., 2021) and deep learning methods (e.g. Chalkidis et al., 2020b) on the other hand. It has to be noted also that these approaches can contain rules crafted by humans manually, like described e.g. in Chalkidis et al. (2017) or illustrated with a basic example in Figure 5 (Section 2.2).

Since more recently LLMs like the BERT (Bidirectional Encoder Representations from Transformers; Devlin et al., 2018, further model description in Section 5.2) gained popularity, even more focus has been set on text understanding and classification for the legal domain (Zhao et al., 2022). It has been shown, that classification of legal texts with multiple labels is possible when enough training data is available (Shaheen et al., 2020).

The use of pre-trained Large Language Models with fine-tuning on legal texts has already been explored and published as LEGAL-BERT (Chalkidis et al., 2020b). LEGAL-BERT has been trained on 12 GB of text from the legal domain (court cases, contracts, etc.) in English. In 2020 BERT-base was already able to classify multi-label EU law cases with an F1-score > 0.5, the authors of the benchmark paper describe that the improvement of 2.5% with LEGAL-BERT is meaningful and shows that the BERT model gains from the domain specific corpus. So far, no language model explicitly trained on German legal texts with the downstream task of zero-shot classification exists to the author's knowledge. Some researchers, like e.g. Van Hofslot et al. (2022), have shown that training LEGAL-BERT on a downstream task of legal text classification performs slightly better than BERT fine-tuned with the same task data set.

A first approach of zero-shot classification with LLM in the legal domain by Sarkar et al. (2021) showed that legal text classification performs worse with zero-shot compared to few-shot classification, where a few annotated examples were shown to the classifier. Since the authors used a strict threshold, only one pre-trained language model for zero-shot classification, and performed solely binary classification, it can be assumed that the usage of different pre-trained and more novel models can outperform those results with an F1-score around 0.6.

2.11 Structured Approaches for Data Mining Projects

So far, methods and research about information extraction from natural language have been discussed. At this point, it should be mentioned that the application of tools for information extraction from large data sets can be called *data mining* (Dawei, 2011). Data mining projects have specific requirements from start to end (Chapman et al., 2000). Therefore, in this section a structured method for conducting a data mining project will be described.

One methodology or framework that helps conducting a data mining project in a structured way is the CRISP-DM. The **CR**oss-**I**ndustry **S**tandard **P**rocess for **D**ata **M**ining (CRISP-DM; Chapman et al., 2000) framework, was developed by a group of data mining experts in the late 1990s. The experts emphasized the importance of a structured, iterative approach to data mining projects. This framework provides a clear and systematic way to navigate through the stages of a data mining project, ensuring that the process remains focused on business objectives and delivering actionable insights. The CRISP-DM methodology consists of six core steps, which are explained below and visible in the CRISP-DM reference model in Figure 14. The arrows in Figure 14 point out that the phases are not conducted sequentially, but moving backwards or forwards can be necessary depending on the outcome of a preceding phase. Hence, the framework is designed to be iterative, meaning that steps can be revisited and refined if necessary. The circle of arrows around the phases symbolizes the continuous ongoing knowledge generation from data mining. This means that e.g. after deployment of one project, the lessons learned can be used as input for future projects.

Since the underlying and core task of the current thesis is a data mining project with information extraction and classification, the phases of the CRISP-DM model provide structure to the project (report). The CRISP-DM phases described below contain references to the corresponding sections of this work. The organization of the next thesis section aligns with the reference model, making it easier to comprehend the underlying nature of the data mining project.

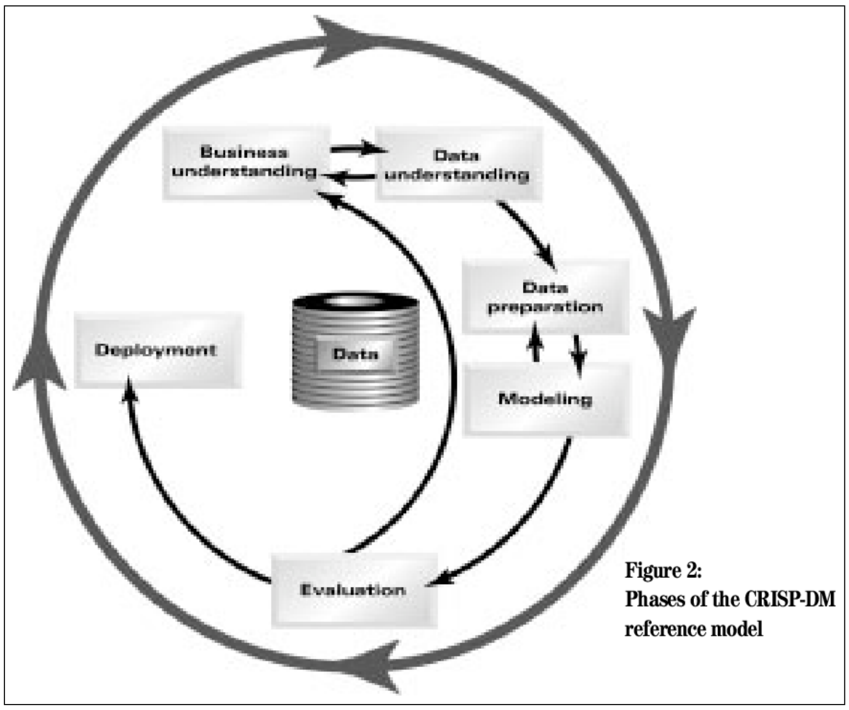


Figure 14: Illustration of the CRISP-DM model by Chapman et al. (2000, p. 13 in their publication). The six phases are describing the life cycle of a data mining project.

1) Business Understanding:

In this initial phase, the focus is on understanding the business problem or objective that the data mining project aims to address. Define the project's goals, requirements, and success criteria from a business perspective. Identify how data mining can contribute to solving the business problem and adding value.

In Section 1, the importance of automated contract analysis as well as area extraction were highlighted. Hence, the project's objective is area extraction and classification. The value of the thesis is investigation of one possibility to process purchase contract documents in order to extract and classify areas with the support of Large Language Models.

2) Data Understanding:

Gather, explore, and assess the available data sources. In this phase everything that helps understanding of the data and how it is obtained is reported. This includes gaining an understanding of the data's structure, quality, subsets, and relationships. Besides that, potential data issues, missing values, outliers, and other anomalies are identified.

Section 3 covers the data understanding in the current thesis. The overall structure of real estate purchase contracts is described in Section 3.1.

3) Data Preparation:

Selecting, cleaning, pre-processing, and any transformation of the data are belonging to this phase of the process. The data is brought into a suitable format for analysis. This includes handling missing values, outliers, and any inconsistencies as well as selection of relevant features or variables.

The data preparation steps are described in Section 4. One part of data preparation is the preprocessing of contract texts (Section 4.4) to extract parts of the text containing the relevant information for further processing. Besides that, the sub-sections contain the description of the data selection process (Section 4.1) as well as the generation of manual reference data, also called ground truth (Section 4.7).

4) Modeling:

In this phase, the appropriate data mining models and algorithms are selected. During application of the models on the prepared data set, certain preliminaries to the data structure can become clear which is why stepping back to phase *3) Data Preparation* can become necessary. If training or parameter tuning is conducted in the data mining project, these steps are part of the modeling phase. This involves adjusting model parameters and features to optimize performance.

The Modeling section below covers the model research, selection (Section 5.1) and description (Section 5.2) of the Large Language Models used for classification of sentence parts as well as the construction of an alternative, rule-based classifier based on regular expressions (Section 5.5).

Even though the model assessment including results in CRISP-DM based data mining projects is usually located in the Modeling section, it will be reported in a separate section to avoid confusion and ease reference. Therefore, all the model results (including zero-shot LLM as well as the alternative, rule-based approach) are reported in Section 6. The Assessment and Evaluation section also covers the metrics used for evaluation.

5) Evaluation:

After the model results are obtained, it is necessary to evaluate the model's performance against the business objectives and success criteria defined in the first step. One should use appropriate metrics to assess the model's accuracy and identify any potential issues or limitations of the model and results.

The final section of this thesis is the Evaluation and Discussion, which includes a summary of the results and a discussion in accordance with the relevant literature (see Section 7).

6) Deployment:

Project closure does not end with model creation – the knowledge gained must be made usable for the customer / business partner. Therefore, in the last phase of the CRISP-DM model the model integration possibilities and knowledge transaction are described. The deployment can be as simple as a report with visualizations or more complex like e.g. integration in the business data mining pipelines. Usually, the model integration steps are carried out by the customer and not the analyst performing the previous steps, but the customer should be provided with the information needed (Chapman et al., 2000).

As model deployment is beyond the focus of this thesis, this step is not detailed further and is not presented in any of the subsequent sections. In the last section (Section 7), the results of the experiments are summarized, and the overall project as well as gained insights are discussed. These insights are then communicated to the business partner, where the author is employed.

Summing up, it can be said that the CRISP-DM framework is designed to structure and clarify steps needed in a data mining project and that the following chapters are (roughly) structured according to the framework.

3 Data Understanding

3.1 Real Estate Purchase Contracts in General

The present work explores extraction and classification methods applied to text from real estate purchase contracts. In Section 1, some contract contents were described to give an intuition about what a purchase contract is and how the information contained can be used. The current section provides an overview of the general content, structure, and characteristics of contracts, outlining how these documents typically appear.

First of all, whenever someone is selling anything, the seller must either own the good(s) or have the legal authority, through some agreement, to sell it/them. In legal terms, the ownership can be called the *title to the object of purchase* (ger. Eigentumsrecht am Kaufgegenstand). Purchase contracts in general and also in the field of real estate involve the description of the title to the object of purchase which is, in course of the purchase contract, transferred from one party (seller) to another party (buyer) through a transaction of money (§§ 1053 ff. ABGB, 1812). Like mentioned in Section 1, this action can also be called *title transfer*.

In Austria, the real estate purchase contract alone is not sufficient for title transfer since the entry in the land register (ger. Grundbuch) is the final step necessary to officially gain ownership of the property (Österreichische Notariatskammer, 2022). All details about the date of ownership transfer and the conditions for the change of ownership (incl. land register entry conditions) are documented in the real estate purchase contract (Österreichische Notariatskammer, 2022; Brandauer, 2021).

The object of purchase can be any kind of real estate property. The *object type* refers to the type of the real estate, which can be distinguished at the most basic level between properties with a building or without. The latter are also called *unbuilt* plots of land. Depending on the purpose, various sub-categories can be described within a document. Contracts that are about a building or a part of a building can be e.g. single-family houses, apartments, commercial real estate, whereas contracts that involve unbuilt plots of land can be e.g. building plot, forest, farmland. There is no official definition of what object types exist, because they depend much on the need of people interested in transaction data. However, the usage types of the Austrian cadastre give a high level overview of types of land and land usage (see § 2 BGBl. II Nr. 116, 2010, for further information) which can serve as a rather rough guideline of types that can be encountered within a purchase contract.

Like any other purchase contract, also real estate purchase contracts have to contain certain information so it is clear what is sold and bought by whom under which conditions (see contract components in Austrian law: §§ 1053 ff. ABGB, 1812). Apart from the legally determined contents of the contract, the level of detail within a contract depends on the one hand on the contract parties and the party setting up the contract but also on the other hand on the individual and/or legal conditions of a property[26] (Brandauer, 2021; Österreichische Notariatskammer, 2022; Brandauer; §§ 1053 ff. ABGB, 1812). For example, the comprehensiveness of the object description can vary from basic location description with land register identifiers[27] to detailed description of every room size and building material of the whole building complex. Therefore, the amount of pages in real estate purchase contracts can vary between less than 10 to even more than 100 pages.

Purchase contracts usually start with a title, surrounded by notary stamps, and underneath the names and personal information (date of birth, address, social security number, company number) of the contract parties (usually only seller and buyer). Below or on the next page, there is usually the preamble, where the ownership status and the location of the *object of purchase* (ger. Vertragsgegenstand) are described. The preamble can consist of just one paragraph but also of several pages. Some notaries design their legal documents with a cover page, giving an overview of the contract, followed by a table of contents, and afterwards a detailed preamble. The structure can vary a lot depending on the notary. The location, top number, and area of an object of purchase is, if it is present in a contract, usually stated in the preamble. A full section about the object of purchase can also be dedicated in later sections of the contract. Repeated descriptions of the object of purchase, including several occurrences of the areas are possible.

Moreover, real estate purchase contracts can contain various appendices and amendments, like plans and images, utilizable value appraisal (ger. Nutzwertgutachten), energy performance certificate (ger. Energieausweis), and construction and equipment description (ger. Bau- und Ausstattungsbeschreibung). An example of an anonymized purchase contract is given in Appendix A.

[26]Special conditions are e.g. liabilities, which are rights of other people in relation to the sold property or parts of the property.

[27]The most basic description is to give cadastral municipality name and land register entry number. It can happen that there exists no address (e.g. plot of land, lake, forest) or the address is not given because the given identification was sufficient for the contract parties.

3.2 Contracts as Data Basis

In Austria, real estate purchase contracts are usually either written by a notary or a lawyer[28]. Notaries have to be involved in real estate transactions because they are an official institution entrusted with the responsibility of confirming authenticity of signatures and documents (Österreichische Notariatskammer, 2023b). In case the contract is written by a lawyer, a notary still has to be involved to guarantee the signatures of the contract parties belong to the real entities they claim to be (also called *notarised signature*, ger. beglaubigte Unterschrift).

Even though a real estate purchase contract has to contain certain information, the style and wording can vary. Currently, there are 536 notaries listed in Austria (Österreichische Notariatskammer, 2023a) and the number of lawyers is even higher. Even if one assumes every notary has internal contract templates and guidelines for wording, there would be hundreds of different styles for a contract possible. This has to be considered in processing the texts of this kind of documents.

Regarding the accessibility of real estate purchase contracts, it has to be mentioned that the Austrian land register is publicly accessible (Bundesministerium für Finanzen, 2023). Further, since 2006 there is an electronic certificate archive for certificates, like real estate purchase contracts, which is hosted by Austrian juridical system (Bundesministerium für Justiz, 2023). The certificate archive contains scanned documents in PDF format with metadata.

The unstructured text data set used in this thesis was obtained from Austrian real estate purchase contracts which are in German. In order to analyse the content and features of the real estate purchase contracts, the textual data needs to be retrieved from the scanned documents. The standard procedure to obtain text from images is optical character recognition (OCR). Since the process of translating text in images to machine-readable text was already executed beforehand, the present thesis will not further describe OCR and refers, with different terms (e.g., text documents, purchase contracts), solely to the textual data obtained from scanned real estate purchase contracts, unless stated otherwise.

Figure 15 shows a typical contract paragraph with object of purchase description – before OCR. It contains the areas of the sold apartment, including areas of terrace, cellar, and garden. Paragraphs like this are the parts in the real estate purchase contracts which are of special interest in the current work, since they contain the relevant areas to extract and classify. The areas are usually mentioned in the continuous text (and not in tables), which can make it difficult to detect which area belongs to what.

[28]It is even possible to set up a real estate contract (except confirmation of authenticity of signatures) without the help of a lawyer or notary, even though this is not recommended (Österreichische Notariatskammer, 2022). The author assumes that this case happens rarely.

Like already mentioned in Section 1, text processing can, to some extend, be performed through mimicking human behaviour in automated text analysis steps. In order to discriminate relevant text parts from irrelevant ones, the surroundings, also called context information, of the areas is further analyzed in the current work and text parts far away from any area are disregarded. In Figure 15, the sentence parts containing areas and their respective surroundings are highlighted with colors, whereas the preceding text without area information is not highlighted. The four different colors in the contract paragraph symbolize the sentence parts forming the *sequences*, to be analyzed in more detail. One reason for selecting surroundings of areas to analyze them further is that humans would also scan the text to find the part of interest and read only the surroundings to comprehend the meaning and find out relations between words and sentence parts.

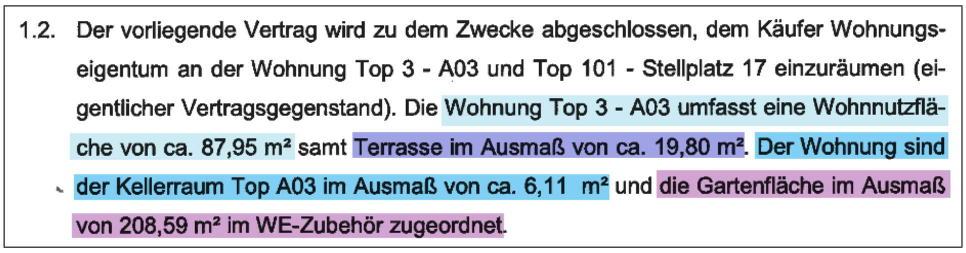

1.2. Der vorliegende Vertrag wird zu dem Zwecke abgeschlossen, dem Käufer Wohnungseigentum an der Wohnung Top 3 - A03 und Top 101 - Stellplatz 17 einzuräumen (eigentlicher Vertragsgegenstand). Die Wohnung Top 3 - A03 umfasst eine Wohnnutzfläche von ca. 87,95 m² samt Terrasse im Ausmaß von ca. 19,80 m². Der Wohnung sind der Kellerraum Top A03 im Ausmaß von ca. 6,11 m² und die Gartenfläche im Ausmaß von 208,59 m² im WE-Zubehör zugeordnet.

Figure 15: Paragraph of a purchase contract containing several areas. The first sentence part highlighted (turquoise) describes the area of an apartment. The second sequence (marked purple) describes the area of the terrace, the third one (blue) the cellar and the fourth one (pink) the garden area.

It has to be noted here that the sequences in the contract image (like e.g. in Figure 15) look similar, but are not exactly the same as the ones analyzed in the text format. The step of transforming images to text with OCR takes place in between. Thereafter cleaning steps and other processing steps are applied in order to have a harmonized text basis. These cleaning steps are explained in the next section and include removing of dashes at line breaks and harmonization of words like "usable living area" (ger. Wohnnutzfläche) to match a distinct class (rather than the two *usable area* and *living area* classes at once).

Purchase contracts can be seen as a part of the legal domain because they are usually written by notaries or lawyers with the incorporation of legal guidelines. The special wording of contract can make it more difficult to clean and harmonize text as well as to detect the areas context correctly.

Areas within the purchase contracts can refer to a variety of features, give an insight of how big the object of purchase is and help estimating the price per square meter. The detection of areas is therefore important to compare objects among each other and select contracts of interest for real estate valuation purposes. Like discussed in the introductory section, the extraction and classification of areas is important for the business partner (see Section 1 for further details on this).

3.3 Defining Labels

The areas within a purchase contract can refer to different objects and a variety of features. Two different sets of labels referring to types of areas are used and evaluated: 12 fine-grained labels and 5 aggregated labels. It is assumed that a label can be used to describe any (pre-processed) sentence part used in the classification process. The so called text *sequences*, which all contain an area, are assumed to be suitable and have sufficient context information to infer the proper categorization of the respective area. Even though the class is derived from the *sentence parts* via the classification methods, it is assumed that the class is referring to the actual *area* contained in the text sequence.

3.3.1 Fine-grained Labels

Initially, a set of 12 types of areas (labels) were identified from the contracts during the data curating process. Those labels describe the most prominent types of areas within the contracts and cover more than the areas of interest for the current investigation. Since the textual data was in German, the selected fine-grained labels for sequence classification were as follows:

- Wohnfläche (engl. living area)
- Nutzfläche (engl. usable area)
- Gesamtfläche (engl. total area)
- Loggia (engl. loggia)
- Zimmer (engl. room)
- Außenfläche (engl. area outside)
- KFZ (engl. motor vehicle)
- Balkon (engl. balcony)
- Grundstücksfläche (engl. plot area)
- Garten (engl. garden)
- Terrasse (engl. terrace)
- Sonstige (engl. other)

In Austria the term *usable area* is defined and regulated by law and describes the floor area without walls, balcony, or terraces of a real estate for residential or commercial use (Draxl, 2023; oesterreich.gv.at, 2023). On the other hand, the term *living area* is not defined by Austrian law but commonly used to describe the sum of areas of the rooms that can be used for living purposes (Draxl, 2023). However, the usable area and the colloquial term *living area* are not always used in a legally correct way in the purchase contracts[29]. Therefore, the two terms living and usable are distinguished and both extracted, even though they can have the same meaning.

In Austrian law there is also a distinction between balcony and loggia. A loggia is open only at one side and has walls or similar boundaries on all other five sides (OGH, 2003). Therefore, the two classes *loggia* and *balcony* are distinct classes.

It has to be noted that the set of classes is not disjoint. Even though *garden* and *terrace* are both areas that are located outside of a building, the class *area outside* was still included in the set of classes used in the experiments. The reason for that is the core goal to detect areas of usable or living area *inside* a building and to discriminate them from other areas. First experiments conducted with contracts not contained in the investigated data set were conducted during development of the overall approach and indicated promising results.

Roughly half of the 12 classes have the German word "Fläche" (engl. area) in the class name. One class without the word for area in it is the abbreviation "KFZ" (engl. motor vehicle). The label was chosen to describe the class of any kind of parking possibility with this rather short wording because it is assumed that the LLMs can infer from a sentence about a parking spot that this has something to do with a *motor vehicle*. The models should classify parking spots like outdoor parking space, garage, carport, and underground parking space belonging to class "KFZ".

The label *other* refers to all areas that are not covered by the 11 specific class labels.

3.3.2 Aggregated Labels

Due to the importance of comparability of built-up objects among each other, a smaller set of labels was chosen for the generation of and evaluation with ground truth data. Those primary class labels selected were: *living area* (ger. Wohnfläche), *usable area* (ger. Nutzfläche), *loggia* (ger. Loggia), *total area* (ger. Gesamtfläche) and *other* (ger. Sonstige). The areas not within the primary class labels were aggregated under the label *other* for evaluation purpose.

[29]The source for this information is a civil engineer working as consultant for the DataScience Service GmbH.

The label *total area* was selected because the area of habitable properties, like apartments, can be stated within a contract with the German term and a text based classification for living area would, depending on the sentence phrasing, not be possible in some cases. Further, the label of *loggia* was selected to be among the aggregated labels set because in the Austrian law a loggia is defined to be usable area (OGH, 2003).

After classification, the resulting fine-grained labels are mapped to the aggregated labels in order to have the same label sets to compare. Since this is simply an aggregation of classes, those labels are called aggregated labels. This will be explained more precisely in the next section.

4 Data Preparation

In order to perform any kind of classification with the chosen labels (see Section 3.3), the data needs to be brought into a suitable format. The data preparation section serves the purpose of conveying information about all the data processing steps conducted to obtain the data that is ultimately classified by the selected classification methods (according to CRISP-DM; Chapman et al., 2000). After explaining the selection of contracts (Section 4.1), the overall workflow of contract processing will be outlined (Section 4.3), followed by more detailed descriptions of the processing steps (starting at Section 4.4). Thereafter, the ground truth generation process is described (Section 4.7) and data set characteristics will be given (Sections 4.2 and 4.6).

4.1 Selection of Contracts

The selection of one or more samples depends on what kind of experiments are conducted and what kind of data is available. Foremost, it should be noted that the present thesis investigates information extraction with pre-trained or rule-based models. This means no model training or hyper-parameter tuning is conducted due to the selection of methods. Hence, no training set was defined or processed. However, for code development and model results evaluation two sets of contracts were selected from all the available contract text documents.

First, a set of contracts was selected randomly to develop a data processing logic with cleaning and information extraction. This set consists of 80 documents that were chosen randomly and are therefore a representative contract sample. Since the contracts were only used for development of the processing logic, they are not described any further. Those random contracts are only mentioned here to point out that the processing logic and class labels have not been developed with the data basis used for evaluation and model comparison to avoid bias.

Second, after the development of the processing logic and cleaning was done, a set of contracts with the purpose of data evaluation was selected randomly. Therefore, a set of 220 contracts was chosen with a random number generator from all available contracts[30]. For the window of selection, contracts processed by the courts between October 2019 and January 2023 were selected, excluding the 80 contracts that were used for initial process development. With this method of selection the set can be seen as a representative sample of purchase contract text documents in Austria.

[30]The cooperation company DataScience Service GmbH buys and processes purchase contract documents since 2019 on a daily basis.

4.2 Characteristics of the Selected Contracts

The PDFs of the 220 contracts, where the texts were obtained from, have on average 42 pages with a median of 15 pages. The length of contracts is positively skewed (1.16)[31]. Besides that, the raw contract texts analyzed have a median length of roughly 15k characters, distributed across 445 lines of text[32] (see Table 2).

Table 2: Characteristics of contracts selected randomly for examination in this thesis. Page counts were determined from the PDFs, while character and line counts were derived from the plain text documents obtained through OCR of the PDFs.

	contract document ($n = 220$)		
	pages	**characters**	**lines**
mean	42.3	74,353.3	1,248.8
SD	45.6	81,492.3	1,378.2
min	4.0	2,165.0	55.0
25%	10.0	15,311.3	270.3
50%	15.0	26,642.5	444.5
75%	68.3	140,113.0	2,037.8
max	152.0	288,065.0	5,699.0

Moreover, the contracts in the subset were distributed across Austrian states with different frequencies. Most transactions took place in Vienna (35%) and least transactions in Vorarlberg (1%). For an overview of the transaction locations see Figure 34 in the appendix (p. 158).

The 220 contracts were processed to some extend with the overall workflow to generate text sequences for manual labelling. Strict manual rules were set to ensure a consistent labelling procedure by the author and to ease distinction of the respective wordings. Detailed sequence labelling rules are described in Appendix B. Before the sequences can be described further, the procedure how the sequences were generated (incl. text pre-processing) needs to be explained.

4.3 Overall Workflow

This subsection outlines the workflow from a plain text document to the extracted and classified areas, as simplified in Figure 2 in the introductory section. The overall approach, illustrating how each contract is processed from text to the classification result of areas, is presented in Algorithm 1. Subsequent subsections provide a more detailed description of the components and steps in this workflow.

[31] unbiased skew, Normalized by $N - 1$

[32] Count of line breaks, repeated line breaks without characters in-between were only counted once.

Algorithm 1 Extract and classify areas of a purchase contract

```
 1: path ← path to purchase contract txt file
 2: method ← type of classification that should be used, which is either solely rule-based
    classification or transformer-based with model and hypothesis
 3: text ← Loading contract txt file by path
 4: If contract contains no areas → return None
 5: Perform initial text cleaning (roughly_clean_contract(text), see Alg. 2)
 6: Split the contract into sentences and sentence parts by suitable patterns. Conduct a split
    at dot (not preceded by digit), comma, semicolon, “und” (engl. and), “sowie” (engl. as
    well), bullet point symbol •
 7: Remove sequences that do not contain any digits or square meter (because they are irrel-
    evant sequences)
 8: Further filter and clean the remaining candidate sequences
        (filter_candidate_sequences(sequence_list), see Alg. 3,
        further_clean_candidates(sequence_list), see Alg. 4)
 9: If no valid sequence remains → return None
10: Reduce the amount of candidate sequences for further processing to $\min\{|\text{sequences}|, 12\}$
11: Extract the areas from the sequences
12: If there is no area in front of $m^2$, or if the area is too large or too small, remove it and the
    related sequence                                        ▷ $< 1.1$ or $> 999$k $m^2$
13: if method == zero-shot classification then
14:     Classify sequences with zero-shot model using transformer pipeline (see Section 5.4)
15:     Select the class with the largest logit for entailment as class result
16: else
17:     Classify sequences with rule-based classification (see Alg. 5)
18: end if                                          ▷ result is one class per sequence
19: return all areas (of a contract) with corresponding classes and the sequences they were
    extracted from
                    ▷ There is at least one combination of (sequence, area, class) per contract
```

Considering the CRISP-DM methodology (see Section 2.11), the steps of *data preparation* include the pre-processing of the text until there are clean, short sequences that can be used with a model (like e.g. a classifier). The description of the data processing to format the text for analysis (in our case classification) is described in Algorithm 1 up to step 12. In between, the contract processing is divided into pre-processing and cleaning (step 1-7 in Algorithm 1), which is further explained in Section 4.4, and additional processing steps (step 8-12 in Algorithm 1), which can be found in Section 4.5.

The processing workflow for the classification of sequences with large language models or a rule-based approach is detailed in the modeling section (see Section 5). Even though the current section is about data preparation, the project workflow is better understood when regarded as a continuous process and therefore modelling parts are mentioned at this point in a full process, with references to the later sections for model details. A more comprehensive explanation of how the zero-shot classification works (steps 14-15 in Algorithm 1) is detailed in Section 5.4, while the rule-based classification (step 17 in Algorithm 1) is discussed in Section 5.5.

After Algorithm 1 is executed on all the contracts in the sample with the zero-shot or rule-based method, the fine-grained labels as well as the related areas are known for every sequence of every contract – if any areas were present. In order to evaluate the class labels on a higher level, the labels are aggregated like described in Section 3.3.2. Assessment of correctness of the labels is conducted with the methods and metrics described in Section 6.1 with a ground truth. The ground truth is generated with the sequences constructed from executing Algorithm 1 until step 12 on all the contracts within the sample and manual assessment afterwards (see Section 4.7).

4.4 Data Pre-processing and Cleaning

To clean and harmonize the text and to split it into short sentence parts, several pre-processing steps are carried out. Since the textual data processed was extracted beforehand from scanned documents with OCR, special attention is given to the pre-processing of the text due to potential errors introduced by OCR. The pre-processing rules were constructed by inspecting and experimenting with contracts not included in the investigated set (like explained in Section 4.1). Sentences and sentence parts, all containing a valid area and context information, are the outcome of the pre-processing and further processing steps.

It is well known in the literature, that LLMs perform best with short sentences and context information consisting of a small set of words (e.g. reported by Pearce et al., 2021; Yang et al., 2023). Therefore, for the current study short sentences and sentence parts, called sequences, were prepared for the classification step.

After the contract text is loaded and confirmed to include area information, it undergoes an initial cleaning cleaning during step 5 of Algorithm 1. This cleaning step involves several heuristically generated steps like harmonization of spaces, punctuation and numbers, and is further described in Algorithm 2. While most steps in this cleaning algorithm are self-explanatory, additional explanation is needed for steps 3-4 due to the contract data being in German. Depending on the notary's stylistic preference, contract writers may use spaces or dots to separate thousands within numbers. Additionally, in German-speaking countries, commas are commonly used as separators for floating-point numbers (see the number formatting in the anonymized purchase contract in Appendix A, p. 4 of 8). Therefore, the number format is harmonized as explained in steps 3-4 of Algorithm 2.

After the initial cleaning is completed, the text is split into sentences and sentence parts containing an area in square meter and context information (starting at step 6 Algorithm 1).

At step 6 of Algorithm 1, the optimal patterns to split the sentences were heuristically generated with contracts not contained in the investigated set. Those patterns contained German words that usually concatenate sentences, like e.g. *und* (engl. and), *sowie* (engl. as well), but also signs like dot, colon, and comma under special conditions, like not near common abbreviations or numbers. Further symbols and words used for splitting were referring to currency or enumerations. This splitting step takes into account, that *circa* (engl. circa) is abbreviated in German as "ca.", it can precede areas and that there should not be a split after this dot. After the sentences and sentence parts were split, only the ones containing an area and context information are kept for further processing (step 7 of Algorithm 1).

Even though pre-processing serves the purpose of harmonizing and cleaning the text basis, in the pre-processing steps some words needed for classification could be removed which could deteriorate the performance. Removing of such words and phrases can happen on the one side due to the sentence splitting (see step 6 in Algorithm 1 above) because the relevant context needed for proper classification is not present anymore in the sequence.

Besides the previously mentioned special attention to cleaning that is needed due to OCR, pre-processing of text from special domains, like the legal domain, needs special attention and different methods as well, compared to text from more general domains (Habernal et al., 2023). Depending on the definition of where pre-processing ends, some authors like Habernal et al. (2023); Ying and Habernal (2022) count word tokenization to the pre-processing. Since tokenization is used as part of the Hugging Face model pipeline, in this thesis the tokenization is not seen as pre-processing, but rather as further processing step which is done in the course

Algorithm 2 roughly_clean_contract(text) – initial cleaning of the contract

```
Input: text, plain contract text, based on OCR, without any cleaning
Output: text_cleaner, after cleaning steps regarding spaces and symbols. The output is
    suitable for splitting into sentences and sentence parts
 1: Replace variants of square meter with $m^2$ as harmonization step
 2: Remove single and double quotes
 3: Remove space within integer and float numbers
 4: Replace comma as a separator in float numbers with a dot, remove dots occurring at 1k
    place
 5: Add dots before line breaks when the next line starts with an enumeration
 6: Undo line breaks with hyphens
 7: Replace line breaks before square meter with space
 8: Replace line breaks before enumeration with dot
 9: Remove lines with $\leq 4$ characters
10: Replace line breaks with spaces
11: Replace multiple occurrences of spaces, hyphens, commas, stars, and dots with a single
    occurrence of those
12: Add a space between digit and $m^2$
13: Add a space between a more than one lowercase letter followed by an uppercase letter
14: Add a dot before certain words that indicate a new sentence (because OCR probably did
    not get it), e.g. ger. „Die “ (engl. ‘The ’)
15: Replace common abbreviations near areas with the written form or remove the comma or
    dot to avoid sentence splitting later, e.g. ger. „rd.“ (engl. ‘approx.’)
16: return text_cleaner
```

of inference with the transformer pipeline (see Section 5.4).

4.5 Further Processing Steps

So far, the pre-processing of the contracts and the splitting of the text into sentence parts as well as selecting the area-related sentence parts were explained. These steps take place during the steps 1-7 of Algorithm 1. Before area extraction and classification takes place, the so far obtained sentence parts need to be further processed and cleaned with some heuristics generated manually (step 8-12 of Algorithm 1). The sequence filtering and cleaning steps are described in Algorithm 3 and 4.

The candidate sequences are filtered due to various conditions (see Algorithm 3), which were determined by investigation of contracts not in the data set used for the ground truth and were determined heuristically and through domain knowledge of the author. The filtering conditions take sequence characteristics like letter-digit-ratios and counts (steps 1, 2, 8, 12-14) as well as repeated words (step 11) and words indicating appendix sections and general building descriptions (steps 3-7) into account. Moreover, sequences with multiple areas are split to ensure that each sequence contains only one area along with its corresponding context information (step 9-10). Duplicated sequences are removed (step 15) since areas can be mentioned more than once in a purchase contract, e.g. once in the prelude and once in the object of purchase section. It is heuristically assumed that the wording is different in case an area information has the same size but refers to a different object.

The candidate sequences are cleaned due to various conditions (see Algorithm 4). The cleaning steps include repairing of wrongly OCRd letters (step 1), modification of non-disjoint class words within sequences (step 2-3) and removing of words that can mislead classification (4-5). In case no sequence for a contract remains after all these processing steps, *None* is returned for the respective contract (step 9 in Algorithm 1).

Once the sequences are prepared, the respective areas are extracted from the strings using a rule-based matching heuristic (step 11 in Algorithm 1). After area extraction, the areas are checked for too high or too low numbers and the sequences are filtered according to that (step 12 in Algorithm 1). The thresholds for filtering the sequences were set heuristically in this case. There can be very small (french) balconies with less than 2 square meters, but most areas below 1.1 m^2 can be assumed to be wrong. The upper bound was set heuristically as well, since most areas of properties are below 999.000 m^2. The initial and rather liberal setting of an area filter heuristic can be adapted and improved in future experiments.

Algorithm 3 filter_candidate_sequences_(sequence_list) – filter candidate sequences and remove implausible ones

```
Input: sequence_list, list of candidate sequences that contain digits and $m^2$
Output: sequences, filtered sentences and sentence parts, each still containing an area and
    context information. The output is suitable for further sequence cleaning as next step.
 1: Remove sequences that contain less than 50% letters                ▷ excluding white space
 2: Remove short sequences with less than 10 characters
 3: Remove sequences that contain energy performance certificate (ger. Energieausweis)
    information                                ▷ pattern defined with authors domain knowledge
 4: Remove sequences that contain appraisal report (ger. Nutzwertgutachten) information
 5: Remove sequences that contain subsidies, housing support
 6: Remove sequences that contain construction and equipment description
 7: Remove sequences that contain spans or ranges such as 50-100$m^2$, from, up to
 8: Remove sequences that contain more than three uppercase words and with at least four
    characters each
 9: in case there is a candidate with >1 $m^2$ occurrences, go over the candidates and split after
    $m^2$
10: Remove sequences that do not contain any digits or square meter      ▷ Again, due to split
    after $m^2$
11: Remove sequences that contain the same word >3 times
12: Remove sequences that contain less than 50% letters          ▷ Again, due to split after $m^2$
13: Remove sequences with more than 20 digits
14: Remove sequences with more than 1.5 times as many uppercase letters as lowercase letters
15: Remove duplicated sequences
16: return sequences
```

Algorithm 4 further_clean_candidates(sequence_list) – cleaning of the candidate sequences

```
Input: sequence_list, filtered candidate sequences
Output: sequence_list, cleaned candidate sequences. The cleaning steps involve expression
    harmonization and character cleaning. The output is suitable for further processing, incl.
    area extraction and classification.
 1: Repair common words containing the German letters "ö, ä, ü" which were wrongly detected
    as 'o, a, u' during OCR          ▷ Using a list of words containing e.g. 'Flache' → 'Fläche'
 2: Unify writing of living area, e.g. "WNF", "Wohnnutzfläche" (engl. living usable area) →
    "Wohnfläche" (engl. living area)
 3: Unify writing of usable area, e.g. "Gesamtnutzfläche" (engl. total usable area) →
    "Nutzfläche" (engl. usable area)
 4: Remove numbers spelled out as words and their surrounding parentheses, e.g. "21 (ein-
    undzwanzig)" → "21 ".
 5: Remove "Wohnungseigentum" (engl. home ownership) from a string containing parking ▷
    due to risk of misclassification with apartment
 6: Remove repeated "$m^2$" and multiple spaces
 7: return sequences_cleaned
```

Another heuristic value is set for the upper bound of sequences processed per purchase contract (Step 10 in Algorithm 1). At the time where the initial cleaning and processing functions were written, contracts not included in the sample under investigation often contained lengthy documents with numerous appendices. Despite preceding filtering steps, some irrelevant sequences remained. The author observed a lack of clear criteria for distinguishing between useful and irrelevant sequences for area extraction and sequence classification. Therefore, as a heuristic measure, the upper limit for the number of contract sequences processed was manually set to include only the first 12 sequences. This number was considered to be appropriate as most contracts (of the set used to write the cleaning and pre-processing algorithms) did not contain nonsense sequences from late appendix sections, and all relevant sequences/areas were preserved. However, since this decision is heuristic, it will be discussed alongside other proposed improvements in Section 7.

As next step after the pre-processing of the sequences, the classification of the sequence follows. This can either take place with a zero-shot model or with the rule-based classifier. Both are explained further in the sections below (see Sections 5.4, 5.5 respectively). As final result, the sequence with area and class per contract are returned (step 19 in Algorithm 1).

4.6 Characteristics of the Processed Sequences

The characteristics of the sequences after all steps of pre-processing and filtering are reported in Table 3. Those *sequences* are text parts, usually sentence parts, containing an area with context information and are used as input for the classification models. Example sequences are shown in Section 3.2 and in Section 4.7. Besides that, sentence parts have a length of 57 characters and 10 words on average (see Table 3). With the described method of data processing, one or more sequences containing an area and context information were extracted in 159 out of 220 contracts (72%).

Table 3: Characteristics of the sequences after pre-processing and filtering is completed. *Sequence* refers to contract sentence parts containing an area and context information.

	sequence ($n = 668$)	
	characters	**words**
mean	57.1	9.6
SD	35.8	4.8
min	11.0	3.0
25%	38.8	7.0
50%	46.0	9.0
75%	65.0	11.0
max	342.0	41.0

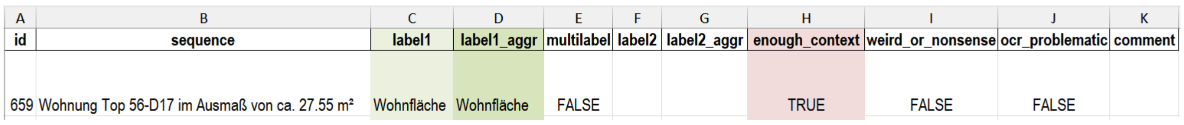
Clearly, a contract can describe the area of more than one object or feature. On average, a contract of the investigated set has 2-3 sequences, with a range from 0 to a maximum of 12 sequences per contract ($mean = 3, SD = 3.4, median = 2, IQR = 0 - 4$)[33]. It should be noted that the maximum of 12 sequences is somewhat artificial, as described above in Section 4.5. To ensure the extraction of relevant sequences, an upper bound per contract has been set heuristically (step 10 of Algorithm 1). Therefore, although the potential number of sequences within a contract could be higher, the upper bound was intentionally set to avoid the accumulation of irrelevant results for single contracts.

[33] IQR is the interquartile range from 25th to 75th percentile.

4.7 Manual Ground Truth Generation

For the evaluation of the classification approaches, the set of 220 contracts was cleaned and processed like they are before application of any kind of classification in the approach described above (Algorithm 1 steps 1-12). The resulting 668 sequences were stored in an Excel file for manual classification. The manual as well as the automated (model-based) classifications are conducted on the basis of the *same* sequences, enabling direct comparisons between the methods. The Algorithm for pre-processing and cleaning was described above (see Sections 4.4 and 4.5).

It is important to note that manual classification and the assessment of additional sequence features, such as identifying gibberish text[34], were exclusively performed by the thesis author. Consequently, no inter-rater reliability analysis can be reported.



A	B	C	D	E	F	G	H	I	J	K
id	sequence	label1	label1_aggr	multilabel	label2	label2_aggr	enough_context	weird_or_nonsense	ocr_problematic	comment
659	Wohnung Top 56-D17 im Ausmaß von ca. 27.55 m²	Wohnfläche	Wohnfläche	FALSE			TRUE	FALSE	FALSE	

Figure 16: Figure showing one finished row of the Excel file used for manual ground truth generation. Initially, only the columns A and B were filled with the *id* and *sequence* respectively.

In the Excel file (see Figure 16 for an example row), the sequences were stored to ease manual classification by a human. This file contains the columns for the fine-grained labels, aggregated labels as well as additional columns to manually detect problematic sequences on the fly. Noting down problematic characteristics of sequences included detection of gibberish content, identification of OCR-related issues, if there is enough context around and describing the area and possible multi-label cases.

An example sequence, which is also shown in Figure 16, is *"Wohnung Top 55-D16a im Ausmaß von ca. 27.55 m^2"* (engl. apartment top 55-D16a in the extent of 27.55 m^2). This sentence part is labelled as *living area*, having enough context information for area classification, having no gibberish, or OCR-related problem, and no additional suitable label.

Another example sequence is *"einer Terrasse von rund 15 m^2 verbunden ist"* (engl. connected (to) a terrace of around 15 m^2). This sequence has the fine-grained label *terrace* and the aggregated label *other*. Again, the context information is assumed to be sufficient for understanding what the area refers to, it is a meaningful sentence part without gibberish and there are no OCR problems. Further example sequences are reported in the footnotes in Section 6.3.3 (see p. 103ff) and in Appendix C.3.

[34] *Gibberish* refers to words not making any sense (Collins English Dictionary).

4.8 Characteristics of the Ground Truth

Following manual classification of the sequences, the characteristics of the sequences can be described as outlined in Table 4. Notably, *Grundstücksfläche* (engl. plot area), *Sonstige* (engl. other), and *Wohnfläche* (engl. living area) were the fine-grained labels assigned most frequently, each occurring more than 100 times within the 668 sequences.

Table 4: Counts of labels obtained by manually classifying the sequences to establish a ground truth. Each sentence part has a first label, and some have a second label.

	sequence classification			
	first label		**second label**	
	fine-grained	**aggregated**	**fine-grained**	**aggregated**
Außenfläche	0	0	1	0
Balkon	68	0	1	0
Garten	38	0	2	0
Grundstücksfläche	154	0	14	0
KFZ	17	0	2	0
Terrasse	28	0	3	0
Zimmer	1	0	13	0
Loggia	12	12	2	2
Nutzfläche	21	21	5	5
Gesamtfläche	67	67	29	29
Wohnfläche	124	124	5	5
Sonstige	138	444	12	48
total count	668	668	89	89

The *second label* (see the two right columns in Table 4) corresponds to *label2* shown in Section 4.7 (Figure 16) when describing the manual data classification process in the Excel file. Despite efforts to properly clean and split sequences during contract processing, 89 out of the 668 sequences can be described with a second label, classifying them as multi-label cases (see Table 5). Most of these cases were caused by splitting and/or string cleaning problems. An example for a sequence that got two labels in manual labelling, due to faulty splitting is *"Gesamtfläche ca. 1912 m^2 Erdgeschoss Garage mit 27 Abstellplätzen"* (engl. Total area approx. 1912 m^2 Ground floor Garage with 27 parking spaces), where both *total area* as well as *motor vehicle* were assigned when classifying the sequence manually.

Table 4 (left) displays the distribution of the class labels in the sequence classification. The first label from type fine-grained labels shows 138 sequences were classified as *Sonstige* (engl. other). Among those labelled cases were 53 sequences referring to cellar, 7 sequences referring to storage room, 42 sequence referring to gibberish and 20 sequences not having enough context to determine the label properly, according to manual sequence classification.

Regarding prevalence of *plot areas* in the contracts, in manual sequence classification, 66 out of 220 (30%) where found to have at least one sequence with the label plot area.

Problematic characteristics of the sequences, which were noted down when assigning the labels to the sequences manually, are displayed in Table 5. Even though strict filtering of sequences was applied, 9% of sequences still contained nonsense words and letters (gibberish) and can be assumed to result in unreliable classification results. In 8% of the sequences the sentence part was presumed to be too short to determine what the area refers to, like e.g. in this case: *"Loggia im Gesamtausmaß von ca. 90.48 m^2"* (engl. loggia with a total amount of roughly 90.48 m^2). This sentence was obviously split at the wrong place and has not enough context information to justify where the area belongs to. Nevertheless, it was manually labelled for the ground truth with *loggia*. Only from more information than in the sequence, e.g. from reading the full paragraph, one sees that the area of the apartment (and not the loggia) is 90.48 m^2.

Apart from too short sequences, another problem were the multi-label cases, which occurred in 13% of sequences and always had two (surprisingly not more) labels. An example for a valid sequence[35] that has two suitable labels is *"das Grundstück 15/6 mit einer Gesamtfläche von 192 m^2"* (engl. the plot 15/6 with a total area of 192 m^2), where both *plot area* as well as *total area* are suitable labels that have been assigned to the sequence during ground truth creation. Nevertheless, when aggregating the labels, the label *other* is assigned anyways and the two (potentially valid) labels do not conflict anymore (see Section 3.3 for explanation of fine-grained and aggregated labels).

Table 5: Counts of special sequence characteristics. Sequences were flagged during manual classification to be problematic (often due to faulty splitting or cleaning). The reference for percentage is the total number of sequences checked ($n = 668$).

	yes (%)	**no (%)**
too little context	53 (8)	615 (92)
peculiar / gibberish	61 (9)	607 (91)
OCR problem	28 (4)	640 (96)
multi-label case	89 (13)	579 (87)

[35] After reviewing the sequence, no obvious mistakes in splitting, cleaning, OCR, or any other aspect are apparent. Therefore, the author would assess this sequence as being valid.

5 Modeling

The models used in this thesis are available via the *Hugging Face* platform[36]. The platform provides data and state-of-the art pre-trained models open source. The Hugging Face transformers framework (Wolf et al., 2020) makes it possible to use transformer based models via API or download them. The currently available models are trained to process textual data, image, audio, and also multi-modal data (Hugging Face, 2023b). Since the models on this platform are free for use, only models from Hugging Face, and none from other platforms, were used for the current study.

The process how the models were researched and selected is described below (Section 5.1). Moreover, this section contains descriptions of the selected models (Section 5.2) as well as an overview of the Hugging Face transformers framework (Wolf et al., 2020) (Section 5.4). In the end, the rule-based approach as an alternative method to classify the sequences, is explained (Section 5.5).

5.1 Model Research and Selection

The models available on Hugging Face can be searched by tags (model attributes) that were assigned by the creators. To find suitable models, the search was filtered for *Zero-Shot Classification* or *Text Classification* in order to find pre-trained models suitable for the zero-shot classification task.

It has to be noted, that the research for models was conducted on 2023-03-05 and the Hugging Face community is constantly contributing with new model resources. Hence, the numbers of models available with certain filtering options can have changed in the meantime.

Nevertheless, in March 2023 Hugging Face hosted 17,232 models tagged as suitable for *Text Classification.* Of these models only 1,836 were tagged with English, 73 were tagged with German language, and 43 multi-lingual respectively. By default, the models are sorted by number of downloads descending. Reading the model description of the top 10 most downloaded text classification models indicates that a model does not always have a language tag and the languages it was trained on can be contained only in the model description.

With regard to the tag *Zero-Shot Classification*, Hugging Face hosted 110 models in March 2023. Of these models 46 were tagged with English, 13 were tagged with German language, and 11 multi-lingual respectively.

[36]The platform is available via the URL: `https://huggingface.co/`

In addition to the task (e.g. Text or Zero-shot Classification) and language, the models can further be filtered on (popular) data sets they were trained on. Data sets used for Natural Language Inference (NLI) usually contain the three-letter abbreviation "nli", which facilitates the search. The documentation of the Hugging Face pipeline to use zero-shot classification recommends to search for suitable models via the term "nli" among the models (Hugging Face, 2023e). The recommended search turned out to not be suitable even though 312 models were hosted in March 2023 with this tag, because several of the most frequent downloaded models found via the search options described in the two previous paragraphs, were not listed. Therefore, search for the term "nli" alone was not regarded any further.

In order to find models for the current use-case, the description of the *Zero-Shot Classification* models with German language was checked first.

Further, the most popular models with English, multi-lingual and no language tag were checked. Several of the models checked had a sparse model description or did not have many monthly downloads. The latter can be interpreted as popularity and it can be assumed that better performing models are more popular than worse performing ones. The same procedure was pursued for the task tag *Text Classification*. For this tag, most models among the top downloaded models had a goal not suitable for the current purpose because they were e.g. about sentiment classification or did not support more than three classes as outcome. To determine NLI-suitable models among them, popular NLI data sets that were already found among the *Zero-Shot Classification*-models were used in filtering.

After reviewing the model charts including model objective and training data description, considering monthly model downloads, model size in GB, date of last maintenance, seven models were assessed to be suitable for the current classification task. Among them were two models with a comparable size in GB by the same author. To avoid comparing similar models, from those two models the model that was trained on more NLI data sets was chosen. Finally, six models were chosen to test their classification potential on contract sentence parts in order to determine the topic (label) of the sentence part.

Some researchers found that larger models in general can be assumed to perform better compared to smaller ones (e.g. Zhong et al., 2021; Wei and Zhou, 2023; Devlin et al., 2018). This would lead into the direction of selecting larger models and assuming larger models perform better compared to smaller ones. Nevertheless, the literature contains contradictory results as well. Recent studies like e.g. Manikandan et al. (2023) describe that larger language models can perform even worse compared to smaller ones. One reason for this is that larger models are often extensively trained on specific tasks with certain distributions, leading them to become overly specific to those distributions or tasks (McCoy et al., 2019; Hendrycks et al., 2020). Another reason is that the NLI tasks itself can have biased schemes which models learn during training and the models do not have to apply generalization or inference in order to perform the task correctly (Gururangan et al., 2018). Therefore, it is assumed that model performance depends more on the data sets and model architecture than on the model size for the currently selected models in this thesis.

The six selected models are described in more detail in the subsection below.

5.2 Model Descriptions

roberta-large-mnli

RoBERTa has been developed by Facebook AI research group and is an improved training method of BERT (Bidirectional Encoder Representations from Transformers; Devlin et al., 2018, initially described above in Section 2.7.5), having the same architecture and amount of parameters as BERT but a larger training batch size and more training data (Liu et al., 2019). Due to the modifications to the BERT pre-training procedure, RoBERTa has shown increased task performance in several benchmark sets in comparison to BERT (Liu et al., 2019). Therefore, the name of the model became **R**obustly **o**ptimized BERT **a**pproach.

RoBERTa in general has been trained on a large corpus of English text data in a self-supervised fashion, which means the model was trained on unlabelled text data in order to obtain an understanding of the text structure (Liu et al., 2019). The *roberta-large* model has been trained on roughly 160 GB of English text data consisting of: Wikipedia, unpublished books, English news articles, web pages, stories (Facebook AI, 2019; Liu et al., 2019). The objective of RoBERTa training was masked language modeling (MLM), which is a method for bidirectional learning of sentence representation via randomly masked words in sentences.

The model that is trained explicitly on natural language inference, *roberta-large-mnli*, has roughly the same amount of parameters as the model *roberta-large* (355M) and can be retrieved via the public GitHub repository[37] or via Hugging Face[38].

Since *roberta-large* is case sensitive and trained on English language, those two characteristics hold for the *roberta-large-mnli* model as well (Facebook AI, 2019). The model *roberta-large-mnli* has been fine-tuned on the NLI corpus Multi-Genre Natural Language Inference (MultiNLI or MNLI; Williams et al., 2018), which is a popular NLI data set consisting of 433k English sentence pairs. Those sentence pairs are generated partially by human annotators in order to create an entailment data set with *premise* and *hypothesis* pairs.

[37]GitHub: `https://github.com/facebookresearch/fairseq/tree/main/examples/roberta`
[38]Hugging Face: `https://huggingface.co/roberta-large-mnli`

Given that this model has exclusively been trained on English text data, its performance on the sequences from German language purchase contracts will be an interesting aspect in the results investigation.

joeddav/xlm-roberta-large-xnli

XLM-RoBERTa is a multi-lingual model based on Facebook's RoBERTa model and is pre-trained on 2.5TB of filtered CommonCrawl data from 100 languages (Conneau et al., 2020). CommonCrawl data is data that has been crawled from webpages and is mixed with snapshots of the web. It consists of webpages in many languages and is processed (i.e. deduplication, language identification) so quality is sufficient for LLM training (Wenzek et al., 2019).

According to Conneau et al. (2020), XLM-RoBERTa has been the first *single* large model for many languages with good performance across all languages. The XLM-RoBERTa with its 561M parameters is suitable to perform masked language modeling, but is actually intended to be further trained (fine-tuned) on a downstream task like sequence classification (Hugging Face, 2020).

The Hugging Face platform currently offers more than 4k models with the tag "xlm-roberta", but only 19 models that have the tag for zero shot classification as well (last checked on 2023-07-26).

The two most popular[39] XLM-RoBERTa models fine-tuned for zero-shot classification on Hugging Face are specifically the ones described in this section (spanning this and the subsequent page). The first model is *joeddav/xlm-roberta-large-xnli*, where the Hugging Face contributor *joeddav* has fine-tuned XLM-RoBERTa on a combination of NLI data sets in 15 languages, including German. The fine-tuning has been conducted on the multi-lingual NLI corpus (XNLI; Conneau et al., 2018), which is crowd-sourced collection of 5k test and 2,5k pairs of the MultiNLI corpus (more detailed explanation of this data set in Section 2.8 and on the previous page) with translations to 14 languages. Since every premise can be combined with the corresponding hypothesis in the other languages, the amount of possible combinations sums up to more than 1.5M combinations (Conneau et al., 2018).

During fine-tuning of this model, the languages have been shuffled such that premise and hypothesis are from different languages (Davison). The model is rather large with its 560M parameters, in comparison to the mono-lingual model roberta-large-mnli described above.

[39]Most popular according to highest number of model downloads since model release. Reference date 2023-07-05.

vicgalle/xlm-roberta-large-xnli-anli

The second popular XLM-RoBERTa based model is the *vicgalle/xlm-roberta-large-xnli-anli*. It has 560M parameters, which makes it a rather large model with more than 2 GB. For the reason that the model chart (Gallego) is quite short with only few model details and no paper is linked, combinations of fine-tuning sets and precise training procedure are unclear. However, based on the model chart and its name, one can infer that the model has xlm-roberta-large as its base architecture and that it has been fine-tuned using the XNLI (Conneau et al., 2018) and the ANLI (Nie et al., 2020) data set.

The XNLI (Conneau et al., 2018) data set was also used for fine tuning of the model joeddav/xlm-roberta-large-xnli, which was explained above. The Adversarial NLI (ANLI Nie et al., 2020) data set used to fine tune the *vicgalle/xlm-roberta-large-xnli-anli* was published to have a more complex and difficult NLI data set compared to the data sets existing at that time (Nie et al., 2020). An issue with previous NLI data sets was, that new LLMs did already reach human-level performance within months after NLI data set publications (Nie et al., 2020). Therefore, ANLI was constructed to be more difficult, so it can be used for model fine-tuning and lead to an increase in performance on this and even other NLI data sets, like e.g. the MNLI (Williams et al., 2018). The data set has been iteratively generated with human-and-model-loops to create long difficult enough examples.

Furthermore, it can be noted that the ANLI data set contains English language only. Hence, the multi-lingual abilities of the model *vicgalle/xlm-roberta-large-xnli-anli* are obtained by the XLM training of the RoBERTa-large model and the fine-tuning on XNLI that set.

MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7

This model is suitable for NLI and zero-shot classification on 100 languages. The underlying base model here is the model *mDeBERTa-v3-base*, which was introduced by Microsoft as the best performing multi-lingual transformer model in December 2021[40](He et al., 2023).

DeBERTa is an enhancement of BERT and RoBERTa models that incorporates two innovative components: *disentangled attention* and an improved mask decoder (He et al., 2021). The main difference between BERT and DeBERTa lies in the input word representation. While its predecessors use a single vector to encode both content and position information, DeBERTa employs two separate vectors for this purpose: one for content and another for position. An introduction to positional encoding was given in Section 2.7.3.

[40]Best performance with base-size. Larger models with the suffix "large" can still perform better, but also have more parameters.

The essence of disentangled attention lies in its calculation of attention weights, which consider both content and relative positions of words. This approach is different from the traditional BERT, which focuses on content alone (He et al., 2021). However, a notable limitation of disentangled attention is that it does not take absolute positions of context words into account, potentially impacting prediction accuracy. To overcome this, DeBERTa introduces an advanced mask decoder that incorporates absolute positional information during masked language modeling (MLM). This enhancement serves to improve the model's performance.

Building upon the DeBERTa architecture, DeBERTaV3 refines the model architecture through an alternative training loss and a weight-sharing method. This refinement results in the DeBERTaV3-base outperforming models of similar sizes in NLI benchmark tests (He et al., 2023). Additionally, *mDeBERTa* has the same structure as DeBERTa but was additionally trained on a multi-lingual data set containing 100 languages (Microsoft). This multi-lingual data set, known as *CC-100*, is constructed from the Common Crawl project (Wenzek et al., 2020), broadening the model's linguistic capabilities and application possibilities across diverse language contexts.

The model *mDeBERTa-v3-base-xnli-multilingual-nli-2mil7*, which is used in this thesis, has the multi-lingual DeBERTa model as base and has been further fine-tuned on the multi-lingual NLI data sets XNLI (Conneau et al., 2018) and *multilingual-nli-26lang-2mil7* (Laurer et al., 2022). The latter data set was created on the base of popular NLI data sets like MultiNLI (Williams et al., 2018) and ANLI (Nie et al., 2020). There are more than 2M premise-hypothesis pairs in 26 languages with >100k pairs per language in that data set. Those sequence pairs also include English and German and were generated with open-source machine translation models. Like also described for the MultiNLI corpus, the languages of the text pairs in the *multilingual-nli-26lang-2mil7* data set were mixed and combined to facilitate model fine-tuning with different languages in premise and hypothesis.

With a total amount of 279M parameters, the model *MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7* is rather small compared to the other ones described above (Laurer).

facebook/bart-large-mnli

BART (Lewis et al., 2019), which has been published by the Facebook AI research group, is a denoising autoencoder that combines bidirectional and auto-regressive transformer architecture (**B**idirectional **A**uto-**R**egressive **T**ransformers). The first (bidirectional) component is known from BERT models (by Google), the latter one (auto-regressive) from GPT models (by OpenAI).

The BART model maps a corrupted text span, which is a text with mask symbols, to the original, complete text. The corrupted text is encoded with a bidirectional model and the likelihood of the original text is determined with an auto-regressive decoder. A difference between BART and BERT architecture is, that BERT has a bidirectional encoder as well, but without the auto-regressive decoder. Additional differences between the two models are that BART does not use a feed forward network, and the decoder layers have another attention connection to the last hidden layer of the encoder (Lewis et al., 2019).

The model fine-tuned on the NLI task is *facebook/bart-large-mnli*, which has a large version of BART model (Lewis et al., 2019) as a base and was further trained on the Multi-Genre Natural Language Inference (MultiNLI; Williams et al., 2018) corpus. A difference between BERT and BART architecture in NLI tasks is, that the end-of-sequence token is passed to both the BART encoder and decoder. Therefore, in contrast to BERT, the BART representation of the end-of-sequence token is used for sentences relation classification (Lewis et al., 2019).

Since the training and fine-tuning of this model exclusively utilized English text data (Lewis et al., 2019; Williams et al., 2018), the assessment of its performance on sequences extracted from German language contracts will be of interest.

valhalla/distilbart-mnli-12-1

This model is the distilled version of *bart-large-mnli* with a distillation technique called *No Teacher Distillation* (Patil). Distillation is a method of compressing the size of a transformer model. Usually, the knowledge from a large teacher model, e.g. BART-large, is transferred to a smaller student model. There are different techniques for distillation, for different LLM models – all having the purpose to reduce model size to save resources (Li et al., 2022). The *No Teacher Distillation* is different from usual distillation processes and the model *distilbart-mnli-12-1* has been the outcome of distillation experiments with the novel distillation method, applied on bart-large-mnli (Patil).

The model *distilbart-mnli-12-1* has been created with a copy of alternating layers from the baseline model bart-large-mnli and additional fine-tuning on the same data. There are several versions/modifications of this model, all performing similar and almost as good as the bart-large-mnli (Patil). The model author published not only this distilled model, but also 3 other. The smallest distilled model, *distilbart-mnli-12-1* with 223M parameters, has almost half the amount of parameters as the baseline model without distillation. The largest of these distilled models is *distilbart-mnli-12-9* with just 12% less parameters compared to the baseline model.

Both the model chart on Hugging Face[41] and the model information on GitHub[42] contain little information about the DistilBart models architecture. Since no publication was linked either, the model configuration was checked to elaborate the differences between the four DistilBart models published by user *valhalla* (Suraj Patil). All models, including the baseline model, have 12 encoder layers, which can be assumed to be the number 12 in the model names. The number of decoder layers varies and corresponds to the last number in Suraj Patil's DistilBart model names. The model chosen for zero-shot classification in this thesis has only one decoder layer.

Considering that the BART-large-mnli model was exclusively trained on English text data, the distilled model is exclusively trained on English text data as well. Its performance on German text sequences has to be and will be investigated. Including this model, half of the selected models were trained and fine-tuned on English data exclusively.

The models described above have different architectures and amounts of parameters. Table 6 gives an overview of the model size and publications, if any were given in the model charts. It is evident that the largest models are the two xlm-RoBERTa based models, whereas the smallest model is the DistilBART model.

Table 6: Models used for zero-shot classification. The amount of parameters is in the second column in Millions.

Model name	**#param**	**Reference**
roberta-large-mnli	355M	Liu et al. (2019)
joeddav/xlm-roberta-large-xnli	560M	XLM-RoBERTa: Conneau et al. (2020)
vicgalle/xlm-roberta-large-xnli-anli	560M	XLM-RoBERTa: Conneau et al. (2020)
MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7	279M	mDeBERTa-v3-base: He et al. (2023)
facebook/bart-large-mnli	407M	Lewis et al. (2019)
valhalla/distilbart-mnli-12-1	223M	BART-large-MNLI: Lewis et al. (2019)

The model characteristics outlined in this section were summarized and outlined to provide an overview of the common data sets used for fine-tuning and the most notable architectural features. However, for models with millions of parameters, this overview only scratches the surface of their architectural complexities. Therefore, the respective model papers are recommended for more in-depth insights into the characteristics of each individual model. Summing up, it can be stated that even though the LLMs selected for the current study are all transformer based, the architectural characteristics vary. Additionally, the fine-tuning procedures and data sets used probably have an impact on their performance on the outcome task.

[41]https://huggingface.co/valhalla/distilbart-mnli-12-1
[42]https://github.com/patil-suraj/distillbart-mnli

5.3 Overview of Tokenizers

Tokenizers (mentioned in Section 2.2 and 4.4) are fundamental in pre-processing text data for LLMs. They are algorithms breaking or segmenting text into text pieces, called tokens with a more complex fashion than just delimiting words at spaces, which has been previously described in Section 2. These tokenizers are trained themselves on a text corpus and learned the optimal tokenization based on the vocabulary (Jurafsky and Martin, 2023a). Since the LLMs selected for the current study are trained on text tokenized with different methods, this subsection covers a brief overview of tokenization algorithms.

The three most common tokenizers are:

- Byte-Pair Encoding (BPE; Sennrich et al., 2016)
- WordPiece (Schuster and Nakajima, 2012)
- SentencePiece (Kudo and Richardson, 2018), in combination with Unigram (Kudo, 2018) or Byte-Pair Encoding (BPE; Sennrich et al., 2016)

The BPE algorithm (Sennrich et al., 2016) is a rather simple representative of tokenizers. The algorithm has the logic of first separating the text into letters, where each last letter of a word has a special symbol, e.g. an underscore, to describe the end of the word (before the whitespace character). At this step, every letter is an entity (a token). In order to create larger entities, the algorithm iteratively checks the corpus of words for consecutive occurrences of letters (Jurafsky and Martin, 2023a). E.g. in a corpus that contains the words ‘letter’ and ‘lawyer’ among other words, these two words would be initially separated as: ‘l’, ‘e’, ‘t’, ‘t’, ‘e’, ‘r’, ‘_’ and ‘l’, ‘a’, ‘w’, ‘y’, ‘e’, ‘r’, ‘_’. The iterative step of joining letters to new entities contains in this example combining ‘e’ and ‘r’ to form ‘er’, and then replacing the single letters with the respective joined entity within the words. The new token is added to the vocabulary, which initially contains all letters of the text corpus words, and in the end, includes all letters and the defined tokens consisting of letter pairs detected in words.

In a text corpus, the BPE algorithm does not just start with arbitrary adjacent letters, but with those that occur adjacent to each other most frequently in the corpus. The tokenization algorithm counts the adjacent entities and merges them into new word tokens as often as determined by an algorithm parameter specifying this number (The parameter is often called k. For further details see Jurafsky and Martin, 2023a; Hugging Face, 2023a).

Once the BPE tokenizer has learned the merges to be performed based on the letters in the training corpus, it can be applied to test text data and perform the learned merges on that data. In a real-world setting, the BPE algorithm iterates thousands of times on a very large input corpus. As a result, most words are represented as full symbols (with all letters), and only the very rare or unknown words are represented by their word parts (Jurafsky and Martin, 2023a).

A variant of the BPE algorithm is *WordPiece* (Schuster and Nakajima, 2012). The WordPiece algorithm also starts with a vocabulary consisting of all the letters of the words in the training text corpus and merges letters iteratively to larger tokens. In contrast to BPE, the selection of entities to merge is not by frequency, but by the probability of an underlying LLM. Whenever a new token (merged from letters of shorter tokens) is increasing the LLM probability, the token is added to the vocabulary (Jurafsky and Martin, 2023d). The process continues iteratively, identifying and merging tokens that enhance the LLM's understanding until a stopping criterion is met, such as reaching a predefined vocabulary size or no further improvement in LLM probability.

The *Unigram* algorithm (Kudo, 2018) is another method for tokenization. Unlike other algorithms like BPE or WordPiece, Unigram starts with a large vocabulary and progressively reduces it to a smaller size. This process involves iteratively evaluating the impact of removing each symbol from the vocabulary based on a defined loss function, typically the log-likelihood, over the training data. Symbols that contribute the least to the loss are removed iteratively until the desired vocabulary size is reached. However, the algorithm always retains base characters to ensure any word can still be tokenized. Probabilities are assigned to tokenizations during training and in contrast to BPE and WordPiece the algorithm is not based on merge rules. This makes tokenization more flexible, offering the most likely tokenization, but also having the option for other tokenizations based on their probabilities (Hugging Face, 2024). LLMs usually not use the Unigram algorithm directly, but together with the *SentencePiece* algorithm.

The *SentencePiece* algorithm (Kudo and Richardson, 2018) is a method for tokenizing text, without the limitation of assuming words are separated by spaces. Traditional tokenization algorithms (like BPE) rely on spaces to separate words, which does not work well for languages that do not use spaces in this way, e.g. Chinese. SentencePiece treats the input text as a raw stream, including spaces in the set of characters it considers. It then applies algorithms like Unigram or BPE to construct a vocabulary of subword units (Hugging Face, 2024).

So far, tokenizers were referred to as methods that break down raw text into individual tokens, which are then converted into numerical representations that a language model can process (Jurafsky and Martin, 2023g). The famous LLM BERT (Devlin et al., 2018), which has almost the same architecture as the originally proposed transformer architecture (see Section 2.7.5 Vaswani et al., 2017), is using the WordPiece algorithm for that matter. This means that BERT, but also its descendants, are not trained on words but on subword tokens. Therefore it has to be considered that in application of any pre-trained LLM, the input text is first tokenized to subwords and then the numeric representation is further processed by the LLM.

Some well-known tokenizers have been already explained in this section. However, the tokenizers used by the models listed in Table 6 are (according to Hugging Face, 2024):

- RoBERTa, DeBERTa, and BART based models use the *BPE* algorithm, with a modified handling of whitespaces (Liu et al., 2019; Lewis et al., 2019; He et al., 2021; Hugging Face, 2023a)
- The distilBART based model uses the pretrained tokenizer of the BART model, which is a modified *BPE* algorithm (Patil, 2020)
- DeBERTa-v2 and DeBERTa-v3 based models use *SentencePiece* (He et al., 2023; Microsoft)
- The xlm-RoBERTa based models use *SentencePiece* tokenization with BPE (Hugging Face, 2020, 2023d)

Because the tokenization step is taking place *before* the application of the LLMs, the different tokenizers implemented and used by default by the models used in the current study were briefly explained at this point for comprehensiveness. For a more detailed explanation of the main tokenization methods refer to text books like Jurafsky and Martin (2023a) and the referred papers by the original authors.

5.4 Inference with the Transformers Pipeline

This section serves the purpose to explain the interface used for application of the models described in Section 5.2.

Overview of the Pipeline

To evaluate the quality of the LLMs in zero-shot sequence classification of the desired text, several open-source models were chosen (the selection process is described in Section 5.1). Since all of the pre-trained models were obtained from the Hugging Face platform, it is convenient to use the platform's open source Python library containing a processing pipeline. It combines the deep learning frameworks PyTorch, TensorFlow, and Jax to ease the training, fine-tuning, usage and application of the models (Hugging Face, 2023b; Wolf et al., 2020).

The Hugging Face framework offers different pipelines depending on the task performed, which can be NLP tasks, image related tasks, audio related tasks or multi-modal tasks combining several types of input processing (Hugging Face, 2023e). The pipelines are available in the *transformer* package (Wolf et al., 2020), which contains a wrapper method, called *'pipeline'*. On initialization, this wrapper is combining the different pipelines and selecting the suitable pipeline and processing methods depending on the string parameter *'task'*. The 'task' determines the type of pipeline being created and which task specific pipeline is returned by the pipeline wrapper method. The *'task'* identifier "zero-shot-classification" returns the task specific "ZeroShotClassificationPipeline" and is used in the current thesis (see step 14 of Algorithm 1 in Section 4.3).

The task specific processing pipelines, e.g. ones specific to NLP tasks, need additional arguments to perform the desired tasks. Two main parameters are the *'model'*, which makes the predictions, and the *'tokenizer'*, which separates the text into tokens. The *'model'* parameter specifies the pre-trained model to be used for the task. This could be a model identifier (e.g. "facebook/bart-large-mnli", which is one of the models from Table 6) or a path to a locally stored model. Hugging Face provides many pre-trained models that can be used for various NLP tasks (and also other as well as multi-modal tasks). If no model is given, there is a default model selected, depending on the given 'task'.

Underneath, the *pipeline*-function handles the loading of the explicitly given or appropriate tokenizer for the specified model. The tokenizer is responsible for converting input text into numerical tokens that the model can understand. Different models can require different tokenization strategies, so the appropriate tokenizer is usually selected based on the specified model.

Even though every model hosted on Hugging Face has a default tokenizer, which is automatically used in the pipeline, different pre-trained and also self-trained tokenizers can be used to process the text before inference via LLMs. The text data used for the current study is from the German legal domain, which is why tokenizers from this field were investigated. Researchers from the English legal NLP field have trained tokenizers for the purpose of processing legal texts without loosing semantic expressions due to sub-word splitting (Ying and Habernal, 2022). Nevertheless, since no tokenizer trained on a legal corpus of German purchase contracts or comparable data was found in the research for the current work, the tokenizers used in the present thesis are such which are pre-trained on general text corpora and the default Hugging Face implementation.

Once the pipeline is created, it can be used to process input text for the specified task. The pipeline handles the entire workflow, from tokenization to the model results, depending on the task type.

In summary, the *pipeline*-function from Hugging Face's *transformers* library creates a simplified interface for executing various NLP tasks using pre-trained models, handling tasks such as model loading, tokenization, and inference internally.

Application of the Pipeline for Zero-shot Classification

In Section 2.8, the natural language inference (NLI) has already been described. Models that have been fine-tuned on an NLI task can be used in the Hugging Face transformers pipeline (Hugging Face, 2023e) via the task identifier *"zero-shot-classification"*. The *transformers* package version used was 4.18.0.

The models used were explicitly trained on sequence classification with NLI tasks. The way those models assess the sequences is with the given premise-hypothesis pairs (see Section 2.8 for further details of NLI). For every given sequence, which is the so called *premise*, the model is called to assess if the hypothesis is entailing. Entailment means that the model assesses that the *premise entails the hypothesis*. Hence, the pre-trained zero-shot model is presented with several hypotheses per premise, always in the form of pairs. In the transformers pipeline, the logit for entailment is taken as the value of the class's validity. The output of the model contains all the class labels combined with the logit values for entailment (Hugging Face, 2023e). In the present thesis, the class with the highest logit value is regarded as the model outcome / classification result.

An illustration of the workflow of classification of a single text sequences is given in the Figure 17 below. The sequence displayed in this figure is already known from previous sequence examples (like Figure 15 from Section 3.2).

Like explained in the Algorithm 1 (see p. 62), after the cleaning and sentence splitting, the classification of the sequences follows. Therefore, the sequence, hypothesis as well as the target classes are passed to the model for processing. Every text sequence is regarded as premise in the premise-hypothesis pairs. The hypothesis templates are used together with the classes one by one, which is shown in Figure 17. The figure illustrates that the LLM assesses if the hypothesis 'This example is *Wohnfläche*' logically follows the text sequence (premise).

In the illustration, only one hypothesis template (H_0, the default) is shown. For every model, several hypothesis templates are used (see Section 6.2).[43]

In the Assessment and Evaluation section (see p. 90ff) it will be determined which model with which hypothesis performed best in inferring the context of the area in the zero-shot approach. In order to define what is "best" a few terms regarding the metrics are covered in the next subsection.

Summing up it can be stated that the initialized and task specific pipeline is used with the arguments *sequence* to classify, the *labels* to choose from and the classification *model*. For

[43]Table 8 (on p. 93) contains an overview of the models and hypotheses that are applied and evaluated.

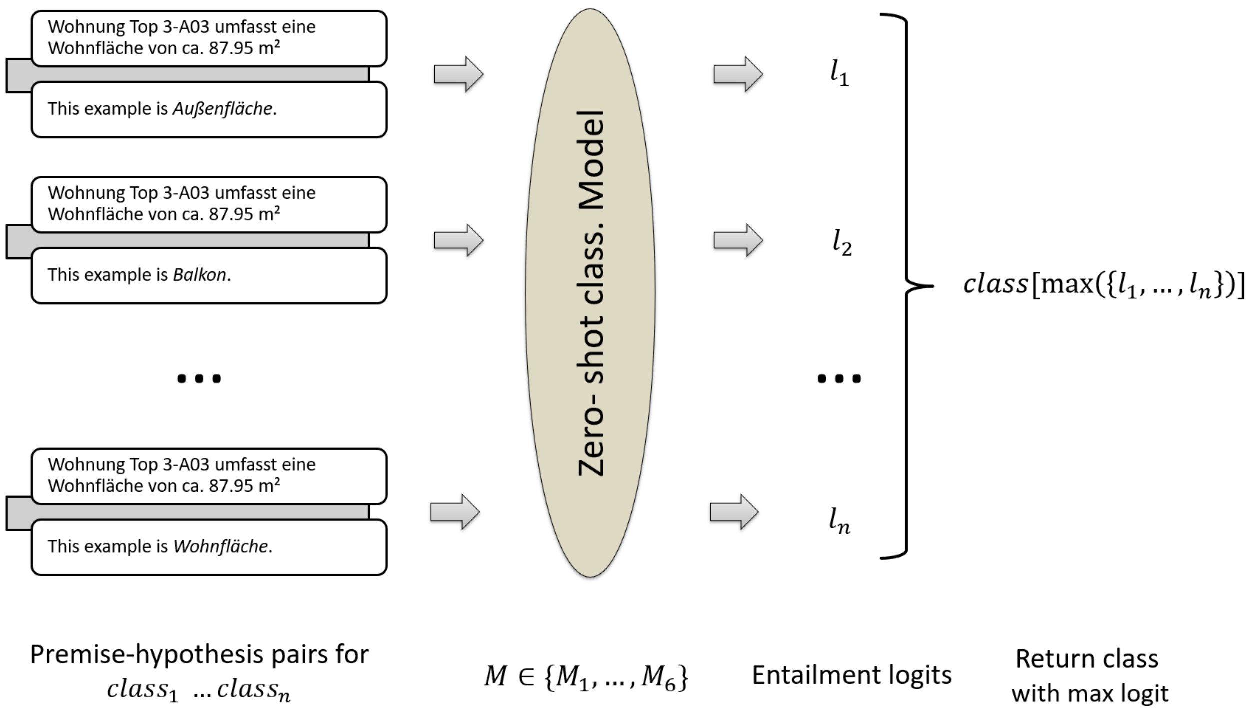

Wohnung Top 3-A03 umfasst eine Wohnfläche von ca. 87.95 m²
This example is *Außenfläche*.
Wohnung Top 3-A03 umfasst eine Wohnfläche von ca. 87.95 m²
This example is *Balkon*.
Wohnung Top 3-A03 umfasst eine Wohnfläche von ca. 87.95 m²
This example is *Wohnfläche*.
Zero- shot class. Model
l_1
l_2
l_n
$class[\max(\{l_1, \ldots, l_n\})]$
Premise-hypothesis pairs for $class_1 \ldots class_n$
$M \in \{M_1, \ldots, M_6\}$
Entailment logits
Return class with max logit

Figure 17: Illustration of the Hugging Face *transformer* (Wolf et al., 2020) *pipeline*-function approach classifying one German sequence (engl. Apartment Top 3-A03 has a living area of approx. 87.95 m²) with hypothesis template H_0. The entailment is assessed by the zero-shot model for every class 1...n. The class with the highest entailment logit is used as classification result.

the model, one of the transformer models suitable for zero-shot classification is used at a time. The full list and description of models used in this thesis has already been given above (see Section 5.2). The classification pipeline returns a dictionary containing scores and labels sorted descending with the class having the highest entailment logit first and lowest one last, for the respective sequence.

5.5 Rule-based Approach

Apart from the zero-shot NLI classification approach explained above, a rule-based approach was applied to the pre-processed sequences in order to establish a straightforward classification method for comparison. To ensure comparability between the rule-based results and the zero-shot results, all pre-processing steps up to the classification (Steps 1-13 of Algorithm 1) were kept identical. This means the rule-based classifier takes as input the same sequences that were obtained after all cleaning and filtering steps (see Section 4.4).

If the classification is to be solely rule-based, Algorithm 5 is applied on the sequence. The outcome is a single class, either the one where the pattern matched or the class *other*.

Taking into account that the rule-based approach should rule out that the LLMs performing zero-shot classification rely solely on pattern matching, the patterns for the rule-based classification were exactly the classes used for the zero-shot classification as well. This implies that for fine-grained label classification, the rule-based classifier attempts to match a class label within the sequence. The matching process involves an ordered list of patterns and can be described as follows:

Algorithm 5 Classify sequences solely rule-based

```
1: sequence ← clean and ready to classify sentence part containing an area and context
   information
2: If sequence is empty return None
3: patterns = {                                              ▷ Define them to use below
       'Wohnfläche': r'Wohnfläche',
       'Gesamtfläche': r'Gesamtfläche',
       'Zimmer': r'Zimmer',
       'Außenfläche': r'Außenfläche',
       'KFZ': r'KFZ',
       'Garten': r'Garten',
       'Grundstücksfläche': r'Grundstücksfläche',
       'Balkon': r'Balkon',
       'Terrasse': r'Terrasse',
       'Loggia': r'Loggia',
       'Nutzfläche': r'Nutzfläche'
       }
4: for label, pattern in pattern.items() do
5:     if pattern detected in sequence then
6:         return category
7:     end if
8: end for
9: return 'Sonstige'               ▷ Whenever no pattern matched, else-condition (default).
```

The Algorithm 5 takes the sequence as input and iterates through each class label, attempting to find a direct match within the sequence. If a match is found, the algorithm assigns the corresponding class label to the sequence. In cases where no match is found for any class label, the sequence is classified as *Sonstige* (engl. other). It can be assumed that for most fine-grained classes the pattern matching should perform well since the preceding text processing steps contained a lot of term and sequence harmonization steps.

To gain a clearer understanding of the labels assigned to sequences using the rule-based approach, the sequences detailed in Section 4.7 are described below, along with the resulting labels:

- *"einer Terrasse von rund 15 m^2 verbunden ist"* (engl. connected (to) a terrace of around 15 m^2).

  In the manual ground truth, the sequence has the fine-grained label *Terrasse* (engl. terrace) and the aggregated label *Sonstige* (engl. other). With the rule-based approach the sequence would get the same class label result as in the ground truth because the pattern 'Terrasse' matches the sequence directly.

- *"Wohnung Top 55-D16a im Ausmaß von ca. 27.55 m^2"* (engl. apartment top 55-D16a in the extent of 27.55 m^2).

  In the manual ground truth, the sequence is labelled as *living area.* However, with the rule-based approach, the label *Sonstige* (engl. other) would be assigned because the term and pattern 'Wohnfläche' does not match directly.

6 Assessment and Evaluation

The current section contains an overview of the metrics (Section 6.1) used to evaluate the models, model abbreviations that are used when reporting the results (Section 6.2), the results and interpretation of the zero-shot models (Section 6.3) as well as the results and interpretation of the rule-based results (Section 6.4). Furthermore, in Subsection 6.5 a comparison of the best zero-shot models to the rule-based classification is reported.

6.1 Evaluation Metrics Overview

In the field of model evaluation, whenever the aim is to evaluate and compare different classification models, the use of performance indicators is recommended (Grandini et al., 2020). It can be stated that the confusion matrix, which is a cross table recording the true and predicted classification, encloses the most relevant information about the algorithm and classification performance. Therefore, confusion matrices will be reported for the best models and the rule-based classification (see below Sections 6.3 and 6.4). The data consisting of manual classifications is used to accomplish the task of results evaluation and was described in Section 4.8.

Metrics based on the confusion matrix are common in the evaluation of classification tasks. The most common ones are namely: precision, recall, accuracy, and F1-score and their formulas are as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\text{Grand Total}}$$

When it comes to multi-class cases, a multi-class measure of the metrics is needed. The F1-score in this case can be distinguished in *micro* and *macro* F1-score.

For the class specific precision and recall measures, confusion matrices focusing on single classes are required. The true positives (TP) are the cases classified correctly with respect to the class k. The false positives (FP) and false negatives (FN) are the wrongly classified cases column and row wise[44], and the true negatives (TN) are all the other cases for a class k.

The *micro* F1-score (see Equation 1) consists of the micro average precision p_{mi} and micro average recall r_{mi}. Since all cases, without considering the size of classes, are considered together in the micro averaging, the formulas are and can be rewritten as follows:

$$p_{mi} = \frac{\sum_{k=1}^{k} TP_k}{\sum_{k=1}^{k} \text{Total Column}_k} = \frac{\sum_{k=1}^{k} TP_k}{\text{Grand Total}}$$

$$r_{mi} = \frac{\sum_{k=1}^{k} TP_k}{\sum_{k=1}^{k} \text{Total Row}_k} = \frac{\sum_{k=1}^{k} TP_k}{\text{Grand Total}}$$

Considering that the F1-score is the harmonic mean of precision and recall and both p_{mi} and r_{mi} are the same values, the micro averaged F1-score is the same value as well (Grandini et al., 2020). All three micro averaged metrics are equal to the accuracy.

$$r_{mi} = p_{mi} = F1_{micro} = \frac{\sum_{k=1}^{k} TP_k}{\text{Grand Total}} = accuracy \tag{1}$$

The *accuracy* gives more importance to big classes, because it considers all the units combined (Grandini et al., 2020). For the accuracy a low performance on small classes is less important compared to a low performance on large classes, because the number of cases belonging to the rare classes is small compared to the size of the overall data set.

Apart from that, another averaged metric for multi-class cases is the *macro* averaged F1-score. At this point, it must be noted that for the calculation of the macro F1-score there exist two different approaches in the literature. The formula described below is recommended by Opitz and Burst (2021) to avoid bias.

$$F1_{macro} = \frac{1}{n}\sum_k F1_k = \frac{1}{n}\sum_k \frac{2p_k r_k}{p_k + r_k} \tag{2}$$

The macro F1-score is basically the arithmetic mean over the F1-scores for each individual class k. The macro precision and recall can also be computed as arithmetic means of scores per class. This is the same computation also used by the scikit-learn developers (2023) functions used in this thesis.

Since all classes are equally weighted in the $F1_{macro}$ and a high $F1_{macro}$ score indicates good performance on all classes and is suitable for imbalanced data sets, the $F1_{macro}$ will

[44]Assuming the columns are the predicted and the rows are the actual, true classifications.

be favored against the *accuracy* ($= F1_{micro}$) in case model performance is not equally good for both metrics and the best model is not clearly the best in regard of accuracy and macro metrics.

The performance metrics accuracy, macro averaged precision, recall, and F1-score will be reported both for the zero-shot classification models and the rule-based classification in Section 6.

In order to have a baseline model to compare the metrics of the zero-shot and rule-based classification to, the metrics will be reported for the majority classes as well. A baseline model helps especially in cases of unbalanced multi-class problems to establish the expected performance level using a simplistic approach (Prakash, 2023).

6.2 Abbreviations and Combinations for Models and Hypothesis Templates

The models used in the experiments were already described in Section 5.2 and are listed below in Table 7, with abbreviations to ease readability of the paragraphs describing and comparing results.

Table 7: Zero-shot models from Hugging Face project that were used in the experiments. The abbreviation column shows the numbers that are used for referring to the models.

Abbreviation	**Model name**
M_1	roberta-large-mnli
M_2	joeddav/xlm-roberta-large-xnli
M_3	vicgalle/xlm-roberta-large-xnli-anli
M_4	MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7
M_5	valhalla/distilbart-mnli-12-1
M_6	facebook/bart-large-mnli

Since different hypothesis templates showed varying classification performance in previous research (e.g. in Yin et al., 2019; Sainz and Rigau, 2021), the current research investigates several different hypothesis templates as well. Every model listed in Table 7 was used with different hypothesis templates. Those ten hypothesis templates are either in English language ($count = 4$) or in German language ($count = 6$) and are shown in Table 8. The templates were inspired by the literature (Yin et al., 2019; Sainz and Rigau, 2021), but apart from H_7 and the default H_0, do not cover the exact same wording.

Table 8: Hypothesis templates used in the models. The first column contains a short form for the templates to refer to them with an abbreviated way. In the second column, the term *{label}* is replaced by the labels respectively when the template is used as hypothesis. The third column contains notes by the author and the fourth column the language of the hypothesis.

Abbrev.	**Hypothesis**	**Note**	**Lang.**
H_0	This example is {label}.	baseline in transformer codes	engl
H_1	Dieses Beispiel ist {label}.	German translation of H_0	ger
H_2	This is about {label}.		engl
H_3	Es geht um {label}.	German translation of H_2	ger
H_4	This is {label}.		engl
H_5	Das ist {label}.	German translation of H_4	ger
H_6	Das ist {label}	same as H_5, but without dot	ger
H_7	The domain of the sentence is about {label}.	performed best in Sainz (2021)	engl
H_8	Dieses Beispiel handelt von {label}.	similar, but longer as H_1	ger
H_9	Die Fläche gehört zu {label}.	containing the ger. area word	ger

It is recommended by several Hugging Face model authors (e.g. Davison; Cusumano) to use a language specific hypothesis template when the premise text is primarily a specific language or the model was trained on a certain language because better classification results can be achieved. However, several models used in this research were trained on multilingual data sets, and/or a combination of languages in the text pairs was already integrated into the model training process. This was done to facilitate the understanding of English hypotheses regardless of the language used in the premise (e.g., in model M_4 by Laurer, see Section 5.2 for further details). Hence, for some hypothesis templates used in the current thesis both languages were used with similar wording.

6.3 Evaluation of the Zero-shot Sequence Classification

To identify the best model for classifying sequences from real estate purchase contracts, the 668 sequences described in Section 4.8 were classified by the models. In this section, the performance of the different models on the classification task is reported. First, the sequence classification is reported in regard of the fine-grained labels and second, it is reported in regard of the aggregated labels.

Like pointed out in Section 6.1, metrics of a baseline model ease interpretation of the metrics obtained by the models. The metrics reported in Table 9 contain accuracy, precision, recall and F1 measure for a fixed class, assuming a classifier always predicting that class. The metrics are reported for the three largest majority classes using fine-grained labels and the two majority classes using aggregated labels.

Table 9: Performance of baseline models predicting one of the three most frequent classes. Reported metrics are the *accuracy*, macro averaged ($_{ma}$) precision (p), recall (r) and F1 measure. *Note that the third column is omitted in the right block because 'plot area' only exists among the fine-grained labels. The aggregated class is 'other'.

	baseline class (fine-grained)			**baseline class (aggregated)**	
metrics	**plot area***	**other**	**living area**	**other**	**living area**
accuracy	0.23	0.21	0.19	0.66	0.19
p_{ma}	0.02	0.02	0.02	0.13	0.04
r_{ma}	0.09	0.09	0.09	0.20	0.20
$F1_{ma}$	0.03	0.03	0.03	0.16	0.06

In Table 9, the first row displays the accuracy of a simplistic (fictive) classifier that consistently assigns the same label. It also outlines the distribution of classifications for the most frequent labels. In the ground truth of the 668 sequences, 23% were labeled as *plot area*, 21% as *other*, and 19% as *living area*. However, since *plot area* is grouped with other classes under *other* in the aggregated labels evaluation, the proportion of sequences belonging to the *other* class is almost 2/3.

6.3.1 Results with Fine-grained Labels

Table 10 displays the metrics of the different hypotheses and models for the fine-grained labels. The maximum accuracy and F1-scores per model are marked in bold, and the average F1-scores per model are provided in the right-most column. The F1-scores varied a lot between the models and hypotheses, with a range of $0.45 - 0.72$ for the *accuracy* ($= F1_{micro}$) and a range of $0.17 - 0.63$ for the $F1_{macro}$. The performance metrics of the zero-shot models exceeded all baseline metrics of a fictive model predicting the majority class (see Table 9).

Table 10: Metrics describing the fine-grained classification results. *accuracy* and macro averaged ($_{ma}$) precision (p), recall (r) and F1 measure are reported for the six models and the ten hypotheses. The average (avg) of the F1-scores per hypothesis are in the last two rows.

		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M1	$accuracy$	0.64	0.62	0.59	0.61	0.63	**0.65**	0.63	0.60	0.63	0.63	**0.62**
	p_{ma}	0.64	0.62	0.59	0.63	0.65	0.63	0.62	0.60	0.65	0.60	
	r_{ma}	0.72	0.67	0.65	0.67	0.69	0.70	0.68	0.67	0.71	0.70	
	$F1_{ma}$	0.61	0.59	0.54	0.57	0.60	**0.62**	0.60	0.55	0.59	0.58	**0.58**
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M2	$accuracy$	0.55	0.50	0.46	0.51	0.45	0.46	0.45	0.49	0.58	**0.62**	0.51
	p_{ma}	0.67	0.55	0.17	0.52	0.15	0.25	0.14	0.44	0.69	0.64	
	r_{ma}	0.38	0.33	0.23	0.32	0.21	0.23	0.21	0.29	0.52	0.60	
	$F1_{ma}$	0.37	0.35	0.20	0.31	0.17	0.19	0.17	0.26	0.44	**0.53**	0.30
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M3	$accuracy$	0.58	0.57	0.46	0.63	0.53	0.53	0.51	0.48	0.58	**0.68**	0.56
	p_{ma}	0.61	0.59	0.52	0.60	0.62	0.62	0.63	0.51	0.60	0.58	
	r_{ma}	0.63	0.61	0.40	0.69	0.56	0.57	0.50	0.42	0.63	0.71	
	$F1_{ma}$	0.54	0.52	0.34	0.56	0.46	0.49	0.43	0.38	0.52	**0.59**	0.48
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M4	$accuracy$	0.49	0.48	0.50	0.51	0.49	0.46	0.45	0.48	0.53	**0.55**	0.49
	p_{ma}	0.61	0.56	0.64	0.71	0.60	0.60	0.50	0.56	0.60	0.57	
	r_{ma}	0.50	0.49	0.52	0.54	0.49	0.41	0.38	0.52	0.58	0.64	
	$accuracy$	0.41	0.40	0.42	0.46	0.40	0.35	0.32	0.40	0.49	**0.55**	0.42
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M5	$accuracy$	0.60	**0.72**	0.59	0.61	0.56	0.68	0.57	0.59	0.62	0.65	**0.62**
	p_{ma}	0.62	0.70	0.62	0.63	0.62	0.69	0.76	0.59	0.57	0.62	
	r_{ma}	0.67	0.66	0.65	0.65	0.62	0.68	0.68	0.64	0.62	0.67	
	$F1_{ma}$	0.54	**0.61**	0.54	0.56	0.52	**0.61**	0.59	0.54	0.53	0.60	0.56
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M6	$accuracy$	0.61	**0.72**	0.58	0.61	0.60	0.62	0.57	0.60	0.54	0.68	0.61
	p_{ma}	0.62	0.65	0.64	0.60	0.65	0.64	0.56	0.60	0.62	0.63	
	r_{ma}	0.66	0.70	0.63	0.65	0.67	0.68	0.62	0.65	0.60	0.70	
	$F1_{ma}$	0.57	**0.63**	0.55	0.55	0.57	0.59	0.51	0.56	0.50	0.62	0.56
	$avg_{accuracy}$	0.50	0.51	0.45	0.50	0.46	0.49	0.45	0.46	0.50	**0.55**	0.49
	avg_{F1ma}	0.43	0.44	0.37	0.43	0.39	0.41	0.37	0.38	0.44	**0.49**	0.42

This implies that even the least performing zero-shot model outperformed a classifier that consistently predicts the same class for all cases.

The highest *accuracy* was achieved with H_1 and the models M_5 and M_6 whereas the highest $F1_{macro}$ was achieved with H_1 in M_6. The hypothesis H_1 : *"Dieses Beispiel ist {label}."* is the German translation of the English default hypothesis H_0 : *"This example is {label}."* (see full list of hypothesis templates in Table 8 and the models in Table 7).

Even though M_6 has been trained on English data only, the model achieved the highest $F1_{macro}$ and additionally had the highest average $F1_{macro}$ in combination with the lowest standard deviation ($M = 0.58$, $SD = 0.03$) across the ten hypotheses used. Hence the results of M_1 can be seen as rather stable across the different hypotheses.

The maximum accuracy and macro F1-score per hypothesis are reported in the last two rows of Table 10. Even though the best classification in terms of F1-scores using fine-grained labels was achieved with M_6 in hypothesis H_1, the hypothesis with the on average highest F1-score was H_9. The hypothesis H_9 was a German sentence and was the only one referring to the German word for area.

In general, the two Bart-based models M_5 and M_6 performed better with H_1 compared to the other hypothesis. In comparison to that the RoBERTa-based models M_2, M_3 and the DeBERTa-based model M_4, which were all three trained on the XNLI data set, performed best with H_9. The model having the highest average F1-score across all ten hypotheses was M_1, which is RoBERTa-based and performed best on H_5 in comparison to the other hypotheses.

Another interesting observation is that the two hypotheses H_5 and H_6, which only differed by the existence of a dot as last character, do not result in the same metric scores across all models. The hypothesis having the dot at the end of the sentence, H_5, lead to higher F1-scores compared to the other hypothesis in all six models. Hypothesis template H_5 used in model M_1 application even achieved the second largest macro F1-score with the fine-grained labels. Moreover, among the ten selected templates, H_6 is the only hypothesis template without a dot after the label.

On average, the F_1 scores achieved with the German hypotheses were higher compared to the English hypotheses (see Table 11). This trend can be observed in most hypotheses where a German and very close English translation were used. Especially in case of H_2 and H_3, the German hypothesis H_3 performed better, in terms of F1-scores, than the English one over all models. For the hypotheses H_4 and H_5 the trend was similar, except in case of M_4, where the German H_5 performed worse than the English equivalent H_4. Nevertheless, one can see mixed trends between the default hypothesis H_0 and the German translation H_1. For the models $M_1 - M_4$ the English default hypothesis leads to higher F1-scores whereas

Table 11: Average F1-scores of classification results (with fine-grained labels) per language.

	Language	
	ger.	engl.
$avg_{accuracy}$	0.50	0.47
$avgF1_{macro}$	0.43	0.39

the Bart-based models M_5 and M_6 achieved higher F1-scores with the German (translation) compared to the English default hypothesis. All in all, it can be concluded that the comparison of the hypotheses demonstrates that the choice of a German hypothesis is more suitable in combination with German text sequences for most of the models used.

Summing up, the two best performing combinations of model and hypothesis template, in terms of macro averaged F1-score, and using the *fine-grained* labels, were:

- M_6, *facebook/bart-large-mnli*, in combination with hypothesis template H_1, *"Dieses Beispiel ist {label}."*, with a $F1_{macro} = 0.63$
- M_1, *roberta-large-mnli*, in combination with hypothesis template H_5, *"Das ist {label}."*, with a $F1_{macro} = 0.62$

Interestingly, these two LLMs performed best among the six selected models, despite being exclusively trained on English text data.

6.3.2 Results with Aggregated Labels

Table 12 shows the metrics of the different hypotheses and models for the aggregated labels. The logic of aggregation of the labels is described in Section 3.3.2. Like in the table showing the fine-grained results (Table 10), the maximum accuracy and F1-scores per model are marked bold and the average accuracy and F1-scores per model are given in the right-most column in Table 12.

The F1-scores, using the aggregated labels, varied a lot between the models and hypotheses, with a range of $0.64 - 0.88$ for the *accuracy* ($= F1_{micro}$) and a range of $0.43 - 0.87$ for the $F1_{macro}$. The performance metrics of the zero-shot models exceeded most baseline metrics of a fictive model predicting the majority class (see Table 9). Nonetheless, given that 66% of the aggregated labels are assigned to the majority class *other*, two zero-shot models (M_3, M_5) either performed equally or less accurately with at least one hypothesis compared to the baseline model that predicts only the majority class. However, the precision, recall and macro averaged F1-score were all a lot better compared to the baseline model. This implies that the best model-hypothesis combination for every zero-shot model outperformed a classifier that consistently predicts the same class for all cases.

Overall, the F1-scores were higher with the aggregated labels as with the fine-grained labels. This was expected since when there are less labels, there is less chance for misclassification (especially because labels are combined). Both, the highest *accuracy* and the highest $F1_{macro}$ were achieved with M_1 in combination with hypothesis H_0. An equally high $accuracy = 0.88$ was also achieved with M_1 in hypothesis H_9, but this result got a lower $F1_{macro}$ compared to the result of M_1 with H_0 (see Table 12). The second highest $F1_{macro}$ was achieved with M_6 using H_9.

It has to be noted that M_1 achieved not only the highest $F1_{macro}$ but also the highest average $F1_{macro}$ in combination with the lowest standard deviation ($M = 0.81$, $SD = 0.04$) across the ten hypotheses used per model. Therefore, the results of M_1 can be seen as rather stable across the different hypotheses with the aggregated labels.

The maximum accuracy and macro F1-score per hypothesis are reported in the last two rows of Table 12. Even though the best classification in terms of F1-scores was achieved with M_1 in the default hypothesis H_0, the hypothesis with the on average highest F1-score was H_9. The hypothesis H_9 is a German sentence referring to the German word for area (*"Die Fläche gehört zu {label}."*) and showed already for the fine-grained labels the highest overall F1-scores. The full list of hypothesis templates and models were reported above in Table 8 and Table 7, respectively.

Table 12: Metrics describing the aggregated classification results (aggregated labels). *accuracy* and macro averaged ($_{ma}$) precision (p), recall (r), and F1 measure are reported for the six models and the ten hypotheses. The average (avg) of the F1-scores per hypothesis are in the last two rows.

		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M1	$accuracy$	**0.88**	0.82	0.81	0.76	0.85	0.86	0.87	0.82	0.68	**0.88**	**0.82**
	p_{ma}	0.83	0.77	0.73	0.74	0.79	0.80	0.80	0.74	0.75	0.78	
	r_{ma}	0.93	0.85	0.84	0.88	0.93	0.92	0.88	0.84	0.88	0.87	
	$F1_{ma}$	**0.87**	0.79	0.76	0.78	0.84	0.85	0.83	0.76	0.76	0.81	**0.81**
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M2	$accuracy$	0.82	**0.83**	0.78	0.76	0.79	0.75	0.76	0.77	0.81	**0.83**	0.79
	p_{ma}	0.75	0.66	0.46	0.70	0.44	0.47	0.40	0.69	0.73	0.72	
	r_{ma}	0.69	0.66	0.54	0.58	0.47	0.51	0.47	0.63	0.76	0.73	
	$F1_{ma}$	0.65	0.65	0.49	0.54	0.45	0.48	0.43	0.56	0.68	**0.72**	0.56
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M3	$accuracy$	0.66	0.67	0.66	0.67	0.73	0.69	0.73	0.64	0.66	**0.82**	0.69
	p_{ma}	0.64	0.65	0.67	0.65	0.74	0.72	0.75	0.69	0.66	0.74	
	r_{ma}	0.87	0.87	0.84	0.87	0.90	0.88	0.85	0.81	0.87	0.92	
	$F1_{ma}$	0.67	0.68	0.65	0.67	0.77	0.74	0.75	0.66	0.67	**0.79**	0.71
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M4	$accuracy$	0.78	0.78	0.77	0.80	0.81	0.80	0.80	0.73	0.80	**0.82**	0.79
	p_{ma}	0.65	0.64	0.65	0.75	0.68	0.64	0.63	0.66	0.71	0.73	
	r_{ma}	0.78	0.78	0.80	0.86	0.80	0.65	0.60	0.83	0.87	0.83	
	$F1_{ma}$	0.69	0.69	0.70	**0.79**	0.72	0.62	0.59	0.71	0.77	0.76	0.71
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M5	$accuracy$	0.69	**0.85**	0.79	0.65	0.64	0.79	0.80	0.83	0.77	0.76	0.76
	p_{ma}	0.67	0.77	0.70	0.70	0.69	0.73	0.81	0.72	0.59	0.73	
	r_{ma}	0.84	0.76	0.84	0.78	0.80	0.83	0.70	0.84	0.73	0.81	
	$F1_{ma}$	0.70	0.75	0.75	0.68	0.69	0.76	0.71	**0.76**	0.64	0.74	0.72
		H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	avg_{F1}
M6	$accuracy$	0.82	0.84	0.82	0.73	0.79	0.84	**0.85**	0.85	0.77	**0.85**	**0.82**
	p_{ma}	0.75	0.76	0.76	0.67	0.75	0.76	0.78	0.75	0.73	0.83	
	r_{ma}	0.83	0.85	0.85	0.81	0.89	0.90	0.85	0.85	0.81	0.90	
	$F1_{ma}$	0.77	0.79	0.79	0.70	0.79	0.81	0.80	0.79	0.74	**0.85**	0.78
	$avg_{accuracy}$	0.66	0.68	0.66	0.62	0.66	0.67	0.69	0.66	0.64	**0.71**	0.67
	avg_{F1ma}	0.62	0.62	0.59	0.59	0.61	0.61	0.59	0.61	0.61	**0.67**	0.61

Before aggregation of the labels, one could observe that models with a similar architecture, or models that were trained on the same NLI data sets, perform similar on the same hypothesis. After aggregation this cannot be observed that clearly anymore. Nevertheless, the RoBERTa-based models M_2, M_3 and the DeBERTa-based model M_4, which were all three trained on the XNLI data set, performed best in terms of *accuracy* with H_9. The model having the highest average F1-scores across all ten hypotheses was M_1, which is RoBERTa-based and performed best on the English default hypothesis H_0 in comparison to the other hypotheses. M_1 was already the best model in the evaluation of the fine-grained labels, but having the highest F1-score with a different hypothesis than with the aggregated labels.

An interesting observation is the deviation between F1-scores of the two hypotheses H_5 and H_6, which only differed by the presence of a dot as last character symbol. After aggregation of the labels, the two hypotheses still have different metric results, but the difference between the average F1-scores of H_5 and H_6 over all models is reduced by half. Moreover, the hypothesis with the dot as last sentence character, namely H_5, shows slightly higher results in the aggregated F1-scores than H_6, which was the other way around with fine-grained results.

On average, the *accuracy* achieved with the German hypotheses were slightly higher compared to the English hypotheses (see Table 13). On the other hand, the macro averaged F1-score does not indicate a performance difference between the languages. This might be due to the fact that the aggregated labels focus solely on the primary, well distinguishable classes whereas the fine-grained labels reported above (see Table 10 and Table 11) distinguish between various types of label *other*.

Table 13: Average F1-scores of classification results (with aggregated labels) per language.

	Language	
	ger.	engl.
$avg_{accuracy}$	0.67	0.66
$avgF1_{macro}$	0.61	0.61

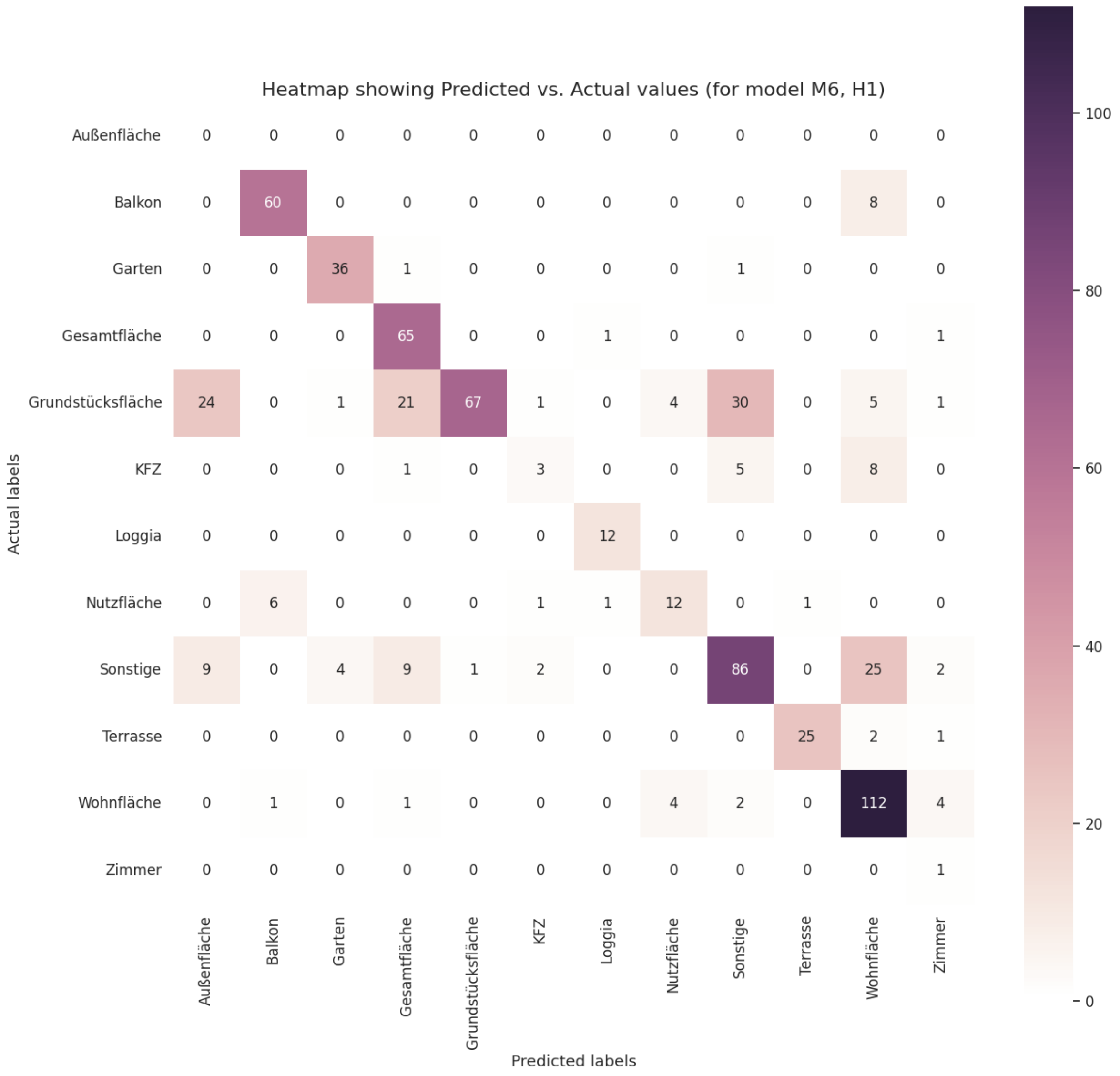
Summing up, the two best performing combinations of model and hypothesis template, in terms of macro averaged F1-score, and using the *aggregated* labels, were:

- M_1, *roberta-large-mnli*, in combination with hypothesis template H_0, *"This example is {label}."*, with a $F1_{macro} = 0.87$
- M_6, *facebook/bart-large-mnli*, in combination with hypothesis template H_9, *"Die Fläche gehört zu {label}."*, with a $F1_{macro} = 0.85$

6.3.3 Analysis of Classification Errors

This subsection contains a further analysis and description of classification errors. To evaluate and compare the effectiveness of two different model and hypotheses combinations for the zero-shot classification, confusion matrices are reported below for the two model and hypotheses combinations that performed best with the two sets of labels, respectively: M_6 with H_1 using the *fine-grained* labels, and M_1 with H_0 using the *aggregated* labels (see Figures 18-20).

The first of these combinations (M_6 with H_1), which is the one that returned the highest macro averaged F1-score for fine-grained labels, is presented in Figure 18 (fine-grained) and in Figure 19a (aggregated).



	Außenfläche	Balkon	Garten	Gesamtfläche	Grundstücksfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Außenfläche	0	0	0	0	0	0	0	0	0	0	0	0
Balkon	0	60	0	0	0	0	0	0	0	0	8	0
Garten	0	0	36	1	0	0	0	0	1	0	0	0
Gesamtfläche	0	0	0	65	0	0	1	0	0	0	0	1
Grundstücksfläche	24	0	1	21	67	1	0	4	30	0	5	1
KFZ	0	0	0	1	0	3	0	0	5	0	8	0
Loggia	0	0	0	0	0	0	12	0	0	0	0	0
Nutzfläche	0	6	0	0	0	1	1	12	0	1	0	0
Sonstige	9	0	4	9	1	2	0	0	86	0	25	2
Terrasse	0	0	0	0	0	0	0	0	0	25	2	1
Wohnfläche	0	1	0	1	0	0	0	4	2	0	112	4
Zimmer	0	0	0	0	0	0	0	0	0	0	0	1

Figure 18: Heatmap fine-grained results M_6 with H_1.

Looking at the first confusion matrix, one can see that there are several classes misleading. The German label *Grundstücksfläche* (engl. plot area) was confused with the semantically related labels *Außenfläche* (engl. area outside), *Gesamtfläche* (engl. total area), but also with the label *Sonstige* (engl. other) in more than 20 cases each. Since especially the cases of the classes *plot area* and *other* are attributed to the aggregated label *other* it is obvious that the aggregated results report an improved rate of correct classifications. Another class with several misclassifications is the fine-grained class label *KFZ* (engl. motor vehicle) which was confused with *other* and *living area* more often than classified correctly. The third class where the model had problems to distinguish the labels was the unspecific fine-grained class *Sonstige* (engl. other), which was confused with *living area* in more than 20 sequences[45], whereas the confusion with other classes like *area outside* was the case in less than 10 times[46] and lack sufficient context information.

Like already mentioned above, the fine-grained labels that were not among the class results of *living area* (ger. Wohnfläche), *usable area* (ger. Nutzfläche), *loggia* (ger. Loggia), *total area* (ger. Gesamtfläche) were aggregated to the class *other*. Therefore, one can observe a large number of classifications with label *other* in Figure 19a, which contains a confusion matrix with the aggregated labels.

During the manual data labelling process, details about problematic sequences were noted and reported above (see Table 5 in Section 4.8). The 189 instances where the zero-shot approach with M_6, H_1 failed to classify the sequences in the data set correctly were investigated in terms of those features of problematic sequences. The potentially problematic features about sequences that were already identified beforehand have partially been a cause for wrong classification results (see Table 14).

The analysis regarding fine-grained labels shows that insufficient context information accounted for 12% of the misclassified cases, while 10% were due to gibberish text, when applying M_6, H_1 (see Table 14). OCR problems were found to be the reason for misclassification in only 4% of cases, and multi-label cases accounted for a quarter of misclassifications. When having a look at the misclassifications with aggregated results, one can see that the multi-label cases accounted for 35% of misclassifications.

[45]The cases of class *other* that were confused with *living area* were for instance sequences like: *"besteht - als Zubehör zum Wohnungseigentum - das Kellerabteil 9 im Ausmaß von ca. 6.91 m²"*, *"Weiters ist der Wohnung der Kellerraum Top C09 im Ausmaß von ca. 5.46 m² im WE-Zubehör zugeordnet"*. One can observe that the German term for cellar occurred quite often in these cases (15 out of 25 times), which points into the direction of using *cellar* (ger. Keller) as a separate label.

[46]The cases of class *other* that were confused with *area outside* were for instance sequences like: *"sohin bei einem Flächenausmaß von 350 m² "*, *"den Teil 1 im Ausmaß von 4 m²"*. One can observe that these cases contain words that are lexicographically similar to the German word *Außen* (engl. outside).

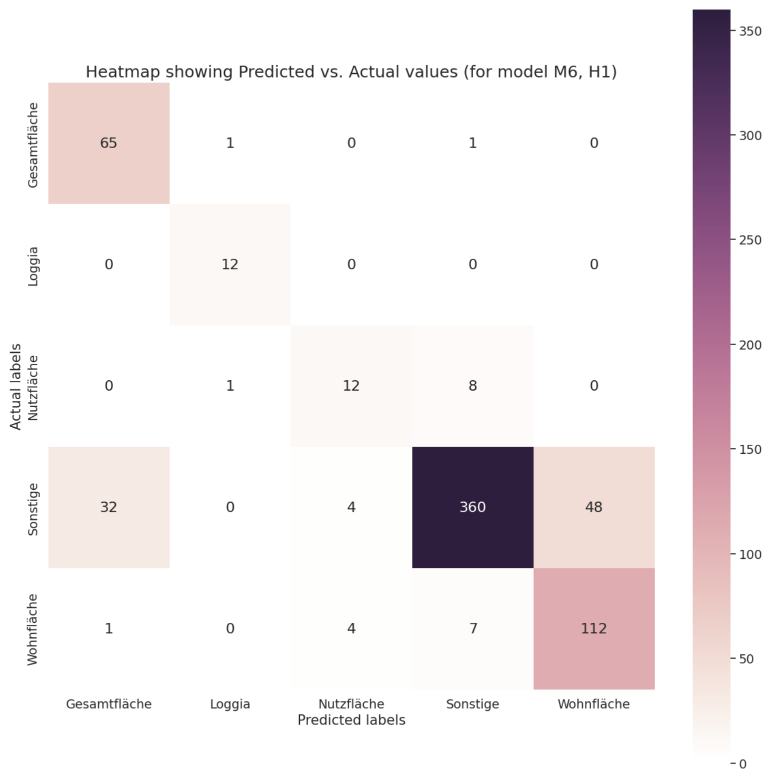
Table 14: Counts of sequence characteristics for wrongly classified cases with M_6, H_1. Sequences were flagged during manual classification to be problematic.

misclassifications with fine-grained labels ($n = 189$)		
	yes (%)	**no (%)**
too little context	22 (12)	167 (88)
peculiar / gibberish	19 (10)	170 (90)
OCR problem	8 (4)	181 (96)
multi-label case	48 (25)	141 (75)
misclassifications with aggregated labels ($n = 107$)		
	yes (%)	**no (%)**
too little context	9 (8)	98 (92)
peculiar / gibberish	6 (6)	101 (94)
OCR problem	5 (5)	102 (95)
multi-label case	37 (35)	70 (65)

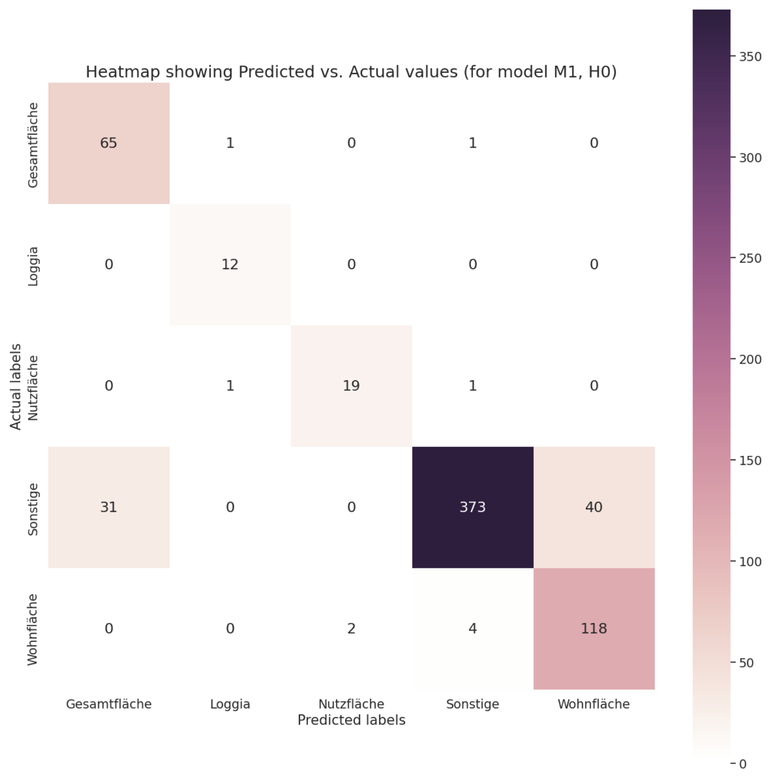
Interestingly, it is observed that some of the sequences that were initially misclassified when only one label was used, are classified correctly with M_6, H_1 when the second label was taken into account. In particular, 17% (32 out of 189) of the fine-grained labels and 25% (27 out of 107) of the aggregated labels that were initially misclassified were classified correctly with the second label[47].

The zero-shot sequence classification model that achieved the highest macro averaged F1-score for the *aggregated* labels was M_1 with hypothesis H_0. For this model and hypothesis combination, Figures 20 and 19b display the confusion matrices. Looking at the heatmap of the fine-grained results in Figure 20, one can see that there are several classes misleading. Especially the classes *Grundstücksfläche* (engl. plot area) and *Sonstige* (engl. other) show several misclassifications. The German label *Grundstücksfläche* (engl. plot area) was confused with the semantically related labels *Außenfläche* (engl. area outside) in more cases than with the above reported model and hypothesis combination M_6, H_1. Besides the confusion with *area outside*, the label *plot area* was confused with the labels *Gesamtfläche* (engl. total area) and *KFZ* (engl. motor vehicle) in more than ten cases each.

[47]An example sequence for such a case where the second fine-grained label is correct is: *"Kinderzimmer im Gesamtausmaß von ca. 59.81 m²"*, where the first label was defined to be class *total area* and the second one class *room*. It has to be noted that this sequence does not have enough context information, but due to the large area of the room, it can be assumed (but is not known for sure) that the area refers to *total area* of an apartment in the contract. Therefore, this sequence is a good example to point out that the amount of context information needed depends a lot on the contract wording and sentence structure.
An example for a misclassified sequence where both the first and second label do not correspond to the model result is: *"laut Teilungsausweis im Flächenausmaß von 37 m²"*, where the first label was defined to be class *other* and the second one class *plot area*. For the second label domain knowledge about contracts is needed to understand that it refers to a plot of land.

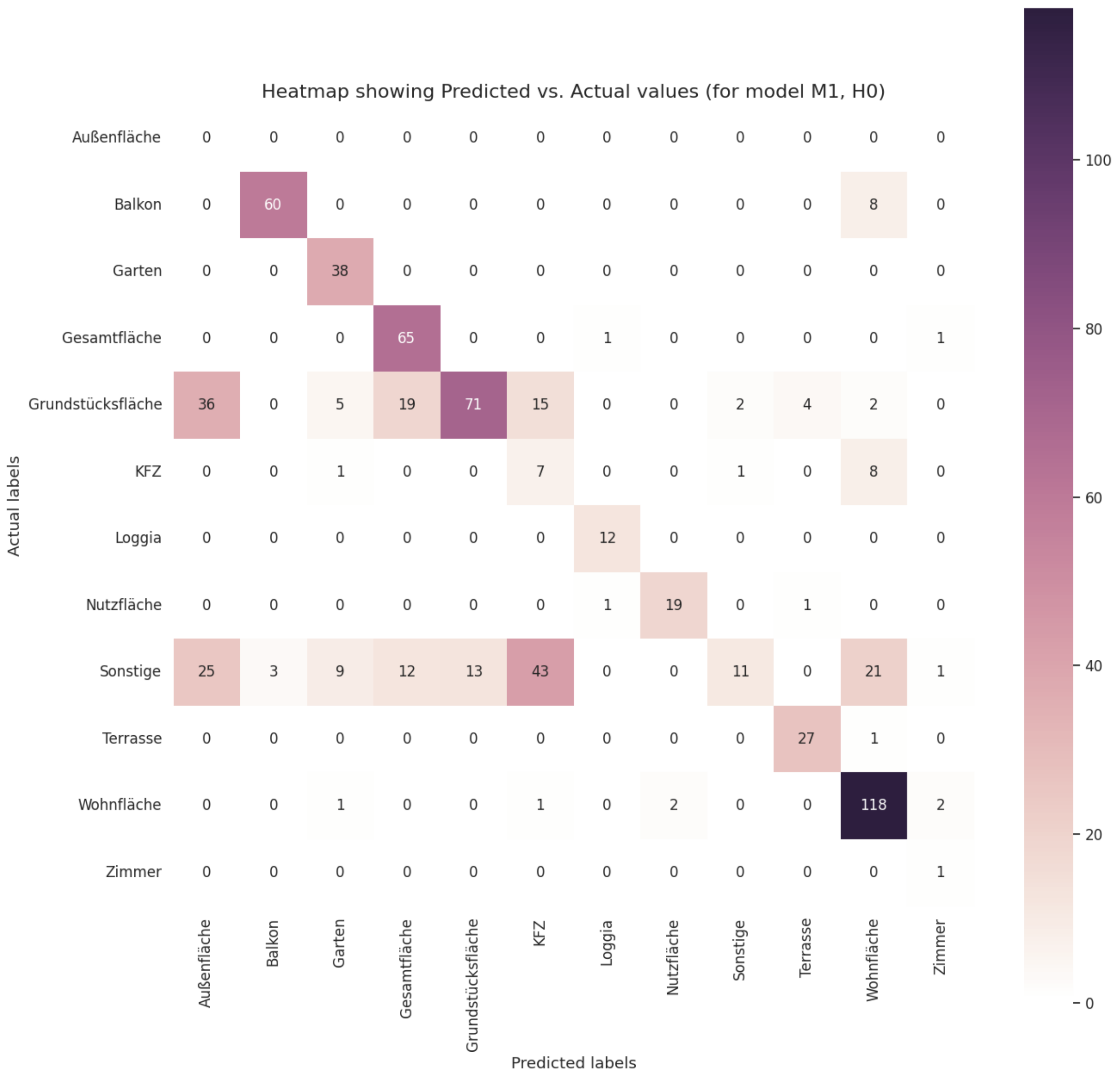


(a) Heatmap aggregated results M_6 with H_1.



(b) Heatmap aggregated results M_1 with H_0.

Figure 19: Heatmaps showing the predicted and true classes for the aggregated labels.



	Außenfläche	Balkon	Garten	Gesamtfläche	Grundstücksfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Außenfläche	0	0	0	0	0	0	0	0	0	0	0	0
Balkon	0	60	0	0	0	0	0	0	0	0	8	0
Garten	0	0	38	0	0	0	0	0	0	0	0	0
Gesamtfläche	0	0	0	65	0	0	1	0	0	0	0	1
Grundstücksfläche	36	0	5	19	71	15	0	0	2	4	2	0
KFZ	0	0	1	0	0	7	0	0	1	0	8	0
Loggia	0	0	0	0	0	0	12	0	0	0	0	0
Nutzfläche	0	0	0	0	0	0	1	19	0	1	0	0
Sonstige	25	3	9	12	13	43	0	0	11	0	21	1
Terrasse	0	0	0	0	0	0	0	0	0	27	1	0
Wohnfläche	0	0	1	0	0	1	0	2	0	0	118	2
Zimmer	0	0	0	0	0	0	0	0	0	0	0	1

Figure 20: Heatmap fine-grained results M_1 with H_0.

In Figure 20, it becomes clear that the zero-shot model does not classify as many cases with class *other* as a human would do manually, compared with the manual ground truth. Apparently model M_1 predicts the class *KFZ* (engl. motor vehicle) more often than the class *Sonstige* (engl. other). Among the sequences that should have been predicted as belonging to class *other*, more than 40 cases were classified to be *motor vehicle*[48] , more than 20 *Außenfläche* (engl. area outside) and *Wohnfläche* (engl. living area) each, and more than 10 cases *Gesamtfläche* (engl. total area) and *Grundstücksfläche* (engl. plot area) each, whereas only 11 cases were actually predicted as *other.* Since especially the cases of the classes *area outside, plot area*, and *motor vehicle* are attributed to the aggregated label *other* it is obvious that the aggregated results report an improved rate of correct classifications compared to the fine-grained result.

Even though the model seems to be sensitive for class *motor vehicle*, among the cases with the true label *motor vehicle* almost as many cases were confused with class *living area*[49]. In general, it can be observed that the class *living area* has false positives from other classes, more precisely eight or more each from the classes *Balkon* (engl. balcony), *motor vehicle* and *other.*

The fine-grained labels that were not among the class results of *living area* (ger. Wohnfläche), *usable area* (ger. Nutzfläche), *loggia* (ger. Loggia), *total area* (ger. Gesamtfläche) were aggregated to the class *other.* Therefore, one can observe a large number of classifications with label *other* in Figure 19b. Comparison of the two heatmaps with aggregated labels, namely M_6, H_1 in Figure 19a and M_1, H_0 in Figure 19b, points out that the latter model and hypothesis combination performs slightly better in the distinction of the classes when they are aggregated. The number of correct classifications on the diagonal is higher for the classes *usable area*, *other*, and *living area* with M_1, H_0 in comparison to the other model and hypothesis combination. For both models it is the case that the zero-shot model does not classify as many cases with class *other* as a human would do manually (and did manually during ground truth generation).

During the manual labelling process, details about problematic sequences were noted and reported above in Section 4.8 (see Table 5). The 239 instances where the zero-shot approach with model M_1 and hypothesis H_0 failed to classify the sequences of the investigated contracts

[48]The cases of class *other* that were confused with *motor vehicle* were for instance sequences like: *"Das Kellerabteil Nr. 20 im Gesamtausmaß von ca. 2.01 m² ist Zubehör"*, which contains the German word for cellar. Other sequences contain gibberish or lack sufficient context information like for instance: *"Aufzug 1312 Fahrräder/Kiwa AR 94.88 m²"*

[49]The cases of class *KFZ* that were confused with *living area* were for instance sequences like: *"Ausmaß mit Wohnungseigentum ca. 13.5 m² verbunden am Autoabstellplatz sind"*. It can observe that the German term for apartment (ger. Wohnung) occurred in all of these cases, which points into the direction that the model is sensitive for the term *Wohnung.*

correctly were investigated in terms of those features of problematic sequences. The potentially problematic features about sequences that were already identified beforehand have partially been a cause for wrong classification results (see Table 15).

Table 15: Counts of sequence characteristics for wrongly classified cases with M_1, H_0. Sequences were flagged during manual classification to be problematic.

misclassifications with fine-grained labels ($n = 239$)		
	yes (%)	**no (%)**
too little context	35 (15)	204 (85)
peculiar / gibberish	41 (17)	198 (83)
OCR problem	10 (4)	229 (96)
multi-label case	43 (18)	196 (82)
misclassifications with aggregated labels ($n = 81$)		
	yes (%)	**no (%)**
too little context	7 (9)	74 (91)
peculiar / gibberish	13 (16)	68 (84)
OCR problem	2 (2)	79 (98)
multi-label case	26 (32)	55 (68)

The analysis regarding fine-grained labels shows that insufficient context information accounted for 15% of the misclassified cases, while 17% were due to gibberish text for the results with M_1, H_0. OCR problems were found to be the reason for misclassification in only 4% of cases, and multi-label cases accounted for 18% of misclassifications. When having a look at the misclassifications with aggregated results, one can see that the multi-label cases accounted for 32% of misclassifications.

Interestingly, it is observed that also with M_1, H_0 some of the sequences that were initially misclassified when only one label was used, are classified correctly when the second label was taken into account. In particular, 10% (24 out of 239) of the fine-grained labels and 27% (22 out of 81) of the aggregated labels that were initially misclassified were classified correctly with the second label[50].

[50] An example sequence for such a case where the second fine-grained label is correct is: *"Kinderzimmer im Gesamtausmaß von ca. 59.81 m²"*, where the first label was defined to be class *total area* and the second one class *room*. This sequence was already reported at this point above, where the results of the other model and hypothesis combination (M_6, H_1) were reported. Both models classify this sequence, which has little context information, as belonging to class *room*.
An example for a misclassified sequence where both the first and second label do not correspond to the model result is: *"1230 m² Wald vorgetragen ist"*, where the first label was defined to be class class *plot area* and the second one *other*. The model M_1 classified this sequence as *living area*, which is far from the actual meaning of the sequence.

6.4 Evaluation of the Rule-based Approach

In order to evaluate and compare the performance of zero-shot models with a rule-based classifier, the rule-based classification (detailed in Section 5.5) was performed on the exact same text sequences as the zero-shot classification. In this section, the performance of the rule-based classification method on the sequences is reported. First, the rule-based results are reported in regard of the fine-grained labels and second they are reported in regard of the aggregated labels (see Table 16 and Figures 21-22).

Table 16: Rule-based classification. The metrics describing the classification results with fine-grained (left) and aggregated (right) labels are *accuracy* and macro averaged ($_{ma}$) precision (p), recall (r), and F1 measure.

	fine-grained	**aggregated**
accuracy	0.63	0.89
p_{ma}	0.78	0.89
r_{ma}	0.71	0.80
$F1_{ma}$	0.65	0.83

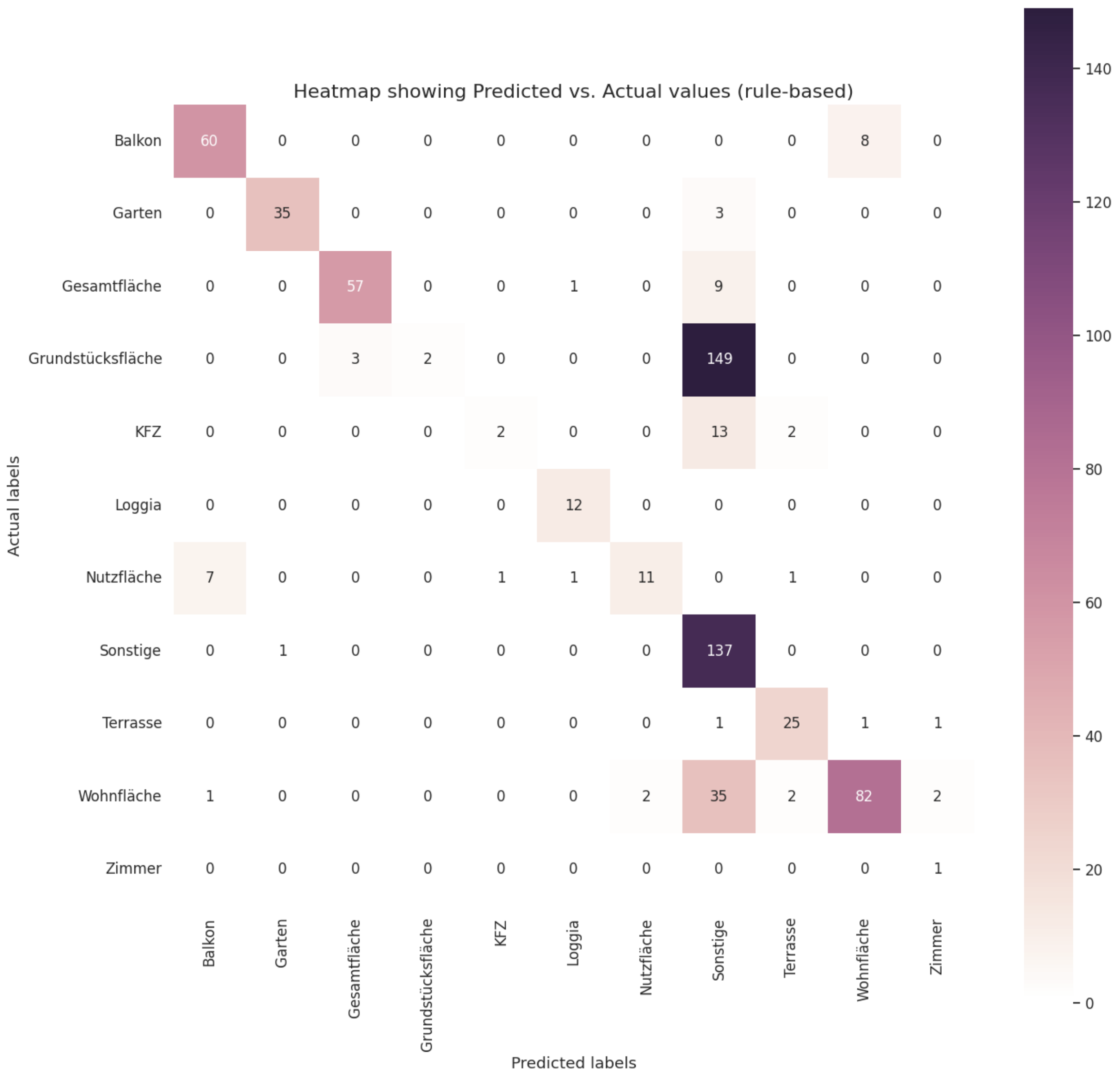
With the rule-based approach, an *accuracy* of 0.63 was obtained for the fine-grained labels (see Table 16, left column). The performance falls between the accuracy results range of the zero-shot models reported in Section 6.3.1 (see Table 10). There were zero-shot models which performed better in regard of the *accuracy* but not in regard of the $F1_{macro}$, where 0.63 was the best result obtained, whereas the rule-based classification approach resulted in a slightly higher score of $F1_{macro} = 0.65$.

In regard of the F1-scores with aggregated labels, the rule-based approach results in *accuracy* 0.01 above the best zero-shot LLM, and a $F1_{macro}$ lying between the range of macro F1 results of the zero-shot models reported in Section 6.3.1 (see Table 12).

The correct and incorrect classifications of the rule-based method can be seen in the heatmap in Figure 21, which presents the confusion matrix of the rule-based classification for fine-grained labels. Looking at the figure there is one class missing, that has been in the fine-grained heatmaps of the model classifications, which is *Außenfläche* (engl. area outside). The reason for that is that for the rule-based approach, no cases were classified with this label and no sequence in the manually labelled data set had this label assigned.[51]

Besides that, it is noticeable that class *Sonstige* (engl. other) contains a lot of cases that should have been labeled to be in different classes. Two cases from class *Grundstücksfläche*

[51] Just one sequence had this label assigned as potential second label for multi-class label cases. Counts of labels were reported in Section 4.8.



	Balkon	Garten	Gesamtfläche	Grundstücksfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Balkon	60	0	0	0	0	0	0	0	0	8	0
Garten	0	35	0	0	0	0	0	3	0	0	0
Gesamtfläche	0	0	57	0	0	1	0	9	0	0	0
Grundstücksfläche	0	0	3	2	0	0	0	149	0	0	0
KFZ	0	0	0	0	2	0	0	13	2	0	0
Loggia	0	0	0	0	0	12	0	0	0	0	0
Nutzfläche	7	0	0	0	1	1	11	0	1	0	0
Sonstige	0	1	0	0	0	0	0	137	0	0	0
Terrasse	0	0	0	0	0	0	0	1	25	1	1
Wohnfläche	1	0	0	0	0	0	2	35	2	82	2
Zimmer	0	0	0	0	0	0	0	0	0	0	1

Figure 21: Heatmap of fine-grained results with *rule-based* approach.

(engl. plot area) were classified correctly, whereas more than hundred cases were confused with class *other*[52].

Another class with many confusions with class *other* is *KFZ* (engl. motor vehicle)[53]. The reason for that is most probably the different ways how the reference to a parking spot can be written. The same is the case for the class *plot area*, where various synonyms and abbreviations can be used. Regarding class *motor vehicle*, it can be observed that two sequences were classified as *terrace*. The reason for that is noisy sequence where the sequences split failed and two areas with context were present in the sequences.

Moreover, almost a third of the cases belonging to class *living area* were confused with class *other*. Sentences and sub-sentences containing the German word for apartment, but not followed by the word for area, are among those misclassifications.

There were minor confusions with less than ten cases each, for example cases of type *Nutzfläche* (engl. usable area) that were classified to belong to class *Balkon* (engl. balcony). The reason for that is that sequences contained the exact German words for *balcony* and *usable area*, and one label is assigned before the other is even checked in the classification algorithm. This is a usual behaviour in an if-else-like decision using an ordered class rule list and can be discussed.

Besides the fine-grained results, especially the aggregated results are of interest for the purpose of approach comparison. The metrics of the rule-based approach with aggregated labels are shown in Table 16 (right column). The logic of labels aggregation was described in Section 3.3.2. In comparison to the fine-grained rule-based results, the aggregated ones had a larger *accuracy* than $F1_{macro}$, which was the other way around for the fine-grained labels. The rule-based approach's $F1_{macro}$ lies with 0.83 below the maximum $F1_{macro}$ of the best zero-shot model's result of 0.87. On the other hand, the $accuracy = 0.89$ obtained with the rule-based approach was above the highest $accuracy = 0.88$ obtained with the best zero-shot classification model. The reason for the high *accuracy* in the rule-based approach could be that there is a high number of correct classifications of label *other*, which can be seen in Figure 22 with a dark purple color. This Figure presents the heatmap with the confusion matrix of the rule-based classification for aggregated labels.

[52]The cases of class *plot area* that were confused with *other* were for instance sequences like: *"zwar in die Grundstücke 1569/1 mit einer Fläche von nunmehr restlich 470 m²"*, which contains the German word for plot and area, but not in the combination where an exact pattern match would result in a positive match.

[53]The cases of class *motor vehicle* that were confused with *other* were for instance sequences like: *"Ausmaß mit Wohnungseigentum ca. 13.5 m² verbunden am Autoabstellplatz"*, which contain a written out word to refer to a car and the parking spot and not the exact pattern "KFZ".

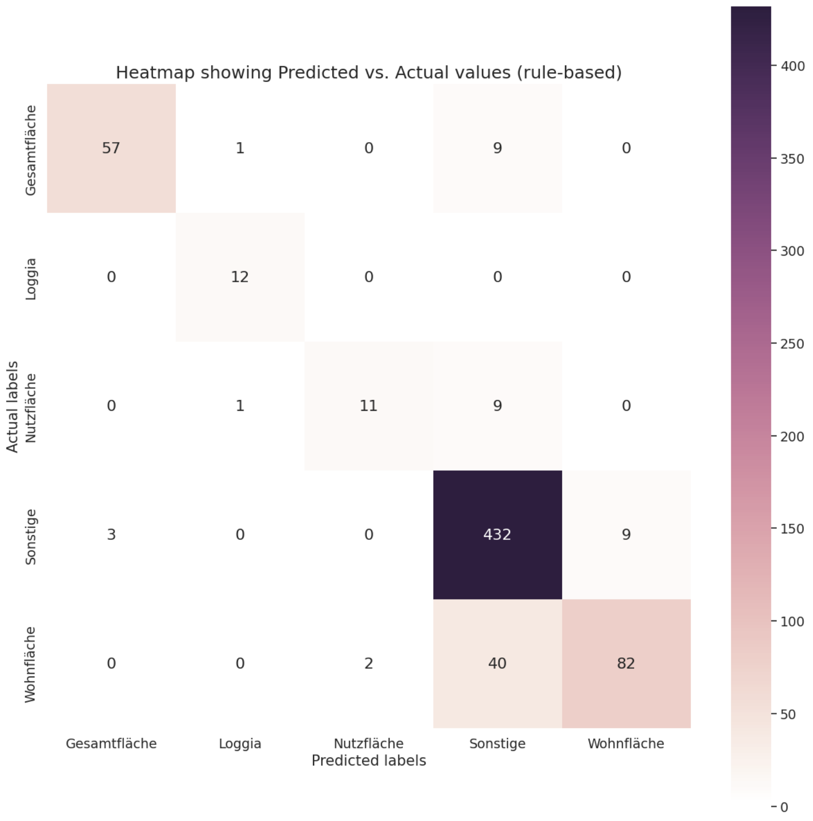


Figure 22: Heatmap of aggregated results with *rule-based* approach.

While the rule-based approach classifies 432 cases correctly as belonging to class *other*, the zero-shot models reported in Section 6.3 classified just 360 (M_6 with H_1) and 373 (M_1 with H_0) respectively, when aggregating the labels. Further comments on the reason for this will be given in the Evaluation and Discussion section (see Section 7).

The 244 instances, where the rule-based approach failed to classify the sequences correctly, were investigated in terms of problematic sequence features. The potentially problematic features about sequences that were already identified beforehand have partially been a cause for wrong classification results (see Table 17).

Table 17: Counts of sequence characteristics for wrongly classified cases with the rule-based method. Sequences were flagged during manual classification to be problematic.

misclassifications with fine-grained labels ($n = 244$)		
	yes (%)	**no (%)**
too little context	23 (9)	221 (91)
peculiar / gibberish	5 (2)	239 (98)
OCR problem	14 (6)	230 (94)
multi-label case	56 (23)	188 (77)
misclassifications with aggregated labels ($n = 74$)		
	yes (%)	**no (%)**
too little context	13 (18)	61 (82)
peculiar / gibberish	2 (3)	72 (97)
OCR problem	8 (11)	66 (89)
multi-label case	31 (42)	43 (58)

The analysis regarding fine-grained labels shows that insufficient context information accounted for 9% of the misclassified cases, while just 2% were due to gibberish text. OCR problems were found to be the reason for misclassification in 6% of cases, and multi-label cases accounted for 22% of misclassifications. When having a look at the misclassifications with aggregated results, one can see that the multi-label cases accounted for 42% of misclassifications.

Interestingly, it is observed that also with the rule-based method some of the sequences that were initially misclassified when only one label was used, are classified correctly when the second label was taken into account. In particular, 9% (23 out of 244) of the fine-grained labels and 30% (22 out of 74) of the aggregated labels that were initially misclassified were classified correctly with the second label.

When comparing the total number and the percentages of misclassifications of the rule-based method from Table 17 with the zero-shot method using M_1, H_0 (see Table 15) it can be observed that the total number of misclassifications as well as the amount of problematic sequences that had an effect on the classification result are rather similar. Both approaches have roughly 240 misclassifications with the fine-grained and roughly 80 misclassifications using the aggregated labels.

Moreover, it can be observed that lacking context information was the reason for more confusions in the rule-based method compared to the zero-shot models (both M_1, H_0 and M_6, H_1), using aggregated labels, and vice versa for the fine-grained labels.

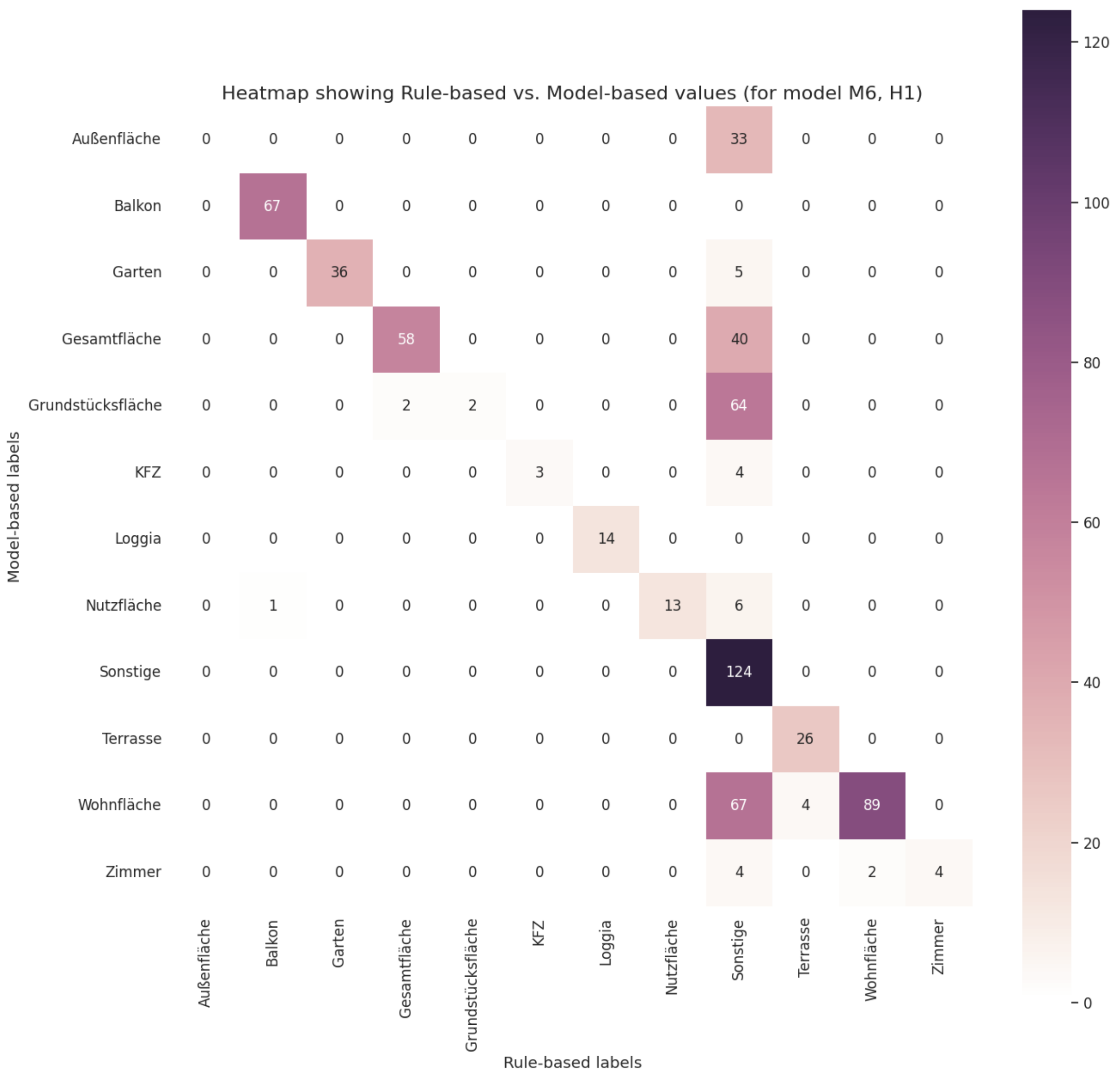
6.5 Evaluation of the Rule-based against Zero-shot Approach

So far, the results of the zero-shot as well as rule-based approach have been reported in comparison to the manual ground truth (see Sections 6.3.1, 6.3.2, 6.4). For the purpose of evaluating whether the zero-shot approach and the rule-based classifier classify the same sequences into the same and/or correct classes, and to gain insights into which sequences are misclassified, confusion matrices comparing both methods are reported. An additional advantage of the heatmaps displaying the confusion matrices is that they make the difference in classifications of the two methods visible. This section first covers not-filtered and coinciding classifications, and focuses on diverging classifications secondly.

Comparing (Coinciding) Classifications

The Figures 23 to 25 contain the confusion matrices for the two model and hypotheses combinations that were assessed to performed best (in comparison to the ground truth). These combinations were M_6 with H_1 using the fine-grained labels, and M_1 with H_0 using the aggregated labels. The zero-shot model M_6 with H_1, which achieved the highest macro averaged F1-score for fine-grained labels, is compared to the rule-based classification in Figure 23 (fine-grained) and in Figure 25a (aggregated).

The strong diagonal in the heatmap (see Figure 23) indicates a high level of consistency between the two classification methods (same class determined by both methods is 436 out of 668 sequences), with some exceptions. Since the rule-based approach contains an else-condition for all the sequences that do not directly match a class pattern (see Section 5.5), it assigns many more sequences to the *other* class than the model-based zero-shot sequence classification. Figure 23 displays the rule-based results on the x-axis, with numerous classifications in the *other* class visible in the right third of the figure. These classifications are due to the else-condition rule, which was already pointed out in Section 6.4. Interestingly, all cases that were



	Außenfläche	Balkon	Garten	Gesamtfläche	Grundstücksfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Außenfläche	0	0	0	0	0	0	0	0	33	0	0	0
Balkon	0	67	0	0	0	0	0	0	0	0	0	0
Garten	0	0	36	0	0	0	0	0	5	0	0	0
Gesamtfläche	0	0	0	58	0	0	0	0	40	0	0	0
Grundstücksfläche	0	0	0	2	2	0	0	0	64	0	0	0
KFZ	0	0	0	0	0	3	0	0	4	0	0	0
Loggia	0	0	0	0	0	0	14	0	0	0	0	0
Nutzfläche	0	1	0	0	0	0	0	13	6	0	0	0
Sonstige	0	0	0	0	0	0	0	0	124	0	0	0
Terrasse	0	0	0	0	0	0	0	0	0	26	0	0
Wohnfläche	0	0	0	0	0	0	0	0	67	4	89	0
Zimmer	0	0	0	0	0	0	0	0	4	0	2	4

Figure 23: Heatmap of fine-grained results comparing rule-based and model-based (M_6, H_1) values.

classified as *other* by M_6 with H_1 were also classified as belonging to this class by the rule-based classifier. However, many cases that were classified as a different class by the zero-shot model were classified as *other* by the rule-based approach.

The largest deviance from the diagonal, where the rule-based approach did not match the same class as the zero-shot approach were the *living area* as well as the *plot area* cases. The zero-shot approach using M_6 with H_1 classified roughly 70 cases more as belonging to *living area* as the rule-based approach. It should be noted that this does not necessarily mean that these classifications are correct, but rather indicates that the concept of *living area* was detected by different wordings than just the name of the class. In fact, the number of cases correctly classified by both approaches (using M_6 with H_1) is 371. However, there were 65 sequences that were classified with the same label by both approaches, but this label was not correct according to the manual ground truth. An illustration of the class distributions where both methods classified the sequences correctly or incorrectly, respectively, is provided in Appendix D.2.

In total, 436 sequences were assigned the same label by both methods (using M_6 with H_1). Moreover, there were 136 sequences where neither of the two methods classified them correctly, with 71 of these resulting in the same (but incorrect) label. Surprisingly, when counting the problematic sequence characteristics, that were noted down for the sequences during manual labelling (see Section 4.8), only 10 out of the 71 misclassified sequences contained too little context to determine the class (manually). Moreover, two cases contained non-meaningful text, four had an OCR problem and 16 an additional class that could be set to be the true class.

Figure 25a displays the aggregated values of the zero-shot model (M_6 with H_1) compared to rule-based classification in a heatmap. The largest remaining blocks of diverging classifications are the *total area* and the *living area*. Among the 40 sequences that were classified as *total area* by the zero-shot model M_6 with H_1 and as other by the rule-based approach, only 6 sequences were actually correct classifications and actually referring to *total area*. An example sentence is *"277/22 im Gesamtausmaß von 843 m²"* (engl: 277/22 with a total extend of 843 m^2). However, this is a rather short sequence were additional context information would probably ease determination of the correct class for LLMs.

An insight in the sequences that were classified as *total area* (zero-shot) and *other* (rule-based) showed that these were mostly short sequences with little context and referring to plot areas but without mentioning the German word for plot directly.[54]

[54] 20 out of 32 cases had the fine-grained label *plot area*.

Figure 24 contains the heatmap of M_1, H_0 compared to the rule-based approach. In contrast to the previous heatmap of the other model (M_6, H_1), the tendency of the model M_1, H_0 to classify the sequences to 'any' class but not to class *other*, becomes visible. The results of M_1 using H_0 were reported previously in Figure 20 and showed only 14 correctly classified cases of fine-grained class *other*, whereas all the other cases got some different class assigned. This is a reason why the models metrics were not that well (compared to other models) in the fine-grained results evaluation as in the aggregated labels evaluation[55]. Due to this behaviour of the model and the opposite behaviour of the rule-based classifier, many deviations in the *other* column in the right third of the Figure 24 are visible. Besides this, the figures containing the fine-grained labels of the two zero-shot models (M_6, H_1 in Figure 23 and M_1, H_0 in Figure 24) compared to the rule-based values, are quite similar.

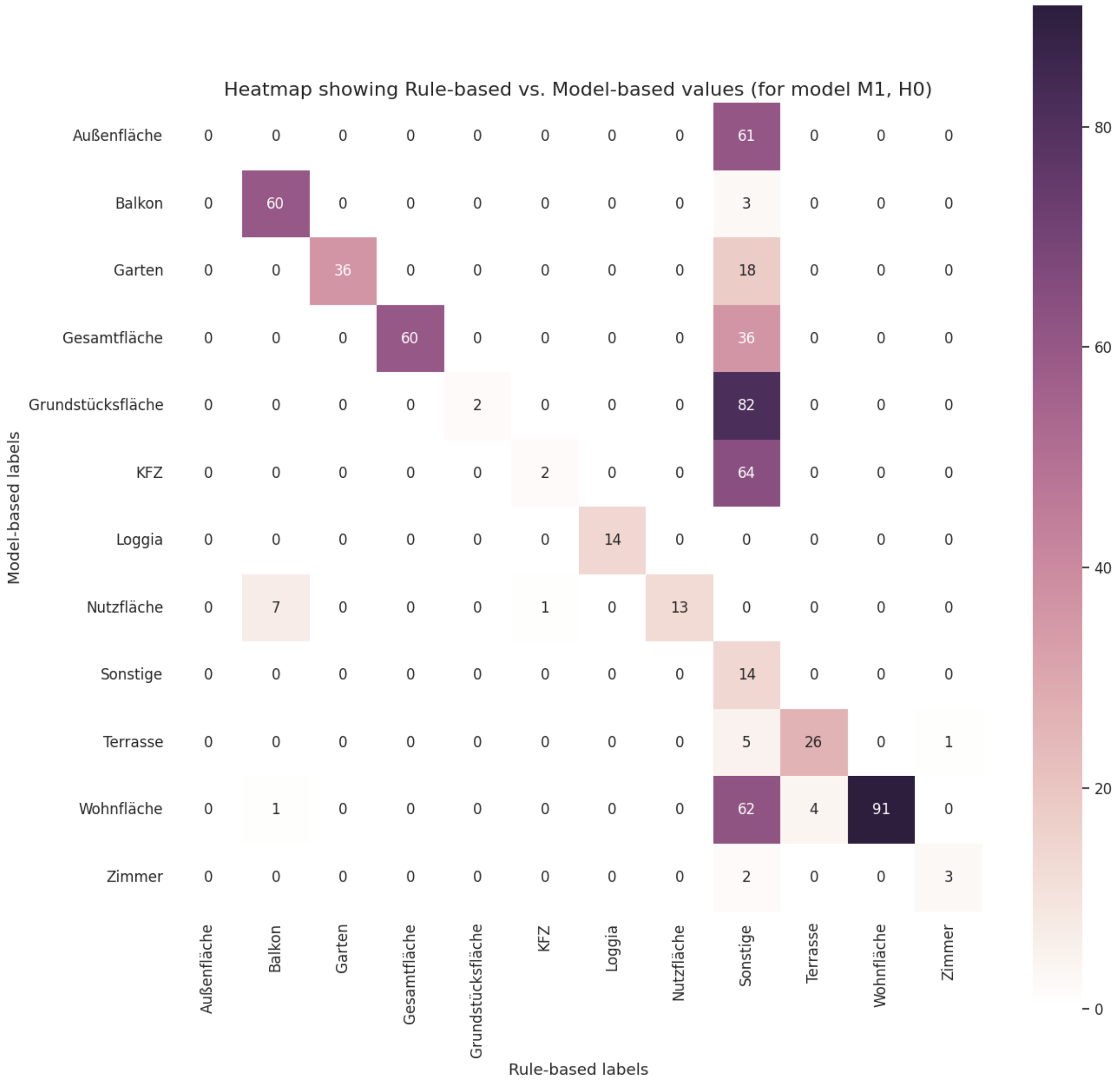
Apart from that, Figure 25, which contains the two heatmaps with the aggregated labels of the two zero-shot models against the rule-based approach, is also quite similar. It does not show much difference in the values of the two zero-shot models on the y-axis, and most of the cells in the heatmaps only diverge by ± 2.

Another interesting observation in Figure 24 is that the strongest concurrence of the zero-shot model and the rule-based classifier is *living area*, which depicts 91 sequences classified by both approaches. Out of the 91 cases where the models agreed on *living area*, 82 (90%) were classified correctly according to the ground truth.

Overall, the number of cases correctly classified by both approaches (using M_1 with H_0) is 298. However, there were 23 sequences that were classified with the same label by both approaches, but this label was not correct according to the manual ground truth. An illustration of the class distributions where both methods classified the sequences correctly or incorrectly, respectively, is provided in Appendix D.2.

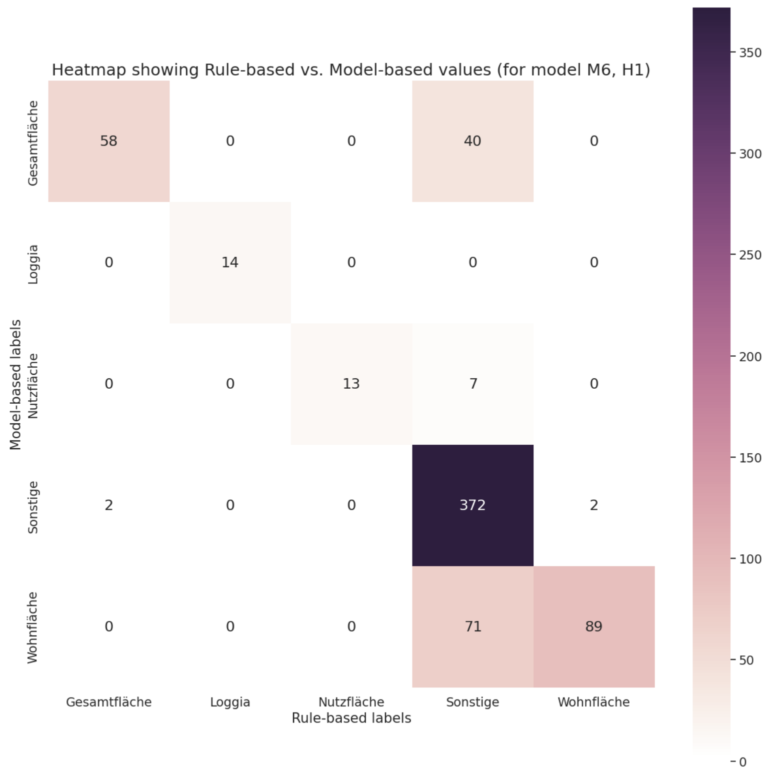
In total, 321 sequences were assigned the same fine-grained label by both methods (using M_1 with H_0). Moreover, there were 113 sequences where neither of the two methods classified them correctly, with 23 of these resulting in the same (but incorrect) label and 90 of these had a different (and incorrect) label. Surprisingly, when counting the problematic sequence characteristics, that were noted down for the sequences during manual labelling (see Section 4.8), only 11 out of the 90 misclassified (and different classified) sequences contained too little context to determine the class (manually). Moreover, two cases contained non-meaningful text, three had an OCR problem and 17 an additional class that could be set to be the true class.

[55]With aggregated labels, M_1 using H_0 even achieved the highest macro averaged F1-score. This is primarily because most of the misclassified sequences were assigned a class that was aggregated to the *other* class.

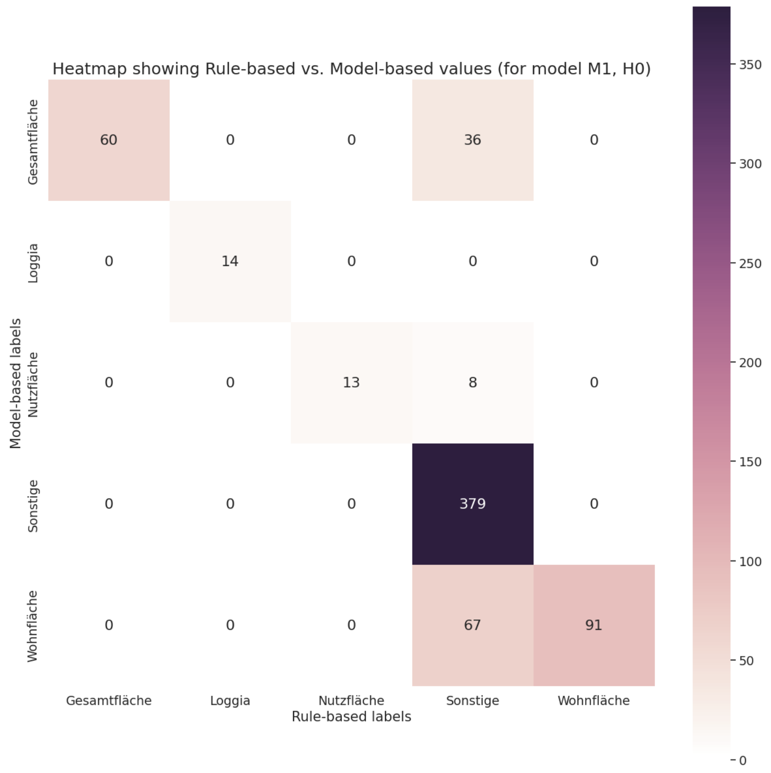


	Außenfläche	Balkon	Garten	Gesamtfläche	Grundstücksfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Außenfläche	0	0	0	0	0	0	0	0	61	0	0	0
Balkon	0	60	0	0	0	0	0	0	3	0	0	0
Garten	0	0	36	0	0	0	0	0	18	0	0	0
Gesamtfläche	0	0	0	60	0	0	0	0	36	0	0	0
Grundstücksfläche	0	0	0	0	2	0	0	0	82	0	0	0
KFZ	0	0	0	0	0	2	0	0	64	0	0	0
Loggia	0	0	0	0	0	0	14	0	0	0	0	0
Nutzfläche	0	7	0	0	0	1	0	13	0	0	0	0
Sonstige	0	0	0	0	0	0	0	0	14	0	0	0
Terrasse	0	0	0	0	0	0	0	0	5	26	0	1
Wohnfläche	0	1	0	0	0	0	0	0	62	4	91	0
Zimmer	0	0	0	0	0	0	0	0	2	0	0	3

Figure 24: Heatmap of fine-grained results comparing rule-based and model-based (M_1, H_0) values.



(a) Heatmap aggregated results of model M_6, H_1 in comparison to the rule-based results.



(b) Heatmap aggregated results of model M_1, H_0 in comparison to the rule-based results.

Figure 25: Heatmaps of aggregated labels describing rule-based and model-based results.

Diverging Classifications

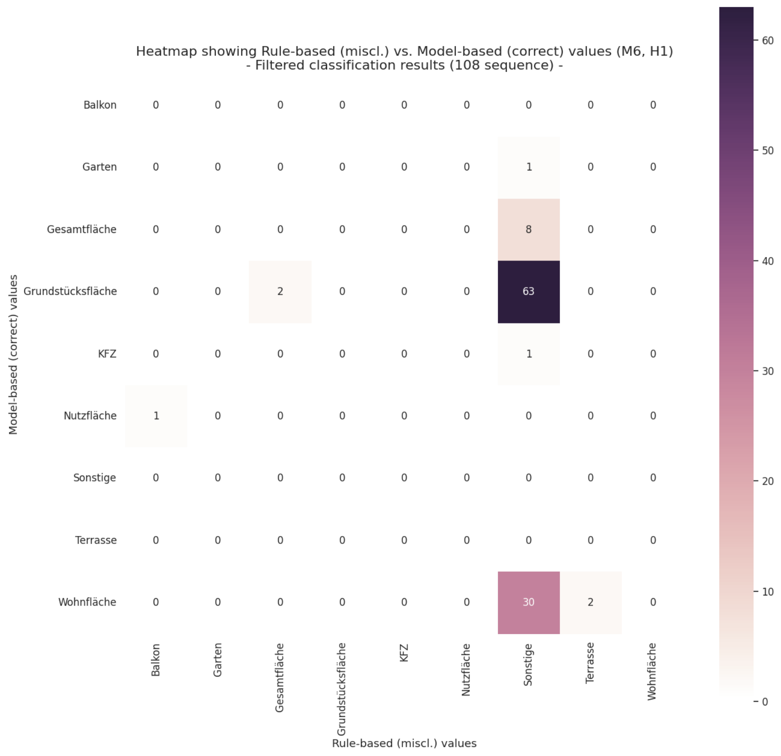
To ease the comparison of which classes exhibited better results with what method, additional analysis were performed. To make it even more clear how many cases of which categories are classified correctly by which method, additional counts, heatmaps and tables are reported below. An interesting number is the number of sequences where one method performed well whereas the other could not determine the correct class.

The model M_6 with hypothesis H_1 performed better compared to the rule-based approach in 108 cases, using fine-grained labels. On the other hand, the rule-based classifier performed better than the model in 53 sequences. These two heatmaps are displayed in Figure 26a and b. In Figure 26a it is visible that the correct classifications of the zero-shot model are mainly attributed to *plot area* and *living area*.

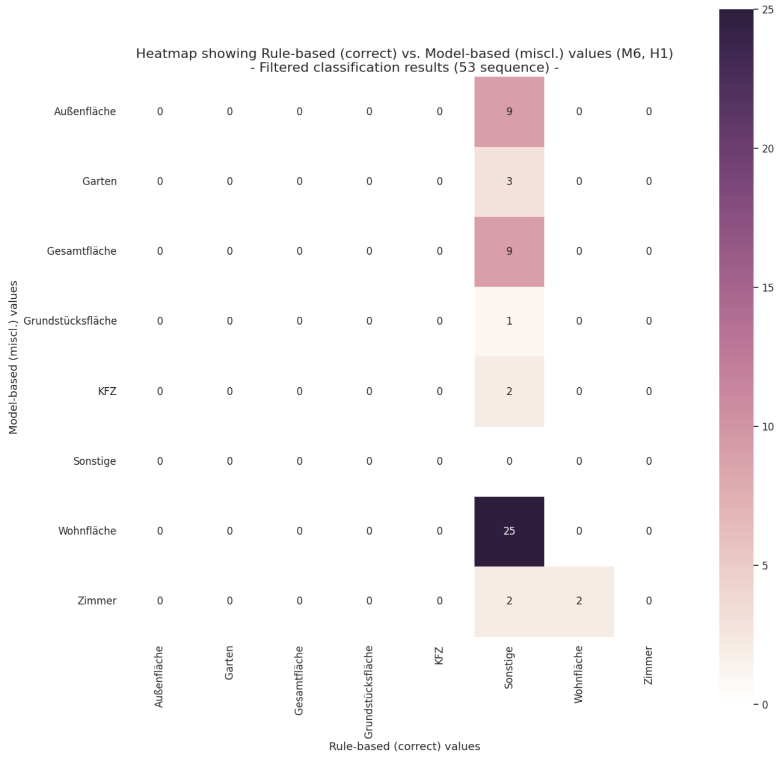
However, in Figure 26b it becomes clear, that the zero-shot model behaviour favouring classification of different classes can become a disadvantage as well. The column *other* is displayed very strong and the distribution of misclassifications, that would actually belong to class *other*, seems like a bunch of randomly assigned classes.

When examining the aggregated label results, the counts of correct classifications, where the other method failed, were more close to each other, with a shift in the order what approach is more correct. The zero-shot model was better in 43 sequences whereas the rule-based classifier outperformed the zero-shot model in 76 cases. These findings are illustrated in Figure 40 (see Appendix D.2).

At this point it should be reminded, that the total number of correct classifications of the two approaches were 479 with the zero-shot model M_1, H_0 and 424 with the rule-based classifier, with fine-grained labels, and 561 with the zero-shot model M_1, H_0 and 594 with the rule-based classifier, with aggregated labels.



(a) Heatmap model M_6, H_1 in comparison to the rule-based results. Filtered on cases with *correct* model and *wrong* rule-based classification.



(b) Heatmap model M_6, H_1 in comparison to the rule-based results. Filtered on cases with *wrong* model and *correct* rule-based classification.

Figure 26: Heatmaps of fine-grained labels, classification results filtered on either model or rule-based correct.

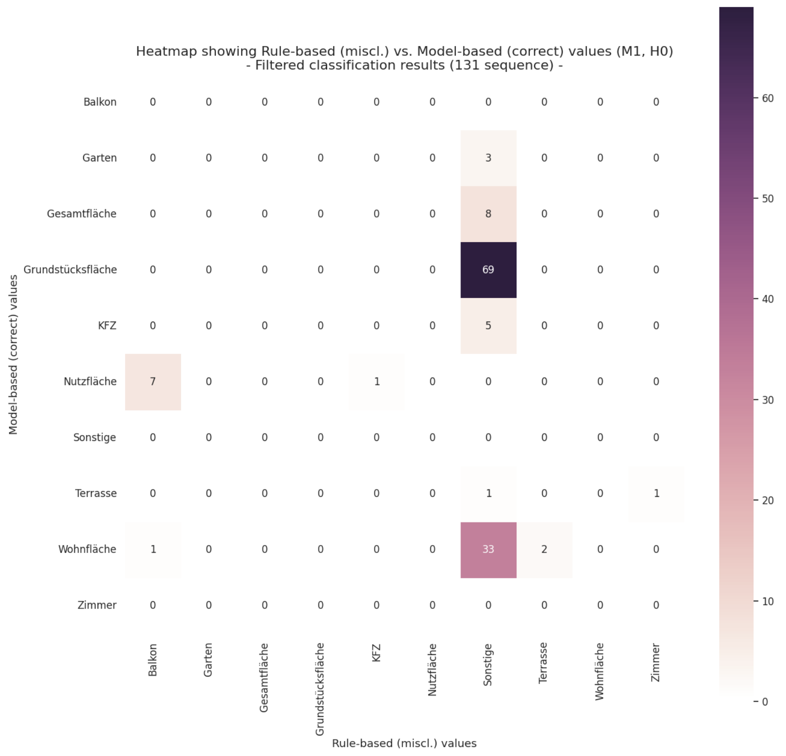
After describing model M_6 using H_1 in comparison to the rule-based approach, model M_1 using H_0 is described as well. The model M_1 with hypothesis H_0 performed better compared to the rule-based approach in 131 cases (using fine-grained labels). On the other hand, the rule-based classifier performed better than the model in 126 sequences. These two heatmaps are displayed in Figure 27a and b. In Figure 27a it is visible that the correct classifications of the zero-shot model are mainly attributed to *plot area* and *living area.* It can be noted as an advantage that the correct classifications are spread across several classes and that the LLM approach favours detection of different classes depending on the sequence. Especially the living area, which is an important class to determine correctly (like pointed out in Section 1), the zero-shot model performs better in detection compared to the rule based method.

However, in Figure 27b it becomes clear, that the zero-shot model behaviour favouring classification of many classes can become a disadvantage as well. The column *other* is displayed very strong and the distribution of misclassifications, that would actually belong to class *other*, seems like a bunch of randomly assigned classes. In cases of uncertainty it might be more suitable to assign class *other* instead of a random but slightly favoured class. Potential solutions to this issue are further discussed in Section 7.

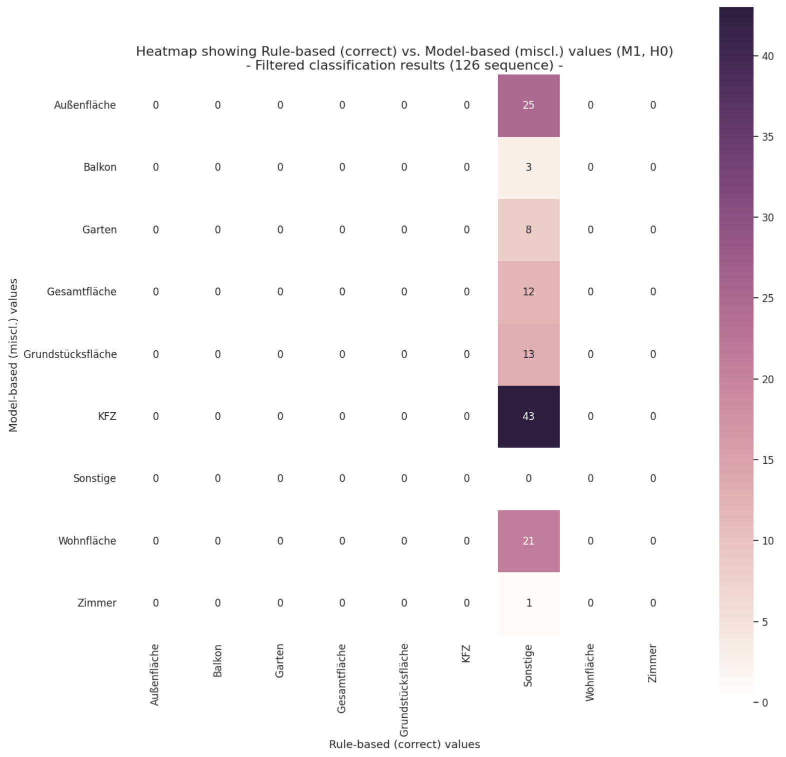
When examining the aggregated label results, the counts of correct classifications, where the other method failed, were still close to each other, with a shift in the order what approach is more correct. The zero-shot model was better in 52 sequences whereas the rule-based classifier outperformed the zero-shot model in 59 cases. These findings are illustrated in Figure 41 (see Appendix D.2).

At this point it should be reminded, that the total number of correct classifications of the two approaches were 429 with the zero-shot model M_1, H_0 and 424 with the rule-based classifier, with fine-grained labels, and 587 with the zero-shot model M_1, H_0 and 594 with the rule-based classifier, with aggregated labels.

Figure 23 above showed a high number of cases where the zero-shot model M_6, H_1 classified sequences as belonging to *living area* whereas the rule-based approach classified them as *other*. When analyzing the specific *living area* sequences that were misclassified by the rule-based approach but correctly classified by the zero-shot model, an obvious assumption is that these cases contain the German word for "apartment" along with a synonym of the German word for "area" or "size". Further examination of these sequences confirms this initial assumption. An example sequence that was correctly classified as belonging to living area (by zero-shot) was *"die Wohnung 45 ca. 31.67 m^2 befinden sich im Dachgeschoss des Gebaudes"* (engl. the apartment 45 approx. 31.67 m^2 is located on the top floor of the building). This sequence contains the German word for "apartment" but not the specific word of the class label that



(a) Heatmap model M_1, H_0 in comparison to the rule-based results. Filtered on cases with *correct* model and *wrong* rule-based classification.



(b) Heatmap model M_1, H_0 in comparison to the rule-based results. Filtered on cases with *wrong* model and *correct* rule-based classification.

Figure 27: Heatmaps of fine-grained labels, classification results filtered on either model or rule-based correct.

served as pattern in the rule-based approach.

However, a case that was correctly classified to belong to *other* (and not *living area*) with the rule-based approach was *"Weiters ist der Wohnung der Kellerraum Top D16a im Ausmaß von ca. 3.17 m^2 im WE-Zubehör zugeordnet"* (engl. Furthermore, the cellar room Top D16a measuring approx. 3.17 m^2 is assigned to the apartment in the condominium ownership accessories), which contains the German word for apartment, but actually refers to the apartments cellar room. This is an example where the zero-shot model (M_6 with H_1) failed to comprehend the context and meaning of the sequence in a way that fulfills the intended task. However, in this case, the model was presented with the hypothesis *"Dieses Beispiel ist class."* and *living area* was apparently the model's best guess among all available classes.

Additional and quite similar examples of sequences containing the German word for apartment (ger. "Wohnung") were also discussed above in Section 6.3.3 when classification errors were initially described. Besides the apartment cases, additional cases with lexicographical similarity in German were previously mentioned. Among them were cases of *area outside* (ger. Außenfläche) that were confused by the zero-shot models with other classes whenever they contained "ausmaß" (engl. extend).

Besides that, an even more abstract classification result are the numerous cases from class *other* that were misclassified by M_1 to belong to class *motor vehicle* (see Section 6.3.3). The *motor vehicle* class was in German called "KFZ" because this abbreviation of the German motor vehicle word was heuristically assumed to be quite frequent in the purchase contract documents. Additionally, it was assumed that the LLMs can transfer their knowledge about terms and words, and find proper synonyms to identify these cases correctly. Apparently, model M_1 with hypothesis H_0 failed this assumption, which is also evident in Figure 27b. The Figure shows a high number of misclassifications where the zero-shot model (somehow) assigned class 'KFZ' to the sequence instead of class "Sonstige" (engl. *other*). Nevertheless, as pointed out earlier, these sequences do not even contain the term 'KFZ' or lexicographical similar terms. A closer investigation showed, that these sequences contained letters of 'KFZ', like e.g. the sequence *"Das* ***K****ellerabteil Nr. 20 im Gesamtausmaß von ca. 2.01 m^2 ist* ***Z****ubehör"* (engl. The cellar compartment no. 20 with a total extend of approx. 2.01 m^2 is an accessory).

Due to the reason that the two best performing zero-shot models were exclusively trained on English data, the models probably made mistakes in unknown words. The default tokenizers used in the transformers pipeline (see Sections 5.4, 5.3) to split the sequences into tokens (words and word parts) might have caused some misclassifications in the tokenization step. Apparently the models can process many German sequences quite well with the tokenized word parts, but the tokenizers' impact was initially underestimated. More thoughts on this are described in the next section, which contains a concluding discussion of the results in the present thesis.

7 Evaluation and Discussion

This section concludes the presented research on zero-shot sequence classification of real estate sequences and includes a summary of the results as well as a discussion in accordance with the relevant literature.

In this thesis, a full workflow from text pre-processing until classification and results evaluation was explained. The text from Austrian real estate purchase contracts was cleaned and sentence parts containing area information as well as context were extracted. It was assumed that the sentence parts contain enough context information to derive what the area information refers to. The goal was to investigate a proper workflow for extraction and classification of real estate areas, especially living area and usable area. A comparative study was conducted to examine the performance of six different transformer-based Large Language Models (LLMs) and a rule-based approach on the classification of the text sequences.

It was assumed that the transformer models are capable of natural language inference (NLI) and therefore can infer through context information what the sentence parts are about – and indirectly what the areas presented in the sentences are about. NLI has already been shown to be effective to classify short text sequences with the use of entailment and without prior training on the respective text domain or labels (e.g. Yin et al., 2019; Sainz and Rigau, 2021). The usage of a classification method that shows considerable results without prior training has the advantage to be rather flexible with adapting the list of target classes. This flexibility can be important when adapting the classes to needs of customers and industry partners.

The data set ($n = 220$ purchase contracts) is from the legal domain and was provided for research purposes by DataScience Service GmbH, a company that purchases and analyzes real estate related documents and other text data on a regular basis. The results and insights from this study are discussed below.

7.1 Results Summary

First, the six zero-shot models were applied with the fine-grained labels in combination with 10 different hypothesis templates. The model performing best with *fine-grained labels* was the model *facebook/bart-large-mnli* (M_6) in combination with the hypothesis template *"Dieses Beispiel ist {label}."* (H_1). Interestingly, this model has only been trained on English text data, but performed well with the German sequences and the German hypothesis template. This finding is new and has not been discovered by the author during intense literature review, since only English text sequences have been classified with that model by the primary literature cited above. Furthermore, model M_6 had the lowest variance across the different hypothesis templates compared to the other models, which indicates rather robust results with this model.

The metric used for assessment of the best performing model was the macro averaged F1-score, which was overall lower for the fine-grained vs. the aggregated labels. This can be explained with the process of label aggregation, where most labels are assigned to the class “other”. Overall, the metrics obtained with the fine-grained labels were for all zero-shot models higher than a baseline model predicting only the majority class. When using aggregated labels, a few accuracy values were below the baseline model predicting only the majority class, whereas all the macro averaged F1 scores were a lot above the established baseline.

The hypothesis template with the on average highest F1-score for fine-grained labels was the German H_9 (“Die Fläche gehört zu {label}.”, engl. “The area belongs to {label}.”). Interestingly, the three models trained on multi-lingual text (including German), performed better on H_9 compared to all the other hypothesis templates applied in this study. Another interesting observation is, that the hypothesis template that performed best in the study by Sainz and Rigau (2021) (H_7) did not perform best our case, but rather as one of the on average lowest performing ones. This finding was unexpected and suggests that the language and domain of the sequences matter for the selection of the hypothesis template.

It was observed, that models with a similar architecture or trained on the same data set performed better with the same hypothesis template compared to other hypothesis templates. Both BART-based models, M_5 and M_6, performed best with H_1 whereas the two RoBERTa-based models trained on XNLI, M_2 and M_3, but also the DeBERTa-based model M_4 trained on XNLI performed best with H_9.

Furthermore, the relevance of punctuation in the hypothesis template was observed. Two hypothesis templates, H_5 and H_6, only differed by the existence of a dot in the end, the template with the dot showed a higher F1 macro across all models. This finding suggests that the extensive sequence pre-processing, cleaning, and harmonization steps, which included standardizing symbols like dots, were effective and necessary to minimize undesired variations in the classification of the sequences.

On average, the English hypothesis templates performed slightly worse compared to the German ones, but with inconsistent results across the models. Mixed trends were observed between the default hypothesis H_0 and the German translation H_1 where some models performed better with the English and some with the German wording. Overall, in hand with the recommendations of model authors (e.g. Davison; Cusumano) the combination of a hypothesis in the same language as the sequence can be recommended.

Second, the evaluation of model results after *label aggregation* was computed. Interestingly, the best performing model changed to M_1, which is namely *roberta-large-mnli* , with H_0 (default hypothesis template) and a F1-score of 0.87. The second highest F1 macro was achieved with M_6 using H_9, which was just 0.02 points lower than the best F1-score computed with the aggregated labels. Furthermore, the model with the highest average F1-score and also lowest SD changed in the aggregated labels also to be M_1.

For both the fine-grained and the aggregated labels the highest average F1-scores were achieved with hypothesis template H_9. The difference of averaged F1-scores of German vs. English hypotheses has shrinked due to label aggregation to almost 0. This is a hint into the direction that with using more labels than needed as outcome and aggregation of classes the language of the hypothesis template matters less.

Reasons for misclassifications with fine-grained and aggregated labels were described in Section 6.3.3 and showed that 35% of misclassifications with aggregated labels were due to sequences where two instead of one label were suitable for the sequences (multi-label cases). Nevertheless, only a quarter of those cases was classified correctly when taking the second label into account[56]. It can be assumed that those sequences were in general more difficult and therefore the correct classification rate lower.

The performance of the zero-shot classification with LLM was compared to a simple rule-based classifier that matched the class names to the pre-processed sequences. With the rule-based approach, a macro F1-score for the fine-grained labels 0.02 higher compared to the best zero-shot macro F1-score was achieved. This result indicates, that the sequences were pre-processed and harmonized well, the labels used as classes matched the sequences and even a basic classifier can perform well on this kind of pre-processed data. Nevertheless, after label aggregation, the F1 macro of the best zero-shot model was 0.04 points above the result of the rule-based classifier. It was described in Sections 6.4 and 6.5, that the rule-based approach got especially high numbers for the “other” class since this was the else-condition of the classification Algorithm. There was no else-condition in the LLM-based zero-shot classification approach and also no lower bound for class assignment was set. The class related to the highest logit value for entailment was selected as the prediction, without further modification for the fine-grained classification result. Overall, the LLM approach without any training on the data set has performed slightly better in comparison to the solely rule-based classification approach. Therefore the LLM approach can be regarded as slightly better than rule-based class-pattern matching. Nevertheless, lower bounds for entailment could be applied to future experiments to improve the results even further.

The models performing best were one RoBERTa-based and one BART-based model. The

[56]Results for M_6, H_1

better performance of these models among others is also in line with the literature, e.g. Pearce et al. (2021), who describe those model architectures as the "highest performing" ones in their study about transformer model comparison in question answering tasks. An interesting observation regarding model size is, that the largest model was not the best one. This is in line with results in the literature (e.g. Zhong et al., 2021; Wei and Zhou, 2023).

7.2 Limitations and Future Directions

Labels and Aggregation

When reporting the results of the rule-based classification, it became obvious that the rule-based approach classified more cases with label *other* compared to the zero-shot-based approach. This is a valid classification approach for this sample, which contained disproportionately many (almost 2/3) cases of class *other*. Nevertheless, label *other* is used as default in the rule-based approach whenever none of the other labels met the pattern. On the other hand, in the zero-shot classification approach, there is no implementation of an *else* condition, since the model does not "know" about the other classes when premise-hypothesis pairs are assessed. Therefore, the "else" condition simply does not exist in the model application and in some cases the class with the highest logit was just a good attempt but not correct guess by the model. This can be a cause for the misclassifications in the investigated sample.

At this point it should be considered, that the model output is a combination of the list of classes and the respective logit values. It sounds like a feasible solution to *a)* either take the logit of every class in respect to the other classes (difference between scores) or *b)* the logit of the most probable class (logit threshold with assigning sequences with only low scores to the "else" class) into account.

During first checks of the zero-shot approach with contracts not contained in the investigated set, both a) and b) have already been tried out. Since there has not been any visible pattern and cases classified correctly were wrongly assigned to the class *other* when taking the logits into account, the logit-threshold and logit-difference were not investigated further. Since with the data set the true classes of the sequences are clear, the logit values could be investigated further and suitable thresholds to avoid misclassification could be implemented as a future step. Hence, the ground truth data generated in this thesis and the logit vectors from the results, will ease future experiments to implement such an else condition and overcome the behaviour of the LLMs to assign not suitable and rather random classes in uncertain cases (like described in Section 6.5).

During the experiments in this study, as well as during experimentation with classification using ChatGPT in the course of another ongoing study, it was found that LLMs perform better with classification tasks when more, especially more suitable and matching, classes are given. The models have difficulties to classify unknown and not suitable classes, and potentially make "good attempts" but wrong guesses. Therefore, the author desires to point into the direction of adding more than less classes for any LLM related classification task. Especially in the course of the experiments conducted in this thesis, it was found that that the models tried out did not classify e.g. cellar well since the cellar was not a distinct class, but subordinated to class *other*. Due to aggregation of the fine-grained results, many misclassifications like the cellar cases were neutralized. Nevertheless, the performance could probably have been improved beforehand with incorporating more classes. Still, natural language inference is working with inference of examples like "this type y is a synonym of class x and therefore class x should get a high probability" and not like "the data scientist applying me as a model wants me to interpret not suitable information as unseen and of class *other*". Since in the literature no research about the performance of LLMs with more vs. less classes in classification tasks or information retrieval was found, this would be an interesting future research and helpful for application of such models.

The paragraphs about the labels above can be seen as limitation. Despite it would take several hours or days to process the 668 sequences from the investigated set again manually and assign new, even more fine-grained classes by hand, it would still be feasible. After reclassification of the sequence set with more class labels, another round of validation with model results could be promising.

Data Processing and Generalization

Moreover, one limitation of the current work is that the thesis author was manually labelling the sequence set alone without validation and quality check from another expert. Therefore, the inter-annotator agreement remains unclear for the sequences used and manually labelled in this study. Whenever a routine task is conducted, it cannot be ruled out that mistakes can happen whenever the attention decreases. In order to avoid ambiguous, conflicting, or inconsistent classifications, samples of the classified sequences were checked 2-3 times and mistakes were fixed whenever encountered in the re-checks. Furthermore, rules for data classification in specific wording-situations were written down as a counter measure of inconsistent classifications (see Appendix B).

Another thought that lays on hand is that pre-processing steps could be improved even further. Since the purchase contracts were regarded as whole documents, without section segmentation or page classification, several areas from appendices were within the extracted sequences. This could be avoided with processing steps beforehand, e.g. page classification. Especially in these contracts, where the appendix pages are scanned (accidentally or on purpose) in the middle of the contract, the page classification would be helpful. Further, a classification of the separate sections within the contract would help to distinguish descriptions of the overall project (e.g. in case of newly build properties) compared to the object of purchase. With the help of pre-classified pages some of the rigid and heuristic pre-processing steps, like the applied maximum number of sequences used per contract,[57] could be omitted.

Whenever an approach to extract information from documents is implemented, it should be considered that the same information can be present in a purchase contract more than once. In the approach presented in this thesis, the deduplication of same-sizes was implemented in way such that *identical sequences* (not sequences with *identical area sizes*) were considered only once (see Section 4.5, Algorithm 3). Depending on the style of the notary, areas (e.g. the area of an apartment) can be mentioned repeatedly and also with a different wording. Whenever the overall area of the object of purchase is considered in combination with the purchase price, it has to be decided if the areas are somehow deduplicated, summed, or post-processed in any other way. Deduplication of areas just by the area itself could be problematic because a contract can be about more than one object of the same area. Deduplication by sequence can be problematic as well because the wording can vary even though the area is referring to the same object of purchase. During data curating process, at least one contract about the transaction of two or more different objects with the same area were observed (e.g. apartment no. 1 and no. 2 having 30 m^2 each are sold together for 400k €). For a potential implementation of a similar approach like the one described in this thesis, where the text is split into short sentence parts, a deduplication with top numbers found nearby the usable or living area could be thought of. This would overcome in many cases the problem of summing up repeated occurrences of the same area as well as help to detect contracts about more than one object (e.g. three apartments where two have the same area).

One limitation of the analysis with contracts in general, is that it is difficult to know enough about the sentence structure of different notaries to incorporate this fact in rules for cleaning and pre-processing. Therefore, it is not clear how the method will perform on all individual contract types of different notaries all over Austria. Even with the knowledge, it would be effortful to write and maintain all the regular expression rules to use them for cleaning and sentence splitting. The data pre-processing steps of the approach described in this thesis are still solely rule-based and there is no recommended and promising approach how to overcome that for the legal domain.

[57]It was 12 in the current study (see Section 4.5.

Moreover, after pre-processing the sequences, they are processed by a tokenizer before being forwarded to the actual transformer model (see Section 5.3). In the current thesis, no specified tokenizer for the underlying domain and language was used. The reason is that there was no tokenizer trained on a German legal corpus available for the zero-shot sequence classification models investigated. Therefore, the default tokenizers provided by the Hugging Face classification pipeline (Hugging Face, 2023e) were applied. Tokenizers are trained on large general (but not specific) text corpora and can be the cause of faulty tokenization in case of domain-specific or legal-specific data sets (Yang et al., 2023; Wahba et al., 2022). The reason for that is that the semantics between words in general vs. legal language can differ (Yang et al., 2023).

Additionally, a more thorough investigation into how LLMs perform with German tokenizers, regardless of the domain, would be useful. It has been observed that sequences matching letters, word parts, or lexically similar words from a label were classified with that label by the LLMs (see Section 6.5). The impact of the English tokenizer on both model performance and the sequences themselves requires further examination, as its influence appears to be greater than initially assumed.

Another matter to be investigated further, is the segmentation of the text. As reported above (see Section 4.8), there have been short sequences in the sample, consisting of just three words. It could be promising to incorporate more context information dynamically and reclassify such sequences whenever the logit scores returned by the zero-shot models are below a certain threshold. Furthermore, a reclassification with e.g. less context information would be interesting for sequences where two or more classes got a similarly high logit with the zero-shot models. In any case, sequence length and extent of the logit scores should be investigated more closely.

Alternative Approaches and Privacy

Overall, comparing the two approaches a) zero-shot classification with a pre-trained language model and b) a rule-based classifier, provided valuable insights into the strengths and limitations of each approach and helped to guide the selection of two approaches that do not need any training data at all.

Since the zero-shot classification approach returned results with an accuracy below 90% and for the satisfaction of customers buying contract data an accuracy of more than 90% is desired, methods to improve the extraction and classification of areas are of potential interest. Besides the improvement and research suggestions above, another improvement possibility lays on hand: In order to obtain better results, it would be possible to further train the LLMs with examples of the manually labelled sequence set. This processes is called few-shot training because in few-shot learning just a *few* examples are presented to the model. As Brown et al. (2020) describes, the state-of-the art transformer models, they are few-shot learners with notable performance increment due to transfer learning and training on a small number of examples. Since all of the models examined in this theses are already trained on NLI tasks with task-specific fine-tuning on data sets of several hundreds and thousands premise-hypothesis pairs, just a few new pairs of the data set for every class could be sufficient. Sarkar et al. (2021) has already shown that legal text classification performs better in a few-shot compared to a zero-shot manner.

Besides that, fine-tuning LEGAL-BERT with labelled data from purchase contracts could be promising as well. E.g. in Van Hofslot et al. (2022) the fine-tuned LEGAL-BERT performed better compared to the out-of-the-box BART-large-mnli. Nevertheless, the researchers in that study already had a big labelled data set available for many-shot training and/or fine-tuning. The fine-tuning of pre-trained base models on the downstream task is only feasible if enough labelled data for the contracts would be available.

Especially GPT-3 and higher are good in generalizing information and structure from few-shot (Brown et al., 2020) since the model has been trained on a vast amount of text data. GPT-3 has 175 billion parameters and is more than 400 times larger than models like BERT-large (Brown et al., 2020; Devlin et al., 2018). Besides the ability to further train LLMs with example pairs of premise and hypothesis, it is also possible to use few-shot classification or information extraction with GPT-3 or higher via prompts (Wei et al., 2023). There have been several publications regarding prompting and how to properly write them for GPT models for various text processing tasks (e.g. Polak and Morgan, 2023; Wei et al., 2023; Li et al., 2023).

Like mentioned in the introductory section, LLMs are continuously improved and there are regularly published newer models performing better on known data sets for NLI, e.g. the HellaSwag data set (Zellers et al., 2019). The human performance on that data set is around 96% whereas the accuracy of BERT-large is around 47%, RoBERTa 85%, GPT-3.5 85%, and GPT-4-base 95% (Zellers et al., 2019; OpenAI, 2023a; Zellers et al., 2023). Originally, when the training and benchmark data set was published, the authors claimed that the poor results of BERT on the data set implicit that BERT is not understanding the text and therefore cannot make logical and common sense inference. With the surprisingly good benchmark results of GPT-4 from March 2023 (OpenAI, 2023a), the commonsense reasoning about everyday events is almost human-level. Nevertheless, model performance on tasks other than benchmark tasks can be worse for very big LLMs since they are often trained on data from the web and could potentially already be trained on parts of the benchmark set which exaggerates the performance (Brown et al., 2020). It should be determined how GPT-4 will perform on analyzing Austrian purchase contracts.

One should consider that since the LLMs are getting better and larger with every generation, it is not possible anymore for many models to download and execute them locally. The models are accessible via API requests, and tasks are submitted in the form of prompts. The largest LLM currently[58] is probably GPT-4, which is a transformer based LLM, trained to perform next word prediction (OpenAI, 2023a). Even though the official report paper about the GPT-4 model release and capabilities does not contain details about data set construction and model size, scientists are speculating that the amount of parameters is around 1 trillion (Albergotti, 2023).

Recently, a more lightweight chat model with an underlying LLM having just 70B parameters[59] has been released open source by Meta: Llama 2 (Touvron et al., 2023). The model has been published in July 2023, only a few months after GPT-4 release and is available for free on the Hugging Face platform (Schmid et al., 2023). Even though Llama 2 has a lower performance compared to GPT-4, it is more light weight and a serious competition to recent GPT models (Saha, 2023). Therefore, the investigation for the use of purchase contracts would be of interest, especially because usage is for free even for commercial use.

Even though real estate purchase contracts in Austria are publicly available, it does not seem intuitive to send parts of a legal document via an API to some server [60], without the certainty how the data will be used for further processing and public availability. This was apparently meriting consideration for business users of OpenAI, which is why OpenAI does not use the data or prompts submitted by users via their API to train or improve their models (anymore), unless desired by the user, according to the latest API data usage policies from June 2023 (OpenAI, 2023b).

To have full control over the data security, local model training, and model storage can be considered. There have been approaches with smaller[61] LLM like BERT to train them in a privacy preserving manner (e.g. Ying and Habernal, 2022; Yu et al., 2023). Nevertheless, pre-training an LLMs like BERT is an expensive task, the default generic vocabulary that does not match the legal-domain usually stays unchanged though out the training and the performance improvements on legal domain specific tasks have shown to be up to 6.7% (Ying

[58]Please note that the research on LLM sizes was performed in July 2023.

[59]This is even smaller than GPT-3 by OpenAI.

[60]A thought regarding security can be the storage of data outside Europe, which can be the case for LLMs used via an API. For example OpenAI does not support EU data residency yet (June 2023) and stores data for 30 days for misuse checks (OpenAI, 2023b).

[61]compared to GPT-4

and Habernal, 2022; Zheng et al., 2021). Model performance after training on a legal data set with a privacy preserving method has shown to be up to 2% better compared to baseline methods. Despite Ying and Habernal (2022) assess this as a success, it can be discussed if this improvement is worth the time, effort, and training resources. Through literature review, it is not clear if few-shot training with BERT, BART and similar rather small LLMs will increase model performance enough to be worth the effort and would increase the accuracy of sequence classification above 90%.

Moreover, recent research has shown that via a process called *using membership inference attacks* it is possible to investigate if certain instances have been part of the training data set, which can be an issue for data privacy (Carlini et al., 2022). Therefore, methods are investigated to “forget” training examples (Pedregosa and Triantafillou, 2023).

Concluding for the zero-shot classification with LLMs in general, it can be stated that the approach is especially useful when the outcome labels can change or training is not desired for any other reason. In situations where there are many labels to classify, but there is not enough labeled data for each individual category, zero-shot classification can be a useful approach. For example, in the case of real estate descriptions, there are many different features or attributes to classify, but there may not be enough labeled data for each individual feature or attribute. And it may not be fixed beforehand which features or attributes should be classified.

Using models capable of zero-shot classification can also be useful in situations where new categories or labels may emerge over time, as it allows the model to adapt to new categories without requiring retraining.

An important finding highlighted in this study is that if the text sequence is clean and contains enough context information, and the class labels are uniquely assignable, zero-shot classification of real estate contract sequences is applicable and yields sufficiently good results for an initial baseline investigation. Nevertheless, future research, workflow adaptions and training are needed to achieve human-like performance.

Overall, the languages spoken and written by humans have been the basis for communication and innovation ever since. In the last years, the research field of Natural Language Processing has increased in popularity, with many publications and innovative models being available. We all can look forward to exciting future model releases, applications making life easier and exciting papers with surprising experiments.

References

Jennifer Abel and Birger Lantow. A methodological framework for dictionary and rule-based text classification. pages 330–337, 01 2019. doi: 10.5220/0008121503300337.

Reed Albergotti. The secret history of elon musk, sam altman, and openai, Mar 2023. URL `https://www.semafor.com/article/03/24/2023/the-secret-history-of-elon-musk-sam-altman-and-openai`. Last visited on 2023-07-09.

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preotiuc-Pietro, and Vasileios Lampos. Predicting judicial decisions of the european court of human rights: a natural language processing perspective. *PeerJ Computer Science*, 2, 2016. doi: 10.7717/peerj-cs.93.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv*, 2016. doi: 10.48550/arXiv.1607.06450.

Ben. What is the difference between a loss function and an error function? - answer 1, 2020. URL `https://stats.stackexchange.com/questions/359043/what-is-the-difference-between-a-loss-function-and-an-error-function#:~:text=An%20error%20function%20measures%20the,negative%20consequence%20of%20an%20error`. Last visited on 2023-10-30.

BEV. Cadastre and Land Register, 2023. URL `https://www.bev.gv.at/en/Topics/Cadastre.html`. Last visited on 2023-08-19.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *arXiv*, 2015.

Bernhard Brandauer. Kaufverträge und Treuhandschaft - Alles über die Abwicklung von Kaufverträgen über Immobilien. URL `https://brandauer-rechtsanwaelte.at/immobilien/kaufvertraege/`. Last visited on 2023-07-12.

Bernhard Brandauer. Die wichtigsten Zutaten zum perfekten Immobilienkaufvertrag, 2021. URL `https://brandauer-rechtsanwaelte.at/2021/02/11/die-wichtigsten-zutaten-zum-perfekten-immobilienkaufvertrag/`. Last modified on 2021-02-11. Last visited on 2022-07-31.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv*, 2020. doi: 10.48550/arXiv.2306.14101.

Jason Brownlee. What are large language models, 2023. URL `https://machinelearningmastery.com/what-are-large-language-models/`. Last Updated on 2023-07-20, last visited on 2023-08-04.

Bundesministerium für Finanzen. Grundbuch - Grundbuchseinsicht, 2023. URL `https://www.oesterreich.gv.at/themen/bauen_wohnen_und_umwelt/grundbuch/Seite.600300.html`. Last visited on 2023-03-12.

Bundesministerium für Justiz. Grundbuch - Urkundensammlung, 2023. URL `https://www.oesterreich.gv.at/themen/bauen_wohnen_und_umwelt/grundbuch/Seite.600140.html`. Last visited on 2023-03-12.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. *arXiv*, 2022. doi: 10.48550/arXiv.2112.03570.

Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. Extracting contract elements. In *Proceedings of the 16th Edition of the International Conference on Artificial Intelligence and Law*, ICAIL '17, page 19–28, New York, NY, USA, 2017. Association for Computing Machinery. doi: 10.1145/3086512.3086515.

Ilias Chalkidis, Manos Fergadiotis, Sotiris Kotitsas, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. An empirical study on large-scale multi-label text classification including few and zero-shot labels. In *Conference on Empirical Methods in Natural Language Processing*, 2020a. doi: 10.18653/v1/2020.emnlp-main.607.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school, 2020b.

Peter Chapman, Julian Clinton, Randy Kerber, Tom Khabaza, Thomas P. Reinartz, Colin Daimlerchrysler, Rüdiger Shearer, and Wirth. Crisp-dm 1.0: Step-by-step data mining guide. 2000. URL `https://api.semanticscholar.org/CorpusID:59777418`.

Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, and Soumya K. Ghosh. *Optical Character Recognition Systems*, pages 9–41. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-50252-6_2.

Xieling Chen, Haoran Xie, and Xiaohui Tao. Vision, status, and research topics of Natural Language Processing. *Natural Language Processing Journal*, 1:100001, 2022. ISSN 2949-7191. doi: https://doi.org/10.1016/j.nlp.2022.100001.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. *arXiv*, 2020. doi: 10.48550/arXiv.2003.10555.

Collins English Dictionary. gibberish definition and meaning. URL `https://www.collinsdictionary.com/dictionary/english/gibberish`. Last visited on 2024-03-10.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov.

Unsupervised cross-lingual representation learning at scale. *arXiv*, 2020. doi: 10.48550/arXiv.1911.02116.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

Giovanni Cusumano. Model card of: Jiva/xlm-roberta-large-it-mnli. URL `https://huggingface.co/Jiva/xlm-roberta-large-it-mnli`. Last visited on 2023-04-02.

Joe Davison. Model card of: joeddav/xlm-roberta-large-xnli. URL `https://huggingface.co/joeddav/xlm-roberta-large-xnli`. Last visited on 2023-04-02.

Jin Dawei. The Application of Date Mining in Knowledge Management. In *2011 Fifth International Conference on Management of e-Commerce and e-Government*, pages 7–9, 2011. doi: 10.1109/ICMeCG.2011.58.

Redaktion Der Standard. Quadratmeterpreise in Österreich: Wie hoch sind sie?, 2021. URL `https://immobilien.derstandard.at/mieten-und-kaufen/quadratmeterpreise-in-oesterreich-wie-hoch-sind-sie/`. Published on 2021-04-21, Last visited on 2023-08-19.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018. doi: 10.48550/arXiv.1810.04805. Last revised 24 May 2019.

Harald Draxl. Wohnflächenberechnung: So gehen Sie vor, 2023. URL `https://www.infina.at/ratgeber/wohnflaechenberechnung/`. Last modified on 2022-12-19, Last visited on 2023-08-28.

Research group Facebook AI. Model card of: roberta-large-mnli, 2019. URL `https://huggingface.co/roberta-large-mnli`. Last visited on 2023-07-25.

James Frew and G. Jud. Estimating the value of apartment buildings. *Journal of Real Estate Research*, 25:77–86, 02 2003. doi: 10.1080/10835547.2003.12091101.

Victor Gallego. Model card of: vicgalle/xlm-roberta-large-xnli-anli. URL `https://huggingface.co/vicgalle/xlm-roberta-large-xnli-anli`. Last visited on 2023-07-25.

Editorial Team of Globalnegotiator. Dictionary of International Trade: What is title transfer?, 2017. URL `https://www.globalnegotiator.com/international-trade/dictionary/title-transfer/`. Last visited on 2024-04-19.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv*, 2020. doi: 10.48550/arXiv.2008.05756.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data, 2018.

Ivan Habernal, Daniel Faber, Nicola Recchia, Sebastian Bretthauer, Iryna Gurevych, Indra Spiecker genannt Döhmann, and Christoph Burchard. Mining legal arguments in court decisions. *arXiv*, 2023. doi: arXiv.2208.06178.

Eya Hammami, Rim Faiz, and Imen Akermi. *A Dynamic Convolutional Neural Network Approach for Legal Text Classification.* Springer International Publishing, 2021. doi: 10.1007/978-3-030-85977-0_6.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv*, 2021. doi: 10.48550/arXiv.2006.03654.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv*, 2023. doi: 10.48550/arXiv.2111.09543. Submitted on 18 Nov 2021 (v1), last revised 24 Mar 2023 (this version, v4).

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. *arXiv*, 2020. doi: 10.48550/arXiv.2004.06100.

Team Hugging Face. Model card of: xlm-roberta-large, 2020. URL `https://huggingface.co/xlm-roberta-large`. Last visited on 2023-07-25.

Team Hugging Face. Byte-pair encoding tokenization, 2023a. URL `https://huggingface.co/learn/nlp-course/chapter6/5`. Last visited on 2024-04-01.

Team Hugging Face. Transformers pipeline documentation, 2023b. URL `https://huggingface.co/docs/transformers/v4.18.0/en/index`. Last visited on 2023-07-10.

Team Hugging Face. Main nlp tasks: Fine-tuning a masked language model, 2023c. URL `https://huggingface.co/learn/nlp-course/chapter7/3?fw=pt#fine-tuning-a-masked-language-model`. Last visited on 2023-07-12.

Team Hugging Face. Model documentation of: xlm-roberta, 2023d. URL `https://huggingface.co/docs/transformers/model_doc/xlm-roberta`. Last visited on 2023-07-25.

Team Hugging Face. Transformers pipeline documentation, 2023e. URL `https://huggingface.co/docs/transformers/main_classes/pipelines#transformers.ZeroShotClassificationPipeline`. Last visited on 2023-03-10.

Team Hugging Face. Summary of the tokenizers, 2024. URL `https://huggingface.co/docs/transformers/main/en/tokenizer_summary`. Last visited on 2024-03-30.

jaygala260 Community Member. Difference between a Neural Network and a Deep Learning System, 2022. URL `https://www.geeksforgeeks.org/difference-between-a-neural-network-and-a-deep-learning-system/`. Last modified on 2022-09-19.

Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

Pratik Joshi, Somak Aditya, Aalok Sathe, and Monojit Choudhury. TaxiNLI: Taking a ride up the NLU hill. *arXiv*, 2020. doi: 10.48550/arXiv.2009.14505.

Dan Jurafsky and James H. Martin. *Regular Expressions, Text Normalization, Edit Distance*, page 4–30. 3rd (draft) edition, 2023a. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *Transformers and Pretrained Language Models*, page 211–227. 3rd (draft) edition, 2023b. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *Fine-Tuning and Masked Language Models*, page 228–243. 3rd (draft) edition, 2023c. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *NLP Application - Machine Translation*, pages 247–268. 3rd (draft) edition, 2023d. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *Naive Bayes and Sentiment Classification*, pages 57–77. 3rd (draft) edition, 2023e. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *Vector Semantics and Embeddings*, pages 102–133. 3rd (draft) edition, 2023f. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *Neural Networks and Neural Language Models*, pages 134–159. 3rd (draft) edition, 2023g. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Dan Jurafsky and James H. Martin. *RNNs and LSTMs*, page 185–210. 3rd (draft) edition, 2023h. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Amirhossein Kazemnejad. Transformer architecture: The positional encoding, 2019. URL `https://kazemnejad.com/blog/transformer_architecture_positional_encoding/`. Last visited on 2023-07-23.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 82 (3):3713–3744, 2022. doi: 10.1007/s11042-022-13428-4.

Helene Kortschak. Attention and transformer models - a complex algorithm, simply explained, 2020. URL `https://towardsdatascience.com/attention-and-transformer-models-fe667f958378`. Last visited on 2023-07-23.

Rudolf Kruse, Christian Borgelt, Christian Braune, Sanaz Mostaghim, and Matthias Steinbrecher. *Computational Intelligence.* Springer London, 2nd edition, 2016. doi: 10.1007/978-1-4471-7296-3.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates, 2018.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv*, 2018. doi: 10.48550/arXiv.1808.06226.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv*, 2020. doi: https://doi.org/10.48550/arXiv.1909.11942.

Moritz Laurer. Model card of: MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7. URL `https://huggingface.co/MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7`. Last visited on 2023-04-02.

Moritz Laurer, Wouter van Atteveldt, Andreu Salleras Casas, and Kasper Welbers. Less Annotating, More Classifying – Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT - NLI. *Preprint*, June 2022. URL `https://osf.io/74b8k`. Publisher: Open Science Framework.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL `http://arxiv.org/abs/1910.13461`.

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. Evaluating ChatGPT's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. *arXiv*, 2023. doi: 10.48550/arXiv.2304.11633.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *arXiv*, 2021. doi: 10.48550/arXiv.2008.00364.

Zheng Li, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. DQ-BART: Efficient sequence-to-sequence model via joint distillation and quantization. *arXiv*, 2022. doi: 10.48550/arXiv.2203.11239.

Elizabeth D Liddy. *Natural Language Processing*. Marcel Decker, Inc, NY, 2nd edition, 2001. URL `https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub`.

Cai-zhi Liu, Yan-xiu Sheng, Zhi-qiang Wei, and Yong-Quan Yang. Research of text classification based on improved tf-idf algorithm. In *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, pages 218–222, 2018. doi: 10.1109/IRCE.2018.8492945.

Jiandong Liu, Ruibin Bai, Zheng Lu, Peiming Ge, Uwe Aickelin, and Daoyun Liu. Data-driven regular expressions evolution for medical text classification using genetic programming. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020. doi: 10.1109/CEC48606.2020.9185500.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*, 2019. doi: 10.48550/arXiv.1907.11692.

Hariharan Manikandan, Yiding Jiang, and J Zico Kolter. Language models are weak learners. *arXiv*, 2023. doi: 10.48550/arXiv.2306.14101.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL `https://aclanthology.org/J93-2004`.

M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417, 1961. doi: 10.1145/321075.321084.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv*, 2019. doi: 10.48550/arXiv.1902.01007.

Liz McQuillan. BERT: How to Handle Long Documents, 2021. URL https://www.saltdatalabs.com/blog/bert-how-to-handle-long-documents. Last modified on 2021-11-06. Last visited on 2024-01-28.

Liz McQuillan. Deep Learning 101: Transformer Activation Functions Explainer - Sigmoid, ReLU, GELU, Swish, 2022. URL https://www.saltdatalabs.com/blog/deep-learning-101-transformer-activation-functions-explainer_relu-leaky-relu-gelu-elu-selu-softmax-and-more. Last modified on 2022-12-18. Last visited on 2022-09-07.

Ted Mei. From static embedding to contextualized embedding, 2020. URL https://ted-mei.medium.com/from-static-embedding-to-contextualized-embedding-fe604886b2bc. Last visited on 2023-10-30.

Microsoft. Model card of: microsoft/mdeberta-v3-base. URL https://huggingface.co/microsoft/mdeberta-v3-base. Last visited on 2023-04-02.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv*, 2013. doi: 10.48550/arXiv.1301.3781.

Mayank Mishra. Convolutional neural networks, explained, 2020. URL https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. Last visited on 2023-09-01.

Elena Musi, Manfred Stede, Leonard Kriese, Smaranda Muresan, and Andrea Rocci. A multi-layer annotated corpus of argumentative text: From argument schemes to discourse relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1258.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

Editorial oesterreich.gv.at. Nutzwertberechnung, 2023. URL https://www.oesterreich.gv.at/lexicon/N/Seite.990074.html. Last modified on 2023-04-07, Last visited on 2023-08-28.

OGH. Entscheidungstext Zivilrecht, Geschäftszahl 5Ob130/03h, Abgrenzung zwischen Loggia und Terrasse, 2003. URL https://www.ris.bka.gv.at/Dokumente/Justiz/JJT_20031209_OGH0002_0050OB00130_03H0000_000/JJT_20031209_OGH0002_0050OB00130_03H0000_000.html. Last visited on 2023-03-11.

OpenAI. GPT-4 Technical Report. *arXiv*, 2023a. doi: 10.48550/arXiv.2303.08774.

OpenAI. API data usage policies, 2023b. URL https://openai.com/policies/api-data-usage-policies. Last updated on 2023-06-14. Last visited on 2023-07-16.

Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv*, 2021. doi: 10.48550/arXiv.1911.03347.

Melih Oznalbant. What is the Rule Based Classification and why we use??, 2023. URL `https://medium.com/@melih.oznalbant92/what-is-the-rule-based-classification-and-why-we-use-e4c6f6364156`. Published on 2023-06-21.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, page 79–86, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118704.

Suraj Patil. Model card of: microsoft/valhalla/distilbart-mnli-12-1. URL `https://huggingface.co/valhalla/distilbart-mnli-12-1`. Last visited on 2023-04-02.

Suraj Patil. Gitlab Repository DistilBart-MNLI, 2020. URL `https://github.com/patil-suraj/distillbart-mnli`. Last visited on 2024-04-01.

Kate Pearce, Tiffany Zhan, Aneesh Komanduri, and Justin Zhan. A comparative study of transformer-based language models on extractive question answering. *arXiv*, 2021. URL `https://arxiv.org/abs/2110.03142`.

Fabian Pedregosa and Eleni Triantafillou. Announcing the first machine unlearning challenge, June 2023. URL `https://ai.googleblog.com/2023/06/announcing-first-machine-unlearning.html`. Last visited on 2023-08-07.

Maciej P. Polak and Dane Morgan. Extracting accurate materials data from research papers with conversational language models and prompt engineering. *arXiv*, 2023. doi: 10.48550/arXiv.2303.05352.

Preethi Prakash. Understanding Baseline Models in Machine Learning, 2023. URL `https://medium.com/@preethi_prakash/understanding-baseline-models-in-machine-learning-3ed94f03d645`. Published on 2023-04-17.

J. Pustejovsky and A. Stubbs. *Natural Language Annotation for Machine Learning*. Number Ed. 9 in A Guide to corpus-building for applications. O'Reilly Media, Incorporated, 2012. URL `https://books.google.at/books?id=QtzmqamXxx4C`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018. URL `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Alec Radford, Jeffrey Wu, Dario Amodei, and Miles Brundage. Better language models and their implications, 2019. URL `https://openai.com/research/better-language-models`. Last visited on 2023-07-10.

Sebastian Ruder. Natural language inference, 2020. URL `http://nlpprogress.com/english/natural_language_inference.html`. Last visited on 2023-08-21.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986. URL `https://api.semanticscholar.org/CorpusID:62245742`.

Shritama Saha. Llama 2 vs GPT-4 vs Claude-2, July 2023. URL `https://analyticsindiamag.com/llama-2-vs-gpt-4-vs-claude-2/`. Last visited on 2023-08-07.

Oscar Sainz and German Rigau. Ask2transformers: Zero-shot domain labelling with pre-trained language models. *arXiv*, 2021. doi: 10.48550/arXiv.2101.02661.

Rajdeep Sarkar, Atul Kr. Ojha, Jay Megaro, John Mariano, Vall Herard, and John P. McCrae. Few-shot and zero-shot approaches to legal text classification: A case study in the financial sector. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 102–106, Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nllp-1.10.

Philipp Schmid, Omar Sanseviero, Pedro Cuenca, and Lewis Tunstall. Llama 2 is here - get it on Hugging Face, July 2023. URL `https://huggingface.co/blog/llama2`. Last visited on 2023-08-07.

Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

scikit-learn developers. scikit-learn documentation: Chapter 3.3. metrics and scoring, 2023. URL `https://scikit-learn.org/stable/modules/model_evaluation.html#from-binary-to-multiclass-and-multilabel`. Last visited on 2023-05-22.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1162.

Zein Shaheen, Gerhard Wohlgenannt, and Erwin Filtz. Large scale legal text classification using transformer models. *CoRR*, abs/2010.12871, 2020. doi: 10.48550/arXiv.2010.12871.

Parul Sharma. Feature engineering, 2020. URL `https://ted-mei.medium.com/from-static-embedding-to-contextualized-embedding-fe604886b2bc`. Last visited on 2023-10-30.

Redaktion Statistik Austria. Immobilien-Durchschnittspreise, 2023. URL `https://www.statistik.at/statistiken/volkswirtschaft-und-oeffentliche-finanzen/preise-und-preisindizes/immobilien-durchschnittspreise`. Last modified on 2023-08-09, Last visited on 2023-08-19.

Octavia-Maria Sulea, Marcos Zampieri, Shervin Malmasi, Mihaela Vela, Liviu P. Dinu, and Josef van Genabith. Exploring the use of text classification in the legal domain. *arXiv*, 2017. doi: 10.48550/arXiv.1710.09306.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, and Vedanuj Goswami ... . Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 2023. doi: 10.48550/arXiv.2307.09288.

Marieke Van Hofslot, Almila Akdag Salah, Albert Gatt, and Cristiana Santos. Automatic classification of legal violations in cookie banner texts. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 287–295, Abu Dhabi, United Arab Emirates (Hybrid), 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.nllp-1.27.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017. doi: 10.48550/arXiv.1706.03762. Submitted on 12 Jun 2017 (v1), last revised 2 Aug 2023 (this version, v7).

Yasmen Wahba, Nazim Madhavji, and John Steinbacher. A comparison of svm against pre-trained language models (plms) for text classification tasks, 2022.

Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9, 04 2022. doi: 10.1007/s40745-020-00253-5.

Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL `https://aclanthology.org/P12-2018`.

Steven H. Wang, Antoine Scardigli, Leonard Tang, Wei Chen, Dimitry Levkin, Anya Chen, Spencer Ball, Thomas Woodside, Oliver Zhang, and Dan Hendrycks. Maud: An expert-annotated legal nlp dataset for merger agreement understanding, 2023.

Jerry Wei and Denny Zhou. Larger language models do in-context learning differently, May 2023. URL `https://ai.googleblog.com/2023/05/larger-language-models-do-in-context.html`. Last visited on 2023-07-17.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. Zero-shot information extraction via chatting with ChatGPT. *arXiv*, 2023. doi: 10.48550/arXiv.2302.10205.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv*, 2019. doi: 10.48550/arXiv.1911.00359.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, 2020. European Language Resources Association. URL `https://www.aclweb.org/anthology/2020.lrec-1.494`.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Hui Yang, Stella Hadjiantoni, Yunfei Long, Ruta Petraityte, and Berthold Lausen. Automatic detection of industry sectors in legal articles using machine learning approaches. *arXiv*, 2023. doi: 10.48550/arXiv.2303.05387.

Mahdieh Yazdani and Maziar Raissi. Real estate property valuation using self-supervised vision transformers. *arXiv*, 2023. doi: 10.48550/arXiv.2302.00117.

Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv*, 2019. doi: 10.48550/ARXIV.1909.00161.

Yin Ying and Ivan Habernal. Privacy-Preserving Models for Legal Natural Language Processing. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, page (to appear), Abu Dhabi, UAE, 2022.

Sixing Yu, J. Pablo Muñoz, and Ali Jannesari. Federated foundation models: Privacy-preserving and collaborative learning for large models. *arXiv*, 2023. doi: 10.48550/arXiv.2305.11414.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv*, 2018. doi: 10.48550/arXiv.1808.05326.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv*, 2019. doi: 10.48550/arXiv.1905.07830.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag Leaderboard. Benchmark approaches to HellaSwag, 2023. URL `https://rowanzellers.com/hellaswag/#leaderboard`. Last visited on 2023-07-01.

Alebachew Zewdu and Betselot Yitagesu. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9, 01 2022. doi: 10.1186/s40537-022-00561-y.

Guolong Zhao, Yuling Liu, and E. Erdun. Review on intelligent processing technologies of legal documents. In *Artificial Intelligence and Security: 8th International Conference, ICAIS 2022, Qinghai, China, July 15–20, 2022, Proceedings, Part I*, page 684–695, Berlin, Heidelberg, 2022. Springer-Verlag. doi: 10.1007/978-3-031-06794-5_55.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arXiv*, 2023. doi: 10.48550/arXiv.2303.18223.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. *arXiv*, 2021.

Ruiqi Zhong, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. Are larger pretrained language models uniformly better? comparing performance at the instance level. *arXiv*, 2021. doi: 10.48550/arXiv.2105.06020.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *arXiv*, 2023. doi: 10.48550/arXiv.2302.09419.

§ 2 BGBl. II Nr. 116. Verordnung des Bundesministers für Wirtschaft, Familie und Jugend über die Angabe und Definition der Benützungsarten und Nutzungen im Grenzkataster (Benützungsarten-Nutzungen-Verordnung – BANU – V), 2010. URL `https://www.ris.bka.gv.at/eli/jgs/1811/946/P1053/NOR12018782#:~:text=Paragraph%201053%2C,durch%20die%20Uebergabe%20des%20Kaufgegenstandes`.

§§ 1053 ff. ABGB. Allgemeines bürgerliches Gesetzbuch (ABGB). Von dem Kaufvertrage, 1812. URL `https://www.ris.bka.gv.at/eli/jgs/1811/946/P1053/NOR12018782#:~:text=Paragraph%201053%2C,durch%20die%20Uebergabe%20des%20Kaufgegenstandes`. Last modified on 2015-04-08.

Österreichische Notariatskammer. Kaufvertrag und Grundbuch, 2022. URL `https://www.oesterreich.gv.at/themen/bauen_wohnen_und_umwelt/grundstueckskauf/3/Seite.200053.html`. Last visited on 2023-03-12.

Österreichische Notariatskammer. Notarfinder - Mein Notariat in der Nähe. Schnell gefunden, 2023a. URL `https://www.notar.at/notarfinder/`. Last visited on 2023-03-12.

Österreichische Notariatskammer. Ihr Notariat in Österreich – Vorausdenken für Generationen, 2023b. URL `https://www.notar.at/`. Last visited on 2023-03-12.

A Purchase Contract Example

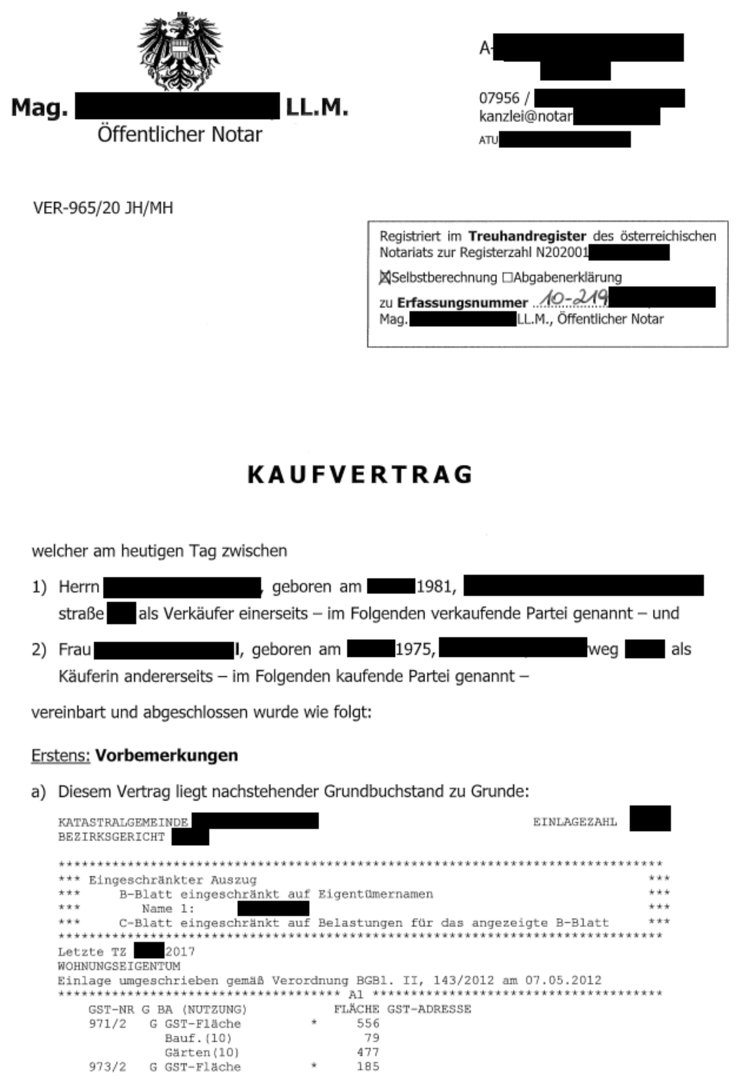
The Figures 28, 29, 30 and 31 display the 8 pages of a short[62] contract that was part of the sample under investigation. The black marks on the pages were placed by the author to protect the identity of all contract parties.

The document contains the usual parts and sections of a real estate purchase contract. On page 1 (Figure 28a), there is the title, with preceding notary information, and underneath the names and personal information of the contract parties are given in enumeration style. Further, on this page the preamble (ger: Vorbemerkungen) starts and describes the ownership status and the location of the object of purchase. The ownership description contains a part of a land register extract regarding the property that is sold (page 1-2). The usable area of the property is further described on page 2 (Figure 28b).

The purchase price is stated on page 3 (Figure 29a), the legal conditions of ownership change is stated afterwards. A summary of the transferred parts of the property is given on page 6, followed by a signature of buyer and seller (Figure 30b).

This is a contract without appendixes, amendments or figures. The last two pages only refer to the authentication of the seller and buyer signature (Figure 31a) and the authentication of the notary (Figure 31b).

[62]Short referring to the contract having less pages than more than 75% of the contracts in the sample investigated.



Mag. [redacted] **LL.M.**
Öffentlicher Notar

A-[redacted]
[redacted]
07956 / [redacted]
kanzlei@notar[redacted]
ATU[redacted]

VER-965/20 JH/MH

Registriert im **Treuhandregister** des österreichischen Notariats zur Registerzahl N202001[redacted]
[✓ Selbstberechnung] ☐Abgabenerklärung
zu **Erfassungsnummer** 10-219[redacted]
Mag.[redacted]LL.M., Öffentlicher Notar

KAUFVERTRAG

welcher am heutigen Tag zwischen

1) Herrn [redacted], geboren am [redacted]1981, [redacted]straße [redacted] als Verkäufer einerseits – im Folgenden verkaufende Partei genannt – und

2) Frau [redacted]l, geboren am [redacted]1975, [redacted]weg [redacted] als Käuferin andererseits – im Folgenden kaufende Partei genannt –

vereinbart und abgeschlossen wurde wie folgt:

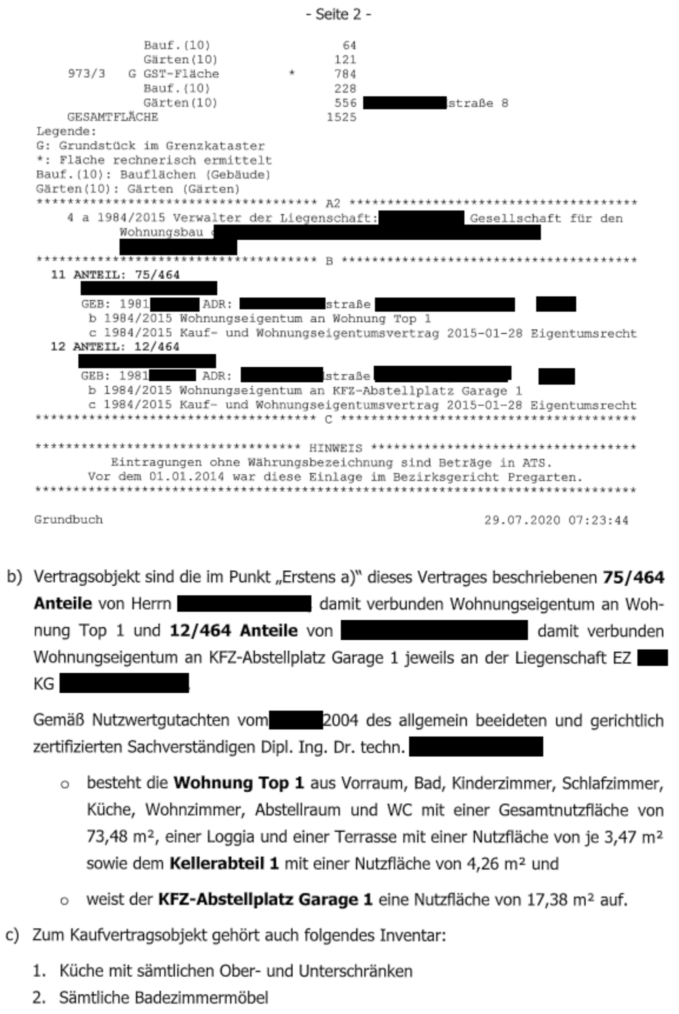
Erstens: **Vorbemerkungen**

a) Diesem Vertrag liegt nachstehender Grundbuchstand zu Grunde:

```
KATASTRALGEMEINDE [redacted]                              EINLAGEZAHL [redacted]
BEZIRKSGERICHT [redacted]

*******************************************************************************
*** Eingeschränkter Auszug                                                  ***
***     B-Blatt eingeschränkt auf Eigentümernamen                           ***
***        Name 1:      [redacted]                                          ***
***     C-Blatt eingeschränkt auf Belastungen für das angezeigte B-Blatt    ***
*******************************************************************************
Letzte TZ [redacted]2017
WOHNUNGSEIGENTUM
Einlage umgeschrieben gemäß Verordnung BGBl. II, 143/2012 am 07.05.2012
************************************** A1 *************************************
    GST-NR G BA (NUTZUNG)           FLÄCHE GST-ADRESSE
    971/2   G GST-Fläche         *     556
              Bauf.(10)                 79
              Gärten(10)               477
    973/2   G GST-Fläche         *     185
```

(a) Page 1 of 8.



- Seite 2 -

```
              Bauf.(10)                 64
              Gärten(10)               121
    973/3   G GST-Fläche         *     784
              Bauf.(10)                228
              Gärten(10)               556 [redacted]straße 8
    GESAMTFLÄCHE                      1525
Legende:
G: Grundstück im Grenzkataster
*: Fläche rechnerisch ermittelt
Bauf.(10): Bauflächen (Gebäude)
Gärten(10): Gärten (Gärten)
************************************** A2 *************************************
    4 a 1984/2015 Verwalter der Liegenschaft: [redacted] Gesellschaft für den
          Wohnungsbau [redacted]
************************************** B **************************************
  11 ANTEIL: 75/464
     [redacted]
     GEB: 1981[redacted] ADR: [redacted]straße [redacted]   [redacted]
      b 1984/2015 Wohnungseigentum an Wohnung Top 1
      c 1984/2015 Kauf- und Wohnungseigentumsvertrag 2015-01-28 Eigentumsrecht
  12 ANTEIL: 12/464
     [redacted]
     GEB: 1981[redacted] ADR: [redacted]straße [redacted]   [redacted]
      b 1984/2015 Wohnungseigentum an KFZ-Abstellplatz Garage 1
      c 1984/2015 Kauf- und Wohnungseigentumsvertrag 2015-01-28 Eigentumsrecht
************************************** C **************************************

********************************** HINWEIS ************************************
        Eintragungen ohne Währungsbezeichnung sind Beträge in ATS.
      Vor dem 01.01.2014 war diese Einlage im Bezirksgericht Pregarten.
*******************************************************************************

Grundbuch                                                  29.07.2020 07:23:44
```

b) Vertragsobjekt sind die im Punkt „Erstens a)" dieses Vertrages beschriebenen **75/464 Anteile** von Herrn [redacted] damit verbunden Wohnungseigentum an Wohnung Top 1 und **12/464 Anteile** von [redacted] damit verbunden Wohnungseigentum an KFZ-Abstellplatz Garage 1 jeweils an der Liegenschaft EZ [redacted] KG [redacted].

Gemäß Nutzwertgutachten vom[redacted]2004 des allgemein beeideten und gerichtlich zertifizierten Sachverständigen Dipl. Ing. Dr. techn. [redacted]

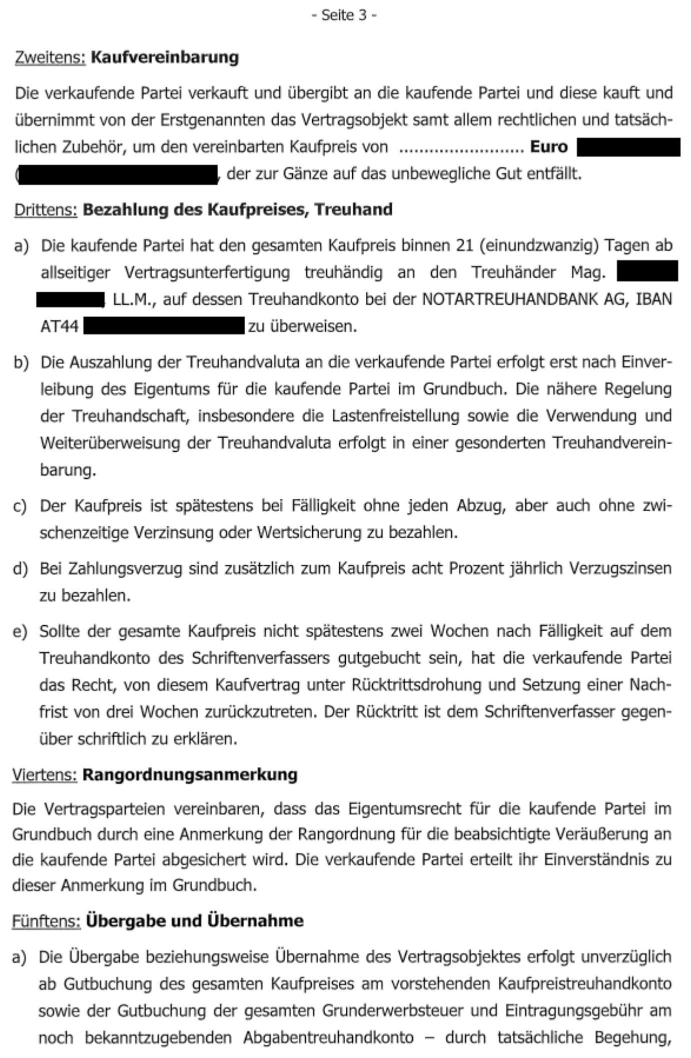
- besteht die **Wohnung Top 1** aus Vorraum, Bad, Kinderzimmer, Schlafzimmer, Küche, Wohnzimmer, Abstellraum und WC mit einer Gesamtnutzfläche von 73,48 m², einer Loggia und einer Terrasse mit einer Nutzfläche von je 3,47 m² sowie dem **Kellerabteil 1** mit einer Nutzfläche von 4,26 m² und
- weist der **KFZ-Abstellplatz Garage 1** eine Nutzfläche von 17,38 m² auf.

c) Zum Kaufvertragsobjekt gehört auch folgendes Inventar:

1. Küche mit sämtlichen Ober- und Unterschränken
2. Sämtliche Badezimmermöbel

(b) Page 2 of 8.

Figure 28: Two contract pages (1-2).



- Seite 3 -

Zweitens: **Kaufvereinbarung**

Die verkaufende Partei verkauft und übergibt an die kaufende Partei und diese kauft und übernimmt von der Erstgenannten das Vertragsobjekt samt allem rechtlichen und tatsächlichen Zubehör, um den vereinbarten Kaufpreis von .... **Euro** [redacted] ([redacted], der zur Gänze auf das unbewegliche Gut entfällt.

Drittens: **Bezahlung des Kaufpreises, Treuhand**

a) Die kaufende Partei hat den gesamten Kaufpreis binnen 21 (einundzwanzig) Tagen ab allseitiger Vertragsunterfertigung treuhändig an den Treuhänder Mag. [redacted] [redacted], LL.M., auf dessen Treuhandkonto bei der NOTARTREUHANDBANK AG, IBAN AT44 [redacted] zu überweisen.

b) Die Auszahlung der Treuhandvaluta an die verkaufende Partei erfolgt erst nach Einverleibung des Eigentums für die kaufende Partei im Grundbuch. Die nähere Regelung der Treuhandschaft, insbesondere die Lastenfreistellung sowie die Verwendung und Weiterüberweisung der Treuhandvaluta erfolgt in einer gesonderten Treuhandvereinbarung.

c) Der Kaufpreis ist spätestens bei Fälligkeit ohne jeden Abzug, aber auch ohne zwischenzeitige Verzinsung oder Wertsicherung zu bezahlen.

d) Bei Zahlungsverzug sind zusätzlich zum Kaufpreis acht Prozent jährlich Verzugszinsen zu bezahlen.

e) Sollte der gesamte Kaufpreis nicht spätestens zwei Wochen nach Fälligkeit auf dem Treuhandkonto des Schriftenverfassers gutgebucht sein, hat die verkaufende Partei das Recht, von diesem Kaufvertrag unter Rücktrittsdrohung und Setzung einer Nachfrist von drei Wochen zurückzutreten. Der Rücktritt ist dem Schriftenverfasser gegenüber schriftlich zu erklären.

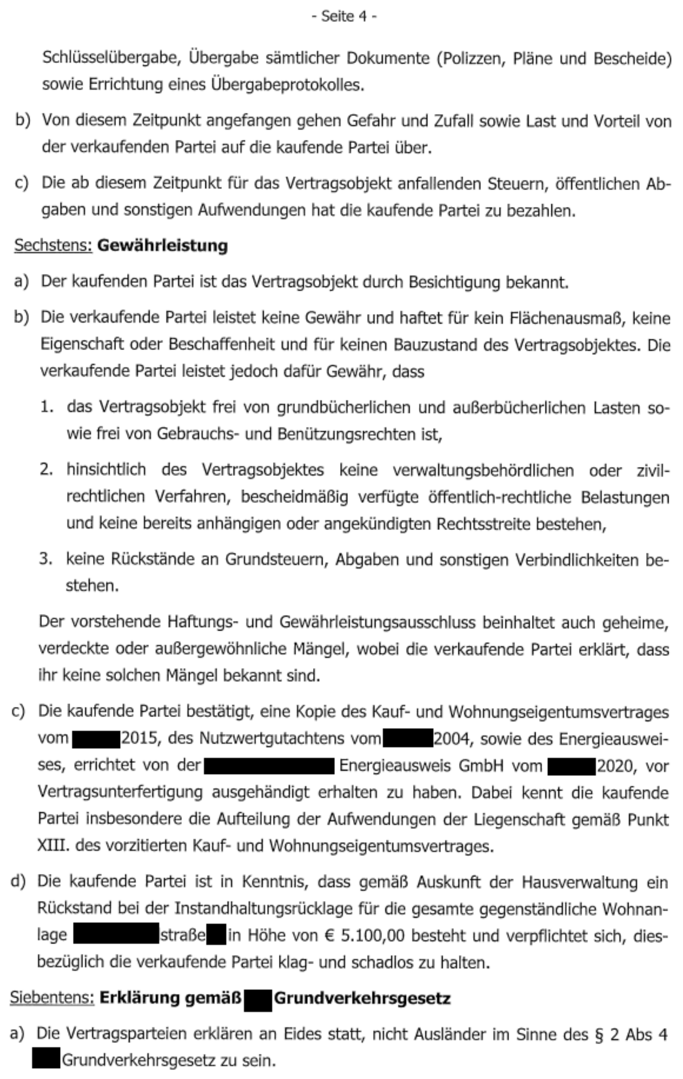
Viertens: **Rangordnungsanmerkung**

Die Vertragsparteien vereinbaren, dass das Eigentumsrecht für die kaufende Partei im Grundbuch durch eine Anmerkung der Rangordnung für die beabsichtigte Veräußerung an die kaufende Partei abgesichert wird. Die verkaufende Partei erteilt ihr Einverständnis zu dieser Anmerkung im Grundbuch.

Fünftens: **Übergabe und Übernahme**

a) Die Übergabe beziehungsweise Übernahme des Vertragsobjektes erfolgt unverzüglich ab Gutbuchung des gesamten Kaufpreises am vorstehenden Kaufpreistreuhandkonto sowie der Gutbuchung der gesamten Grunderwerbsteuer und Eintragungsgebühr am noch bekanntzugebenden Abgabentreuhandkonto – durch tatsächliche Begehung,

(a) Page 3 of 8.



- Seite 4 -

Schlüsselübergabe, Übergabe sämtlicher Dokumente (Polizzen, Pläne und Bescheide) sowie Errichtung eines Übergabeprotokolles.

b) Von diesem Zeitpunkt angefangen gehen Gefahr und Zufall sowie Last und Vorteil von der verkaufenden Partei auf die kaufende Partei über.

c) Die ab diesem Zeitpunkt für das Vertragsobjekt anfallenden Steuern, öffentlichen Abgaben und sonstigen Aufwendungen hat die kaufende Partei zu bezahlen.

Sechstens: **Gewährleistung**

a) Der kaufenden Partei ist das Vertragsobjekt durch Besichtigung bekannt.

b) Die verkaufende Partei leistet keine Gewähr und haftet für kein Flächenausmaß, keine Eigenschaft oder Beschaffenheit und für keinen Bauzustand des Vertragsobjektes. Die verkaufende Partei leistet jedoch dafür Gewähr, dass

1. das Vertragsobjekt frei von grundbücherlichen und außerbücherlichen Lasten sowie frei von Gebrauchs- und Benützungsrechten ist,
2. hinsichtlich des Vertragsobjektes keine verwaltungsbehördlichen oder zivilrechtlichen Verfahren, bescheidmäßig verfügte öffentlich-rechtliche Belastungen und keine bereits anhängigen oder angekündigten Rechtsstreite bestehen,
3. keine Rückstände an Grundsteuern, Abgaben und sonstigen Verbindlichkeiten bestehen.

Der vorstehende Haftungs- und Gewährleistungsausschluss beinhaltet auch geheime, verdeckte oder außergewöhnliche Mängel, wobei die verkaufende Partei erklärt, dass ihr keine solchen Mängel bekannt sind.

c) Die kaufende Partei bestätigt, eine Kopie des Kauf- und Wohnungseigentumsvertrages vom [redacted]2015, des Nutzwertgutachtens vom [redacted]2004, sowie des Energieausweises, errichtet von der [redacted] Energieausweis GmbH vom [redacted]2020, vor Vertragsunterfertigung ausgehändigt erhalten zu haben. Dabei kennt die kaufende Partei insbesondere die Aufteilung der Aufwendungen der Liegenschaft gemäß Punkt XIII. des vorzitierten Kauf- und Wohnungseigentumsvertrages.

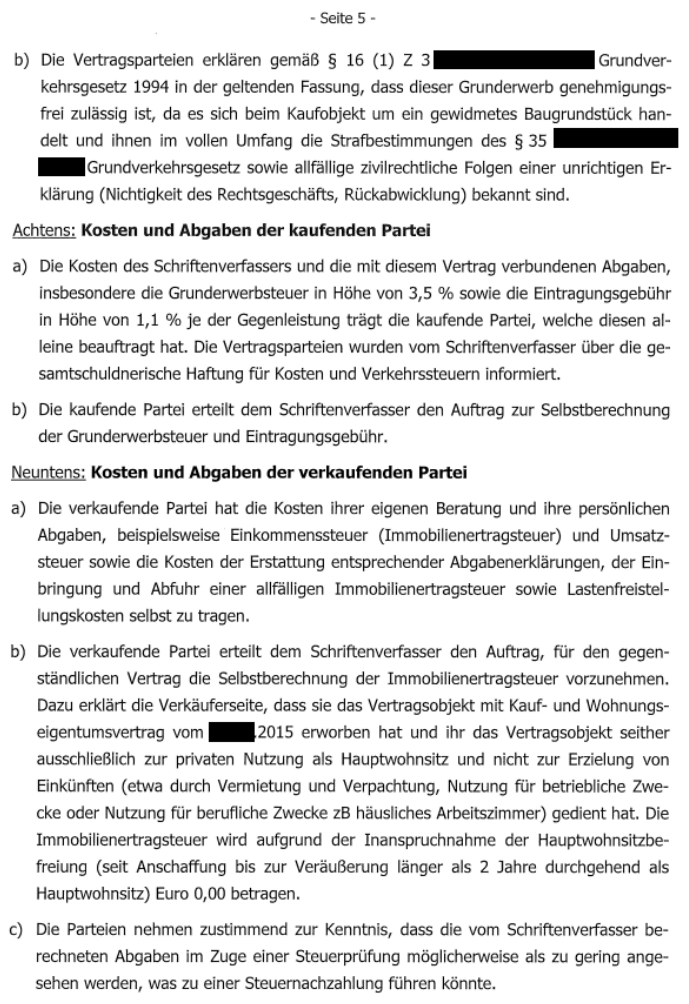
d) Die kaufende Partei ist in Kenntnis, dass gemäß Auskunft der Hausverwaltung ein Rückstand bei der Instandhaltungsrücklage für die gesamte gegenständliche Wohnanlage [redacted]straße[redacted]in Höhe von € 5.100,00 besteht und verpflichtet sich, diesbezüglich die verkaufende Partei klag- und schadlos zu halten.

Siebentens: **Erklärung gemäß [redacted] Grundverkehrsgesetz**

a) Die Vertragsparteien erklären an Eides statt, nicht Ausländer im Sinne des § 2 Abs 4 [redacted]Grundverkehrsgesetz zu sein.

(b) Page 4 of 8.

Figure 29: Two contract pages (3-4).



- Seite 5 -

b) Die Vertragsparteien erklären gemäß § 16 (1) Z 3 [redacted] Grundverkehrsgesetz 1994 in der geltenden Fassung, dass dieser Grunderwerb genehmigungsfrei zulässig ist, da es sich beim Kaufobjekt um ein gewidmetes Baugrundstück handelt und ihnen im vollen Umfang die Strafbestimmungen des § 35 [redacted] [redacted] Grundverkehrsgesetz sowie allfällige zivilrechtliche Folgen einer unrichtigen Erklärung (Nichtigkeit des Rechtsgeschäfts, Rückabwicklung) bekannt sind.

Achtens: **Kosten und Abgaben der kaufenden Partei**

a) Die Kosten des Schriftenverfassers und die mit diesem Vertrag verbundenen Abgaben, insbesondere die Grunderwerbsteuer in Höhe von 3,5 % sowie die Eintragungsgebühr in Höhe von 1,1 % je der Gegenleistung trägt die kaufende Partei, welche diesen alleine beauftragt hat. Die Vertragsparteien wurden vom Schriftenverfasser über die gesamtschuldnerische Haftung für Kosten und Verkehrssteuern informiert.

b) Die kaufende Partei erteilt dem Schriftenverfasser den Auftrag zur Selbstberechnung der Grunderwerbsteuer und Eintragungsgebühr.

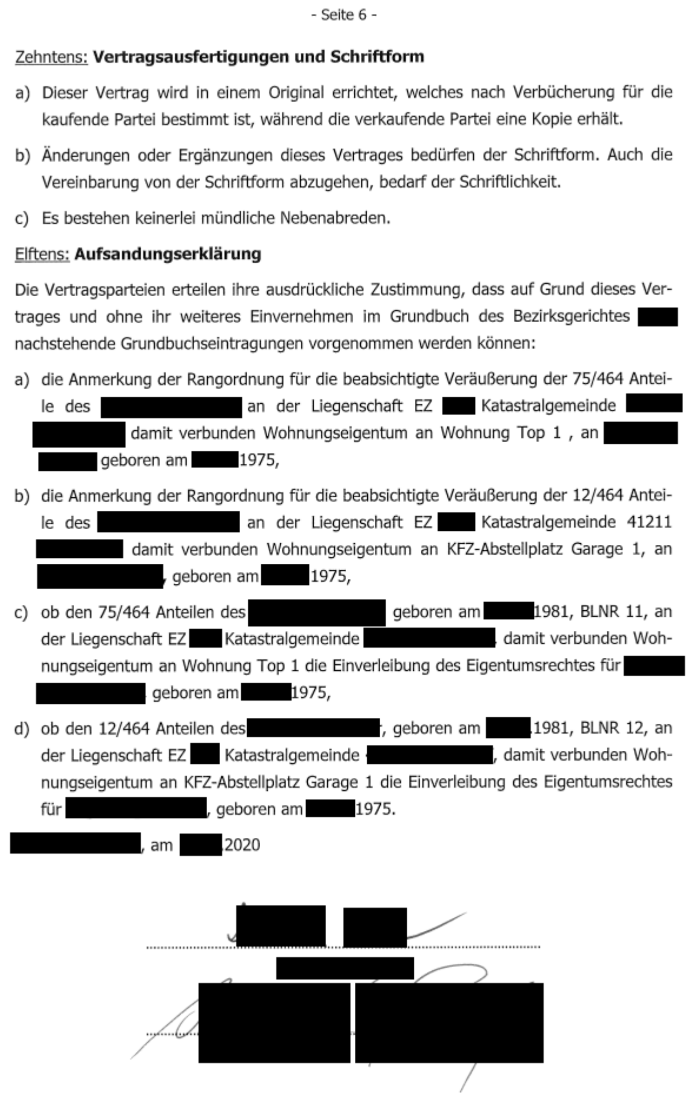
Neuntens: **Kosten und Abgaben der verkaufenden Partei**

a) Die verkaufende Partei hat die Kosten ihrer eigenen Beratung und ihre persönlichen Abgaben, beispielsweise Einkommenssteuer (Immobilienertragsteuer) und Umsatzsteuer sowie die Kosten der Erstattung entsprechender Abgabenerklärungen, der Einbringung und Abfuhr einer allfälligen Immobilienertragsteuer sowie Lastenfreistellungskosten selbst zu tragen.

b) Die verkaufende Partei erteilt dem Schriftenverfasser den Auftrag, für den gegenständlichen Vertrag die Selbstberechnung der Immobilienertragsteuer vorzunehmen. Dazu erklärt die Verkäuferseite, dass sie das Vertragsobjekt mit Kauf- und Wohnungseigentumsvertrag vom [redacted] 2015 erworben hat und ihr das Vertragsobjekt seither ausschließlich zur privaten Nutzung als Hauptwohnsitz und nicht zur Erzielung von Einkünften (etwa durch Vermietung und Verpachtung, Nutzung für betriebliche Zwecke oder Nutzung für berufliche Zwecke zB häusliches Arbeitszimmer) gedient hat. Die Immobilienertragsteuer wird aufgrund der Inanspruchnahme der Hauptwohnsitzbefreiung (seit Anschaffung bis zur Veräußerung länger als 2 Jahre durchgehend als Hauptwohnsitz) Euro 0,00 betragen.

c) Die Parteien nehmen zustimmend zur Kenntnis, dass die vom Schriftenverfasser berechneten Abgaben im Zuge einer Steuerprüfung möglicherweise als zu gering angesehen werden, was zu einer Steuernachzahlung führen könnte.

(a) Page 5 of 8.



- Seite 6 -

Zehntens: **Vertragsausfertigungen und Schriftform**

a) Dieser Vertrag wird in einem Original errichtet, welches nach Verbücherung für die kaufende Partei bestimmt ist, während die verkaufende Partei eine Kopie erhält.

b) Änderungen oder Ergänzungen dieses Vertrages bedürfen der Schriftform. Auch die Vereinbarung von der Schriftform abzugehen, bedarf der Schriftlichkeit.

c) Es bestehen keinerlei mündliche Nebenabreden.

Elftens: **Aufsandungserklärung**

Die Vertragsparteien erteilen ihre ausdrückliche Zustimmung, dass auf Grund dieses Vertrages und ohne ihr weiteres Einvernehmen im Grundbuch des Bezirksgerichtes [redacted] nachstehende Grundbuchseintragungen vorgenommen werden können:

a) die Anmerkung der Rangordnung für die beabsichtigte Veräußerung der 75/464 Anteile des [redacted] an der Liegenschaft EZ [redacted] Katastralgemeinde [redacted] [redacted] damit verbunden Wohnungseigentum an Wohnung Top 1 , an [redacted] [redacted] geboren am [redacted] 1975,

b) die Anmerkung der Rangordnung für die beabsichtigte Veräußerung der 12/464 Anteile des [redacted] an der Liegenschaft EZ [redacted] Katastralgemeinde 41211 [redacted] damit verbunden Wohnungseigentum an KFZ-Abstellplatz Garage 1, an [redacted], geboren am [redacted] 1975,

c) ob den 75/464 Anteilen des [redacted] geboren am [redacted] 1981, BLNR 11, an der Liegenschaft EZ [redacted] Katastralgemeinde [redacted] damit verbunden Wohnungseigentum an Wohnung Top 1 die Einverleibung des Eigentumsrechtes für [redacted] [redacted] geboren am [redacted] 1975,

d) ob den 12/464 Anteilen des [redacted], geboren am [redacted] 1981, BLNR 12, an der Liegenschaft EZ [redacted] Katastralgemeinde [redacted], damit verbunden Wohnungseigentum an KFZ-Abstellplatz Garage 1 die Einverleibung des Eigentumsrechtes für [redacted], geboren am [redacted] 1975.

[redacted], am [redacted] 2020

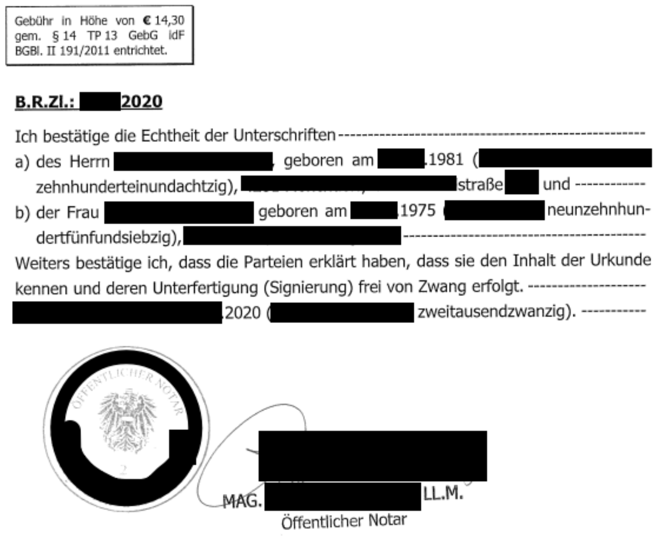
[redacted] [redacted]

[redacted]

[redacted] [redacted]

(b) Page 6 of 8.

Figure 30: Two contract pages (5-6).



Gebühr in Höhe von € 14,30 gem. § 14 TP 13 GebG idF BGBl. II 191/2011 entrichtet.

B.R.Zl.: [redacted]2020

Ich bestätige die Echtheit der Unterschriften ---

a) des Herrn [redacted], geboren am [redacted].1981 ([redacted] zehnhunderteinundachtzig), [redacted]straße[redacted] und ------------

b) der Frau [redacted] geboren am [redacted].1975 ([redacted] neunzehnhundertfünfundsiebzig), [redacted] ---

Weiters bestätige ich, dass die Parteien erklärt haben, dass sie den Inhalt der Urkunde kennen und deren Unterfertigung (Signierung) frei von Zwang erfolgt. --------------------

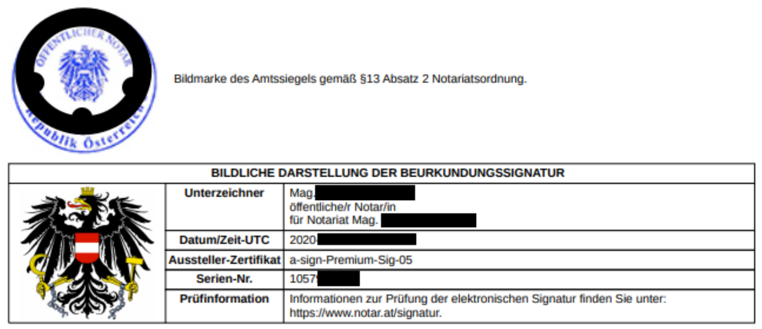
[redacted].2020 ([redacted] zweitausendzwanzig). -----------

ÖFFENTLICHER NOTAR

[redacted]

MAG. [redacted] LL.M.
Öffentlicher Notar

(a) Page 7 of 8.



ÖFFENTLICHER NOTAR
Republik Österreich

Bildmarke des Amtssiegels gemäß §13 Absatz 2 Notariatsordnung.

BILDLICHE DARSTELLUNG DER BEURKUNDUNGSSIGNATUR		
	Unterzeichner	Mag.[redacted] öffentliche/r Notar/in für Notariat Mag. [redacted]
	Datum/Zeit-UTC	2020[redacted]
	Aussteller-Zertifikat	a-sign-Premium-Sig-05
	Serien-Nr.	1057[redacted]
	Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://www.notar.at/signatur.

(b) Page 8 of 8.

Figure 31: Two contract pages (7-8).

B Rules for Ground Truth Labelling

In order to have data for evaluation and model comparison purpose, a ground truth was generated. This section reports some rules used to generate the ground truth of the sample. The rules displayed in Table 18 below were noted down whenever a case was encountered to make sure the cases are treated consistently all the time. Some rules were not noted down because the cases were clear and obvious.

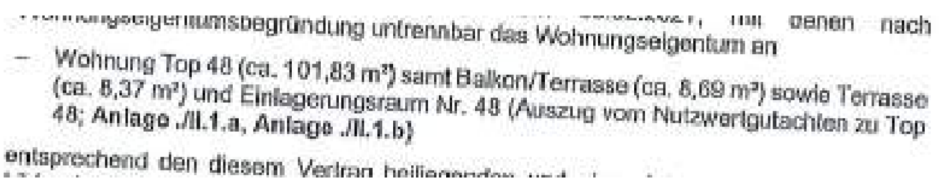
Table 18: Terms and phrases that were noted down during generation of the ground truth data set. The notes were taken in German, but are explained with English comments. In the first column, the English notes are within the parentheses after the German ones. The labels reported here are fine-grained labels, displayed in German (second column) and English (third column). Multi-label cases are noted with "beides" (ger.) or "both" (engl.).

term/phrase (with notes)	**label (ger.)**	**label (engl.)**
Gesamtausmaß Wohnung	Wohnfläche	living area
Wohnung mit Fläche von	Wohnfläche	living area
Wohnung: … m^2	Wohnfläche	living area
Wohnung im Ausmaß	Wohnfläche	living area
Wohnung mit Dachterrasse im Ausmaß von	Terrasse	terrace
mit einer Fläche von (short sentence, no ref to or context info about object)	Sonstige	other
… im Ausmaß von (with reference to a class not in the labels, e.g. Lokal)	Sonstige	other
Kellerabteil im Ausmaß	Sonstige	other
Garage Nutzfläche	Nutzfläche	usable area
Kellerabteil mit einer Nutzfläche	Nutzfläche	usable area
mit einer Nutzfläche von	Nutzfläche	usable area
Nutzfläche exklusive Balkon	Nutzfläche	usable area
Nutzfläche ... Wohnungseigentumsobjekts	Nutzfläche	usable area
Wohnung ... Nutzfläche von	Nutzfläche	usable area
Loggia im Ausmaß	Loggia	loggia
KFZ-Abstellplätze je	KFZ	motor vehicle
Carportstellplatz	KFZ	motor vehicle
Stellplatz im Ausmaß	KFZ	motor vehicle
Gst im Ausmaß	Grundstücksfläche	plot area
Gesamtausmaß (short sentence, no ref to or context info about object)	Gesamtfläche	total area
Gesamtausmaß (with reference being far away, comma inbetween)	Gesamtfläche	total area
im Gesamtausmaß von (short sentence, no ref to or context info about object)	Gesamtfläche	total area
Wohnnutzfläche (with OCR mistake)	beides: Wohnfläche, Nutzfläche	both: living area, usable area
Wohnung ... m^2… Balkon … m3 (or some other add on, where sentence split failed due to faulty m^2)	beides: Wohnfläche, Balkon	both: living area, balcony
Bad … m^2 (or any other common room type)	beides: Sonstige, Zimmer	both: other, room
Schlafzimmer im Ausmaß von	beides: Sonstige, Zimmer	both: other, room
Loggia ... Nutzfläche von	beides: Nutzfläche, Loggia	both: usable area, loggia
Liegenschaft … Gesamtausmaß	beides: Grundstücksfläche, Gesamtfläche	both: plot area, total area
Grundstück mit Gesamtgröße	beides: Grundstücksfläche, Gesamtfläche	both: plot area, total area
Grundstück im Gesamtausmaß	beides: Grundstücksfläche, Gesamtfläche	both: plot area, total area
GST im Gesamtausmaß	beides: Gesamtfläche, Grundstücksfläche	both: total area, plot area
Gesamtausmaß Gärten	beides: Garten, Gesamtfläche	both: garden, total area

C Edge Cases and Other Issues

C.1 Scan Quality

In the investigated sample, there were no PDFs with extraordinary low scan quality. Nevertheless, they can occur and make area extraction difficult. Figure 32 and Figure 33 show paragraphs of purchase contracts outside the test sample, that have a bad scan quality for illustration purpose.

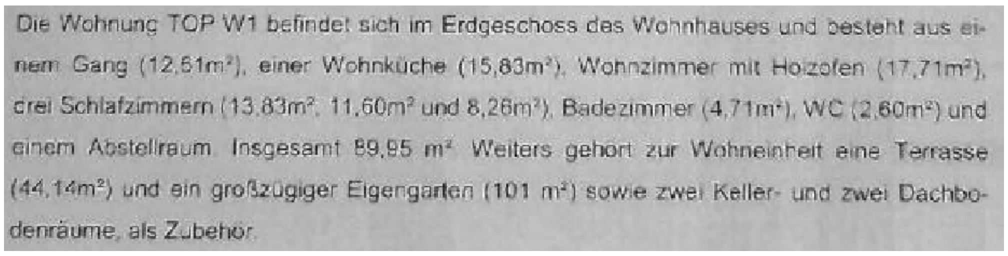


Wohnungseigentumsbegründung untrennbar das Wohnungseigentum an [illegible] mit denen nach

– Wohnung Top 48 (ca. 101,83 m²) samt Balkon/Terrasse (ca. 8,69 m²) sowie Terrasse (ca. 8,37 m²) und Einlagerungsraum Nr. 48 (Auszug vom Nutzwertgutachten zu Top 48; Anlage ./II.1.a, Anlage ./II.1.b)

entsprechend den diesem Vertrag beiliegenden [illegible]

Figure 32: Paragraph of a scan with a slight rotation and low resolution, which is causing deteriorated scan and OCR quality.



Die Wohnung TOP W1 befindet sich im Erdgeschoss des Wohnhauses und besteht aus einem Gang (12,51m²), einer Wohnküche (15,83m²), Wohnzimmer mit Holzofen (17,71m²), drei Schlafzimmern (13,83m², 11,60m² und 8,26m²), Badezimmer (4,71m²), WC (2,60m²) und einem Abstellraum. Insgesamt 89,95 m². Weiters gehört zur Wohneinheit eine Terrasse (44,14m²) und ein großzügiger Eigengarten (101 m²) sowie zwei Keller- und zwei Dachbodenräume, als Zubehör.

Figure 33: Paragraph of a scan with too little contrast, which is causing deteriorated scan and OCR quality.

C.2 OCR Issues

Especially when areas matter, cleaning steps need to be involved. Eventhough cleaning serves the purpose of harmonization, it cannot solve the problem of missing characters all the time. Some examples, where OCR issues were not able to be fixed, can be seen below:

> *"mit welchen Wohnungseigentum an den Räumlichkeiten W 5(im Ausmaß von ca 49 18 m² samt Terrasse mit ca 13,37m2) und Einlagerungsraum 5(die Räumlichkeiten untrennbar verbunden ist(die kaufgegenständlichen Liegenschaftsanteile)."*

The raw, which means not cleaned, sentence part above is a good example to point out importance of cleaning steps to add or remove spacing and harmonize how square meters are written. OCR issues in numbers cannot be fixed despite the usefulness of cleaning. The area of the apartment lacks the comma, so it is not clear where the split should happen. One could think about adding a comma separator whenever there is a space within a number. But the example sentence below shows that this is not a straight forward rule:

> *"Bauplatz, der eine Gesamtfläche von ca 34 34 m² umfasst"*

In this case the space is random and the area of the total project area is >3k square meter. If a dot would have been added as comma separator, the area for an unbuilt plot with the purpose to build it, would be far to small.

C.3 Wordings of Contracts

Even though it is not an edge case per se, it should be mentioned that contracts are a type of text that contains surprises. Without knowledge about the domain, one would assume contracts contain information or behave in a way to be expected. Several points of unexpected cases are described below. Those were encountered during generation of a ground truth for the contract sample.

No Areas

One would assume that contracts contain areas in general. Nevertheless, contracts can contain phrases like the German examples below:

1. *"Über Zustand und das tatsächliche Flächenausmaß konnte sich der Käufer im Rahmen der Besichtigung selbst ein Bild machen und seiner Kaufentscheidung zu Grunde legen."*
2. *"Der Verkäufer haftet nicht für den Zustand, das Ausmaß, Erträgnis oder eine besondere Beschaffenheit des Vertragsobjektes."*

Both phrases are examples for German wording in contracts that describes that the area of the property is not written in the contract. Either because the buyer knows in real live how big the object of purchase is (1.) or the seller does not want to be legally accountable for a certain condition of the property (2.).

Sentence Length

Sentences can be long and understanding is tricky even for humans tricky. One example of a long sentence:

> *"1.2. Die Vertragsparteien erklären beziehungsweise haben Kenntnis davon, dass*
> *1.2.1. das Vertragsobjekt der der Frau [redacted] gehörende Hälfteanteil, B-LNR [redacted], der Liegenschaft Einlagezahl 303 Katastralgemeinde [redacted] im unverbürgten Ausmaß von 966 m^2(neunhundertsechsundsechzig Quadratmeter) samt dem sich darauf befindlichen Einfamilienhaus ist;"*

Typos

Mistakes can happen - even in the legal domain. Whenever there are typos, switched letters and symbols in the sentences of interest, it is more difficult to extract the true areas. Two examples of typos and similar mistakes are:

- *"Vorraum und Abstellraum im Aumaß von 56 m^2"*: In this sentence part the letter 's' is missing in Ausmaß (ger. Ausmaß, engl. extend).

- *“2.1. Die verkaufende Partei verkauft [...] folgende Anteile: 66/22345 Anteile, mit denen untrennbar das Wohnungseigentum an der Wohnung Top 8.09 samt Balkon und ER 8.09 im Ausmaß von etwa 58.0558,05 m2(Wohn- nutzfläche), etwa 9,42m2 Balkon gewidmet als Wohnung, entsprechend [...] verbunden ist(im Folgenden„Woh- nung").“*: In this shortened sentence the notary made a copy and paste mistake where the area was pasted twice. Therefore the area is far too high for an apartment.

D Additional Plots

D.1 Location of Transactions

The sample analyzed in this thesis was drawn from all real estate purchase contracts the data providing company DataScience Service owns. The location of the contracts / the properties the contracts from the sample are about are displayed in the Figures 34 and 35 with red dots. Since the sample was

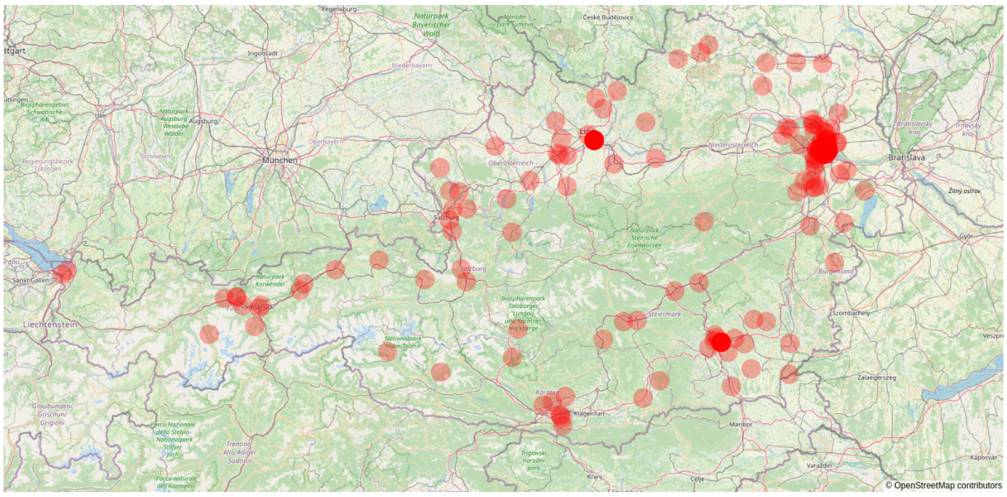


Figure 34: Location of transactions in the test data sample ($n = 220$).

drawn from all real estate purchase contracts DataScience Service owns, with a limitation to the time frame (see section 3), the total amount of contracts the sample was drawn from is $> 450,000$. The locations of all the contracts DataScience Service owns ($> 500,000$) is displayed in the Figure 35, with red dots indicating the location of the contracts from the random sample analyzed in this study.

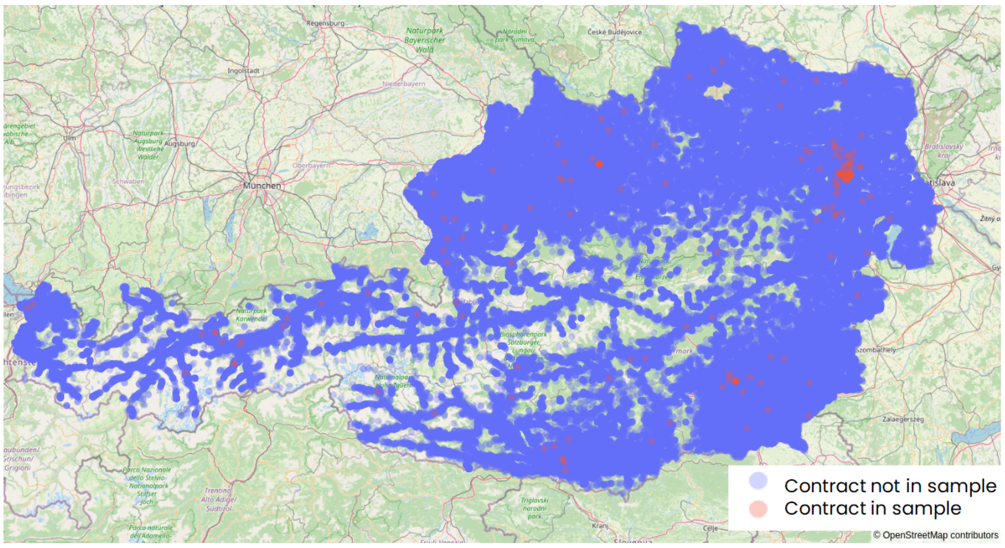


Figure 35: Location of all real estate purchase contracts of DataScience Service (reference date: 2023-07-07). Color red indicates that contracts are in the study sample.

D.2 Heatmaps Filtered for Correct/Incorrect Classifications

This appendix section contains heatmaps comparing the model-based vs. rule-based classification results, filtered for correct (Figure 36) or incorrect (Figure 37) classifications.

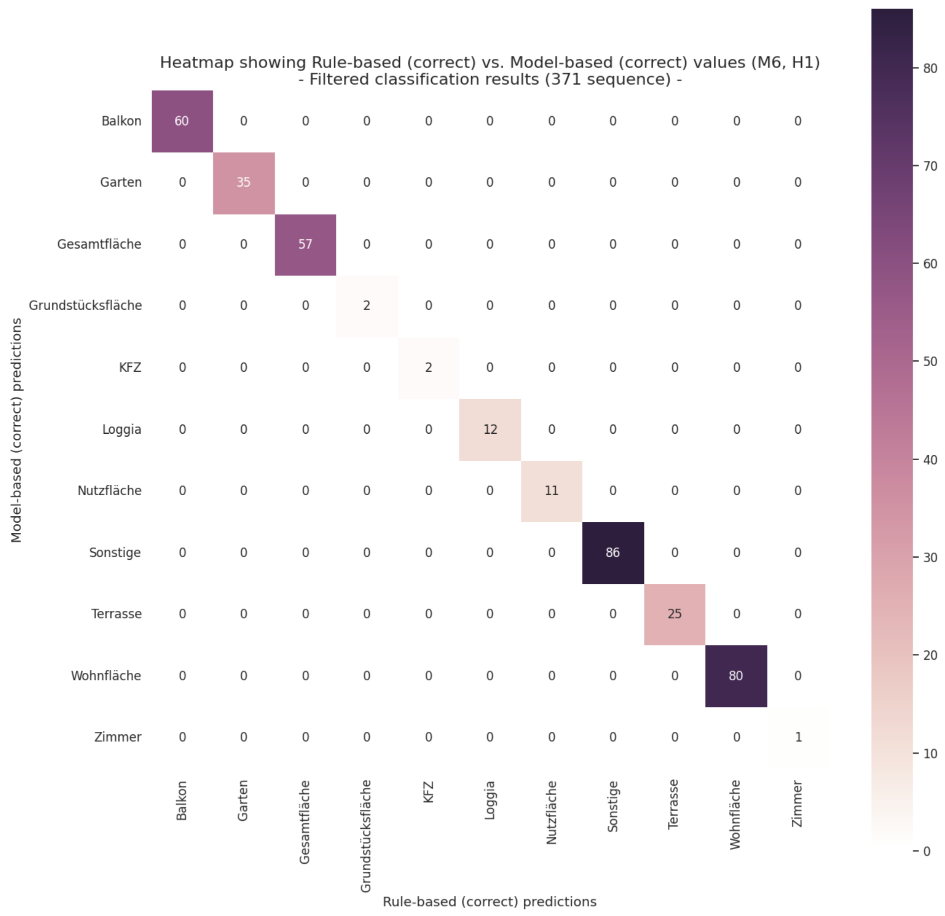


Figure 36: Heatmap of fine-grained results comparing rule-based and model-based (M_6, H_1) values. The values were filtered so the figure only displays the classification results, where both methods classified the sequences correctly.

The 371 out of 668 sequences, where both methods classified them correctly, are displayed in Figure 36 along the diagonal. The largest classes, with most sequences being assigned to in the manual ground truth, were *Grundstücksfläche* (engl. plot area), *Sonstige* (engl. other), and *Wohnfläche* (engl. living area), each occurring more than 100 times within the 668 sequences (see Section 4.8). Interestingly especially the class *plot area* shows few correct classification by both methods.

The 136 out of 668 sequences, where both methods (rule-based versus model-based using the model hypothesis combination M_6, H_1) misclassified them, are displayed in Figure 37. Due to the rule-based approach having an else-condition favoring the class *other*, there are many misclassifications across that respective column in the heatmap.

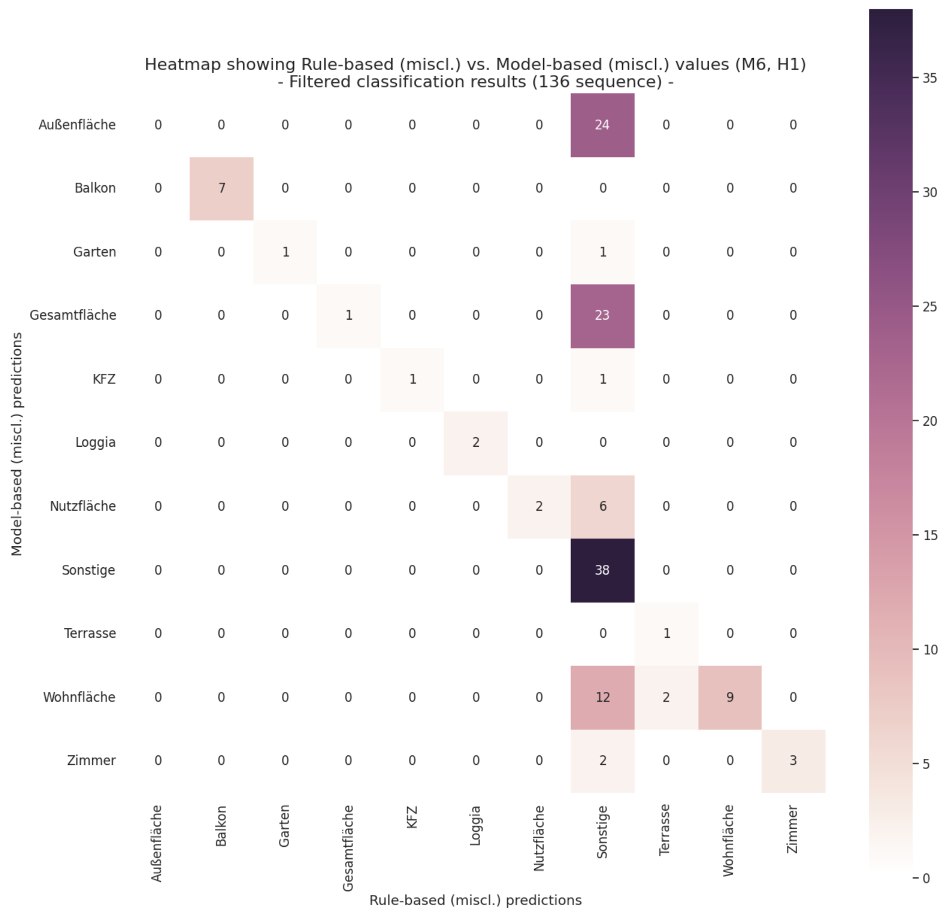


Figure 37: Heatmap of fine-grained results comparing rule-based and model-based (M_6, H_1) values. The values were filtered so the figure only displays the classification results, where both methods misclassified the sequences.

The 298 out of 668 sequences, where both methods (rule-based versus model-based using the model hypothesis combination M_1, H_0) classified them correctly, are displayed in Figure 38 along the diagonal. One of the largest classes, with most sequences being assigned to in the manual ground truth, was *Wohnfläche* (engl. living area), which occurred more than 100 times within the 668 sequences (see Section 4.8). Interestingly, this class exhibited the highest concordance in this heatmap. Additionally, *plot area*, which was also one of the most prominent classes in the manual ground truth, shows only 2 correct classification by both methods.

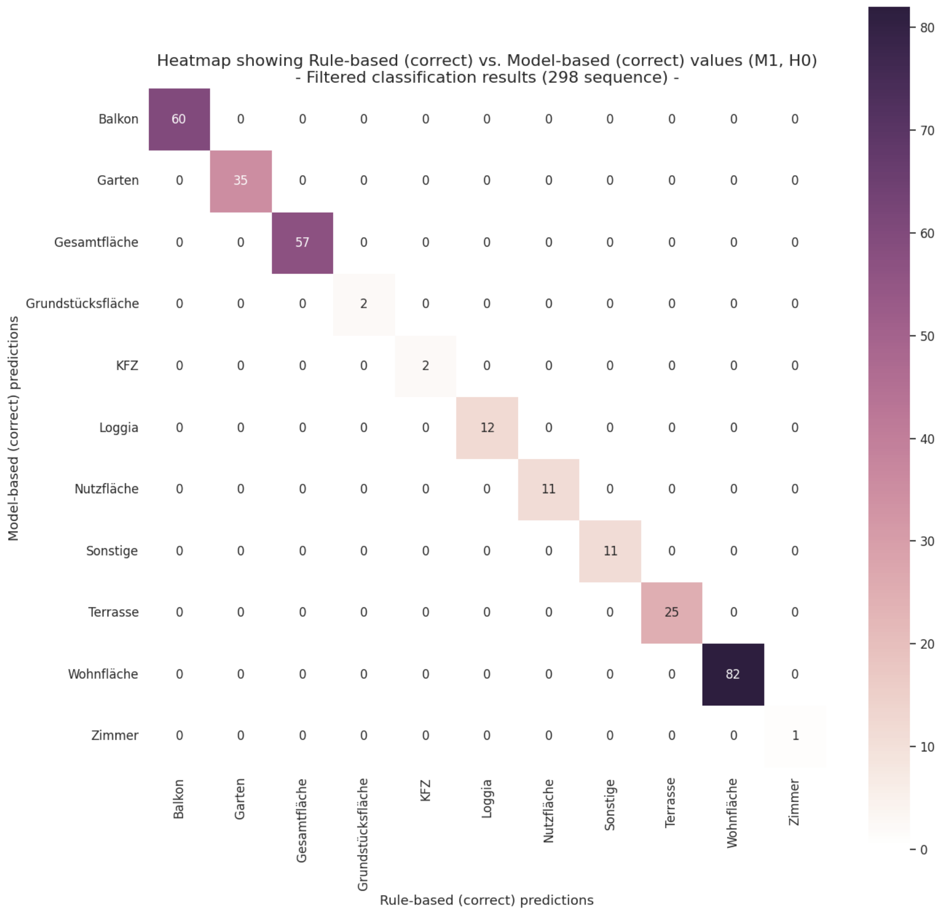
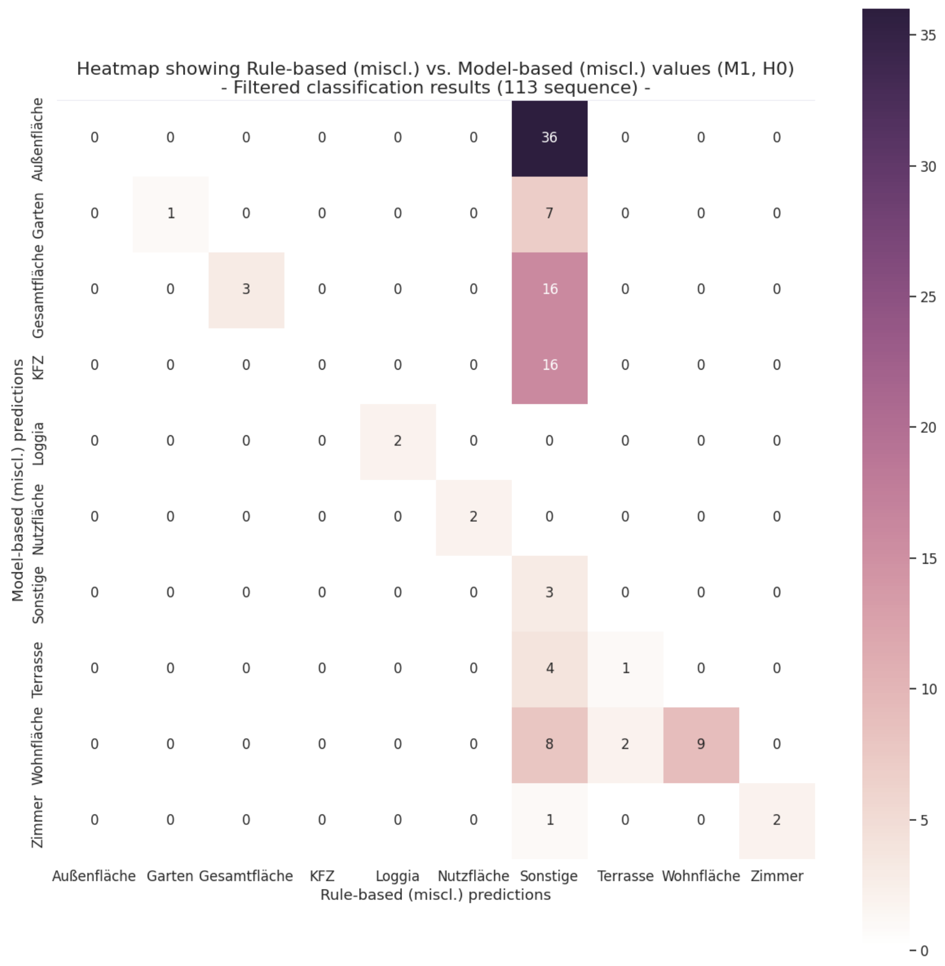


Figure 38: Heatmap of fine-grained results comparing rule-based and model-based (M_1, H_0) values. The values were filtered so the figure only displays the classification results, where both methods classified the sequences correctly.

The 113 out of 668 sequences, where both methods (rule-based versus model-based using the model hypothesis combination M_1, H_0) misclassified them, are displayed in Figure 39. Due to the rule-based approach having an else-condition favoring the class *other*, there are many misclassifications across that respective column in the heatmap.

Another interesting observation is, that there were no sequences of class *balcony* misclassified by both methods.

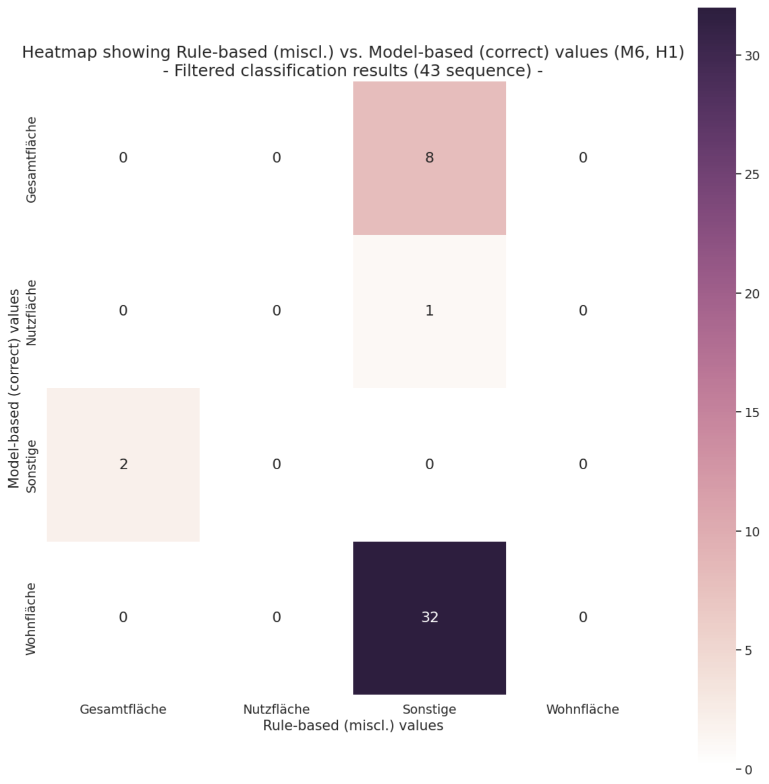


	Außenfläche	Garten	Gesamtfläche	KFZ	Loggia	Nutzfläche	Sonstige	Terrasse	Wohnfläche	Zimmer
Außenfläche	0	0	0	0	0	0	36	0	0	0
Garten	0	1	0	0	0	0	7	0	0	0
Gesamtfläche	0	0	3	0	0	0	16	0	0	0
KFZ	0	0	0	0	0	0	16	0	0	0
Loggia	0	0	0	0	2	0	0	0	0	0
Nutzfläche	0	0	0	0	0	2	0	0	0	0
Sonstige	0	0	0	0	0	0	3	0	0	0
Terrasse	0	0	0	0	0	0	4	1	0	0
Wohnfläche	0	0	0	0	0	0	8	2	9	0
Zimmer	0	0	0	0	0	0	1	0	0	2

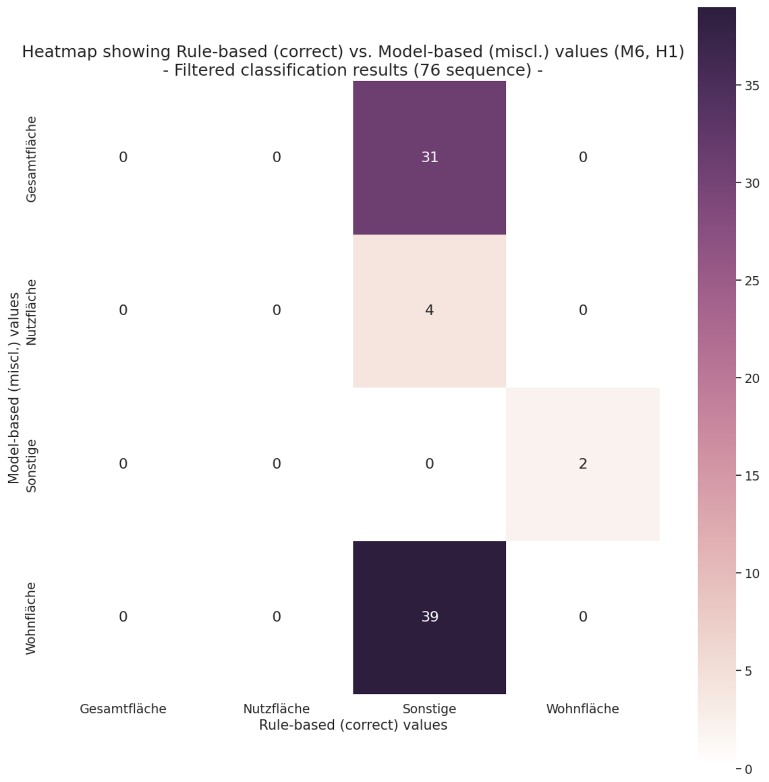
Figure 39: Heatmap of fine-grained results comparing rule-based and model-based (M_1, H_0) values. The values were filtered so the figure only displays the classification results, where both methods misclassified the sequences.

Heatmaps of Aggregated Labels Filtered

Below are two heatmaps of zero-shot models (first M_6, H_1, second M_1, H_0) in comparison to the rule-based results, where either the model results are correct or the rule-based ones. The data was filtered for one correct and the other having incorrect classifications results.

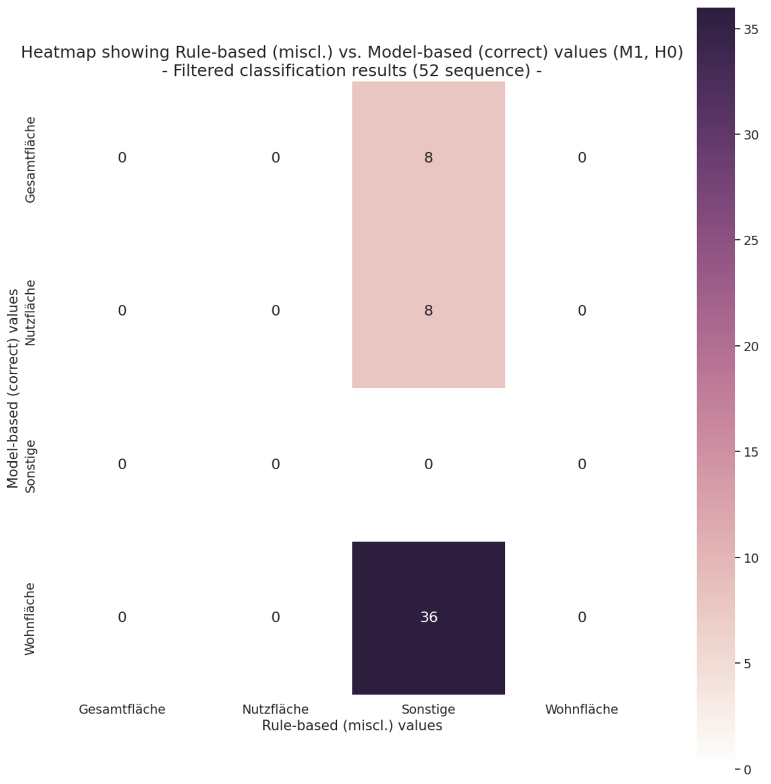


(a) Heatmap model M_6, H_1 in comparison to the rule-based results. Filtered on cases with *correct* model and *wrong* rule-based classification.

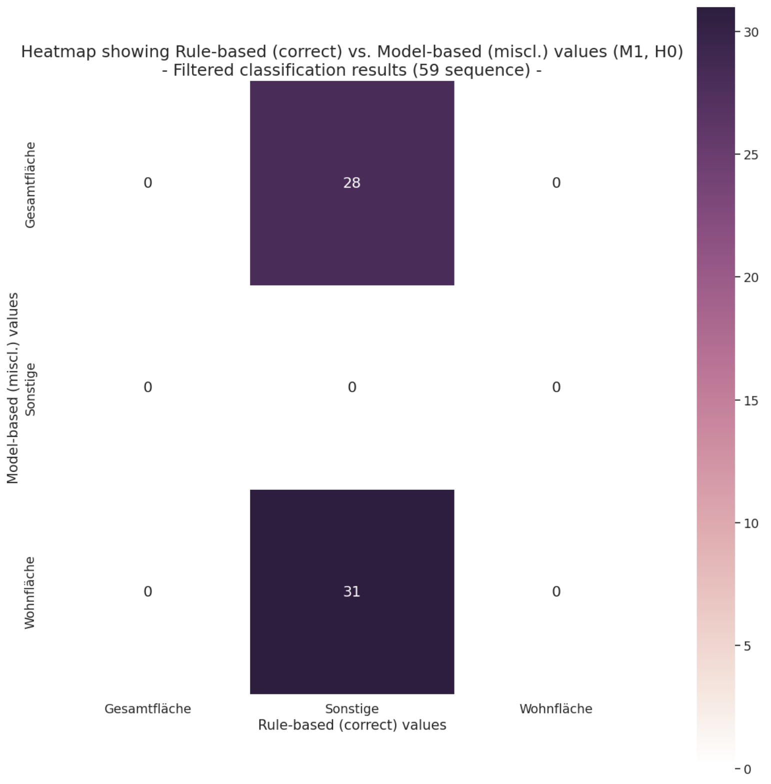


(b) Heatmap model M_6, H_1 in comparison to the rule-based results. Filtered on cases with *wrong* model and *correct* rule-based classification.

Figure 40: Heatmaps of aggregated labels, classification results filtered on either model or rule-based correct (M_6, H_1).



(a) Heatmap model M_1, H_0 in comparison to the rule-based results. Filtered on cases with *correct* model and *wrong* rule-based classification.



(b) Heatmap model M_1, H_0 in comparison to the rule-based results. Filtered on cases with *wrong* model and *correct* rule-based classification.

Figure 41: Heatmaps of aggregated labels, classification results filtered on either model or rule-based correct $(M_1,\ H_0)$.