Erasmus University Rotterdam

Classifying Dutch Fiscal Case-Law Articles using Natural Language Processing

Kenneth de Ronde
456332

A Master Thesis presented for the degree of
Data Science and Marketing Analytics

Erasmus School of Economics
Supervisor: M. van de Velden
Second assessor: S.L. Malek
Company supervisor: L.M. Kranenburg
November 30, 2021

Acknowledgements

My sincere thanks go to the founders of Bluetick for the opportunity, trust and freedom given to me during this research and the great time I had during and after work. In particular, I would like to thank Thijs Kranenburg for the interesting discussions and his availability for a variety of questions.

In addition, I would like to thank Michel van de Velden for the critical but constructive feedback during the writing of the thesis. Our discussions kept me going in the right direction and helped me keep my focus. I have enjoyed the process, for which I would also like to thank Stacey Malek.

Special thanks should be given to Rosanne, Stefan, Mike and my parents. They helped me to maintain a healthy balance during the writing of the thesis and to keep my priorities clear.

Abstract

The goal of this research is to classify Dutch fiscal case-law articles into sub-areas of taxation using Natural Language Processing (i.e., analysing text). To achieve this a method for document classification consisting of two steps is proposed. In the first step, a part of the data is labeled through a straightforward approach of looking only at specific components of a document. In the second step a Transformer-based model, which is a type of a neural network specifically designed for Natural Language Processing, is applied. The Transformer-based models train on a labeled subset of the data generated by the first step to be able to classify unseen documents. The method that achieves the best result is to create a training dataset by searching for specific contextual keywords corresponding with each sub-area of taxation and to train the newest Dutch Transformer-based model (named 'RobBERT') on it. This achieves 87% accuracy on the classification of the test set, which is more than the unsupervised topic modelling method of Latent Dirichlet Allocation (that only achieves 20%). The proposed method can easily be applied in other domains (if a part of the data can be labeled correctly using a straightforward approach) and requires no manual labeling efforts.

Keywords – Transformer, Document Classification, Long-Document Transformer, Natural Language Processing (NLP)

Contents

1 Introduction

Artificial intelligence is used in an increasing number of fields. Various activities which used to be done manually are now more attractive for computers to handle efficiently. Machine learning is a part of artificial intelligence. In the field of machine learning people generally instruct and train algorithms to analyse characteristics of the data in order to let them make a certain prediction based on the information that was given. A rising topic of research within artificial intelligence is processing human language (i.e., text or speech) to retrieve specific information and produce output such as a translation or a summary. This field of machine learning is called Natural Language Processing (NLP). The development of NLP has been rapid in recent years in areas like machine translation (such as Google translate), information retrieval (such as sentiment analysis for product reviews), recommender systems (such as LinkedIn recommending suitable job vacancies), question answering and speech recognition (such as Apple's Siri and Amazon's Alexa) (Chowdhury, 2003; Smeaton, 1999). This research focuses on the NLP-area of document classification. With the increasing possibilities to process and analyse enormous amounts of textual data the efficiency of NLP-tasks has increased (Deng and Liu, 2018). It is therefore interesting to analyse how relatively new NLP-methods perform on a document classification task.

Human language is complicated because it is diverse and full of ambiguities. For humans it is often manageable to understand when someone uses sarcasm or humour (although people do make mistakes in this area) because it can be derived from context. However, in the example of processing a text a computer could face difficulties identifying this kind of advanced communication. An algorithm can process a text by analysing and counting (the frequency of) words in a text and possibly the direct meaning of a word via word embeddings. A word embedding is a numerical representation of a word, where similar words have similar numerical representations. Another example of why the human language is difficult to process for an algorithm is the usage of one word that has multiple meanings (homonym). The word 'book' can mean something to read or the act

of making a reservation for an event. Humans use the context of a sentence to determine what is meant, but for an algorithm this can be problematic. These characteristics of the human language can be, amongst others, a problem for the processing of textual data by algorithms.

The goal of this research is to classify Dutch fiscal jurisprudence (hereafter: case-law) articles to the specific type of taxation that the article is about. A case-law article is a document in which a judicial decision on a certain case is stated. So, Dutch fiscal case-law articles are all judicial decisions in the field of tax law in The Netherlands. The Dutch tax legislation consists of multiple sub-areas in terms of taxation types such as income tax, corporate tax, sales tax etc. The company Bluetick has developed a data-driven search-engine for the search to case-law articles and aims to simplify the search for relevant case-law articles by providing recommendations in terms of similar articles to a customer's query. This research can contribute to the improvement of Bluetick's search-engine and thereby strengthening their market position. In general, this research can also contribute to cases where a dataset containing (legal) documents is entirely unlabeled and needs to be labeled. For example, if someone has an unlabeled dataset with news articles the same research methods can be applied to provide labels in terms of specific sub-areas of news (i.e., sports, financial or politics). The only condition to apply the method is that a part of the documents can be labeled using a simple strategy (e.g., by looking at specific keywords). Before the completion of this research Bluetick only provided filters within the search-engine that could select results from certain legal areas (such as administrative law, tax law and civil law) or legal authorities (such as the Supreme Court, Council of State, Courts of Justice and lower courts). With this research the fiscal case-law articles can be further classified into sub-area of taxation (such as income tax, corporate tax and sales tax), which gives Bluetick the possibility to include an extra filter in their search-engine. If the applied methods in this research are successful, it also ensures that people no longer need to manually label newly published fiscal case-law articles. This can be a competitive advantage because of lower labour costs, possibly in a wider range of markets than just the legal market.

There are two main contributions to the existing academic literature and one main contribution for managerial purposes. The first contribution to the literature is the overall structure of this research. The Dutch fiscal jurisprudence derived from www.rechtspraak.nl is entirely unlabeled in terms of type of taxation. So, users can only figure out to what type of tax an article is about by reading it themselves. The only label the tax articles in the dataset contain is a general label that indicates that it concerns a tax law article. Still, one needs the presence of correct labels to be able to train a supervised machine learning model and evaluate the performance accordingly.

The first step in this research is to relatively straightforward provide labels for (a part of) the fiscal case-law articles in the dataset. For this the following three approaches are considered:

1. Labels provided by an unsupervised method that directly classifies all articles (described in Section 2.1.1).

2. Labels provided by looking at references to legal articles (described in Section 2.1.2);

3. Labels provided by looking at specific keywords which are frequently mentioned within a sub-area of taxation (described in Section 2.1.3);

The first approach is an unsupervised topic modelling method and classifies all case-law articles in the dataset. It is called Latent Dirichlet Allocation. The second and third method provide a type of taxation label for a part of the case-law articles in the dataset, since not all articles contain a word or phrase which is processed by these methods.

The second step in this research is to train supervised machine learning models on the labels provided by the methods mentioned above to classify the rest of the dataset. The two supervised models that will be considered are types of a newer neural network structure used specifically for Natural Language Processing: The Transformer (which will be explained in detail later). The performance of the methods is determined by comparing it with a dataset containing fiscal case-law articles with manually annotated labels. The first contribution to the literature lies in the overall structure of the research: going from

an entirely unlabeled dataset to a partially labeled dataset, using this as training data and subsequently labeling the rest of the data with a new supervised machine learning method for Natural Language Processing called The Transformer. In this research the goal is to find the best performing combination of the two steps described above and compare it with the unsupervised topic modelling method. This knowledge can then be used to classify documents without labels from other application areas as well.

The second contribution to the literature is the comparison of relatively advanced supervised Natural Language Processing methods on a Dutch (legal) document classification task. The rapid rate of progress and diversity of techniques in the NLP-field, combined with the wide range of application areas within the field (such as translation, information retrieval, recommender systems etc.) can make it difficult to estimate and compare performance of models for specific tasks in various languages. A model can perform well on one task, but poorly on others. Recent years have been full of introductions of new models. It is important to check how these models perform compared to each other and to older models using this particular task of classifying Dutch fiscal case-law articles into types of taxation. There are leaderboards[1] available that compare different models on several English NLP-tasks. However, the performance of models on relatively complex and long documents (in a language other than English) remains to be examined.

The third contribution is more from a business and managerial perspective. The application of NLP-models on the task of classifying Dutch case-law articles into sub-area of taxation is a contribution to the practical business knowledge. This research demonstrates the results for a specific part of text classification: the fiscal (case-law) field. If the applied methods in this research are accurate, one does not have to label the articles manually into the type of tax but can run the preferred model to classify the fiscal jurisprudence, which saves time and money. Big Dutch publishers of legal content like Wolters Kluwer and SDU usually hire multiple working students to manually label newly published legal content. This content is not openly available and can only be seen

[1]See: (Super)Glue Leaderboard - https://gluebenchmark.com/leaderboard.

by paid users. If the models applied in this research are successful the working students become mostly redundant for this specific task. The research is also useful to newer search engines for legal documents in order to include an extra filter within the engine that filters on sub-areas of taxation (or add any other desired sub-categories).

The task of providing labels for Dutch fiscal case-law articles in terms of sub-area of taxation is twofold. On one side, it starts with an entirely unlabeled set of fiscal case-law articles that (partially) needs labels in order to be the input for a subsequent model. This is the first step of the research. The idea is to label a part of the dataset by looking at certain characteristics of the articles such as specific words or references to legal articles. Not all articles contain these specific items, so only a part of the data can be labeled by this step. The accuracy of the first step in this research determines the overall accuracy to a large extent, because if the specified labels are incorrect the supervised model trains on incorrect labels. Next, a more advanced supervised model is used to label the rest of the case-law articles. This is the second step of the research. The labels derived from the first part of the research serve as the input for the subsequent model to train on. The models compared in this second step are two Transformer-based models (see Section 2.2.8) which are currently the state-of-the art (i.e., the current highest level of development) within the NLP-field. The preceding twofold problem leads to the following research question:

– *Which combination of methods is most successful in classifying Dutch fiscal case-law articles into type of taxation?*

So, the word "combination" in this research question implies the linking of a method that provides labels for a part of the dataset in order to serve as a training set for a supervised model (described above as the first step of this research). A supervised classification model labels the rest of the dataset based on the information from the training set (described above as the second step of this research). An overview of the combinations of methods that are compared in this research is shown in Table 1, where the BERT and RoBERTa models are Transformer-based models (explained in Section 2.2.8 to 2.2.10).

Finally the performance of the combinations of steps are evaluated by comparing the results with a manually annotated dataset of fiscal case-law articles.

Table 1: Combinations of research methods.

One-step method	
Latent Dirichlet Allocation (Clustering)	
Two-step methods	
Train labeling	**Full labeling**
Searching for references to legal articles	BERT
Searching for references to legal articles	RoBERTa
Searching for specific contextual keywords	BERT
Searching for specific contextual keywords	RoBERTa

This research is structured as follows. In Section 2, literature about Natural Language Processing in the context of this research is discussed. Section 3 describes the data used in this research with additional exploratory analysis. In Section 4 the methodology for both providing labels based on certain characteristics of the text as well as subsequent models (shown in Table 1) is described. The Latent Dirichlet Allocation method is also explained in this section. In Section 5, the results are presented to answer the research question and in Section 6 a discussion with conclusions and recommendations is given.

2 Literature Review

In this section an overview of the relevant literature for this research is given. In the first part of the literature review, clustering methods and unsupervised methods that can label data based on certain characteristics of a case-law article are described. This is necessary because the starting point of this research is an entirely unlabeled dataset of Dutch fiscal case-law articles in terms of sub-area of taxation. In the second part of the literature review, important components of Natural Language Processing methods in the context of this research are briefly discussed combined with the current state-of-the art (i.e., the highest level of development) methods that can be used to classify fiscal case-law articles into the specific sub-area of taxation. These are supervised methods that use a training set provided by the methods from the first part of the literature review. A schematic overview of the research structure is given in Figure 1 below. In Section 5, combinations of the methods from the first and the second part of the literature review are compared in their performance to successfully classify fiscal case-law articles.

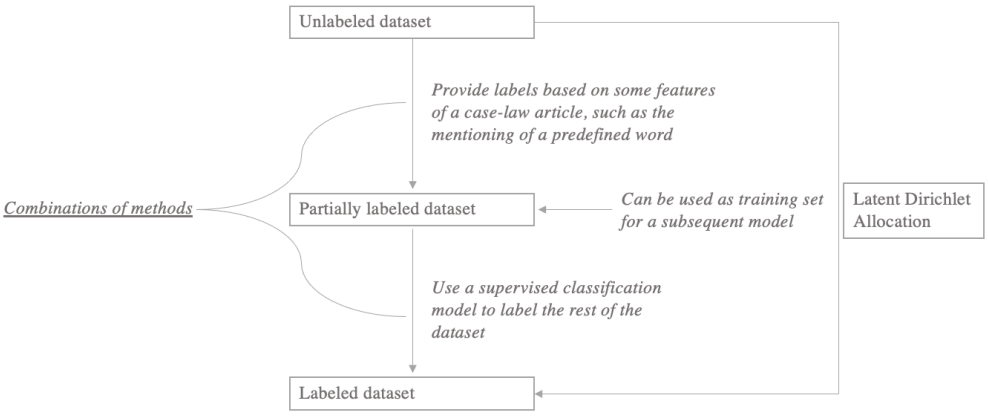

Figure 1: Research design.

2.1 Absence of labels in the dataset

Classification algorithms are useful when observations need to be labeled (i.e., classified) into a fixed number of classes. Usually, a model trains on labeled data and uses the

knowledge accumulated during this process for classifying future observations into pre-specified categories. A problem arises when a training dataset contains incomplete, noisy or completely unavailable labels (Iscen et al., 2018, p. 7643). In this case, a model learns incorrect associations from the training set or can only derive conclusions about the data through unsupervised methods (in the event of total absence of labels). In this research's dataset no labels for types of taxation are given for all case-law articles. The absence of labels makes it impossible to directly apply a supervised classification method. A frequently used solution to this problem is to use an unsupervised method like clustering or, in the case of textual data, a topic modelling method like Latent Dirichlet Allocation to classify the data (MacQueen et al., 1967; Kanungo et al., 2002). Another solution for textual data (e.g., text documents) could be to generate labels for a part of the data based on some specific characteristics of a document. The labeled part of the data can then be used as a training dataset for a subsequent supervised classification model as shown in Figure 1. The possible unsupervised methods for the classification task in this research are explained in the following sections.

2.1.1 Clustering

Like discussed in the previous paragraph, a possible solution for classifying observations into categories without the presence of labels is by applying clustering. Clustering algorithms classify each observation into a group based on (dis)similarities in the data. Intuitively, data observations within the same group should have similar characteristics and observations from different groups should have dissimilar characteristics. In the case of textual data (in the form of case-law articles) a variable is the occurrence of a specific word. So, the clustering algorithm classifies a document to a specific group based on the occurrence of words that also occur in other documents in that group.

Two possible problems for clustering algorithms in the context of this research arise. Firstly, for text classification there are usually many variables since a variable is the occurrence of a word. Clustering algorithms face difficulties when dealing with high-dimensional data because similarity measures become computationally more complex

with the number of dimensions (Steinbach et al., 2004, pp. 12-13). This is because the distance between two observations (in this research: case-law articles) is calculated by the difference between all variables for those observations. Similarly, a cluster center is calculated as the average of all variables for the data observations belonging to that cluster. So, when the dimensionality of the data is high the calculations of the distances between observations as well as the cluster centers become larger and thus computational complexity increases. Secondly, clustering algorithms face difficulties in this context because many variables are equal to zero for most observations. This is called sparse data. Specifically, the number of variables is equal to the number of different words in the corpus (i.e., in all documents in the data), which can become a large number. This means that many variables equal zero since a case-law article only contains a small proportion of words used in the full dataset. For calculating distances between observations this is useful, since calculations become easier when many variables equal zero. The reason why clustering algorithms experience difficulties in this context is because observations that have a zero value for the same variables are interpreted as similar and therefore classified to the same group. However, text documents that both do not contain a word (or multiple words) are not necessarily about the same topic. It is more useful to cluster documents based on words they both contain instead of clustering them on words they both do not contain.

To tackle the difficulties, the data can be transformed before applying clustering and/or a suitable clustering algorithm for text data can be selected. One can transform the data into a Term Frequency – Inversed Document Frequency matrix to better cover the relative frequency of a word within a case-law article (Bafna et al., 2016). This is the case because the matrix accounts for how many times a word occurs within a document, relative to how many times that word occurs in all documents in the dataset. Still, a zero-value for a variable remains the same in the Term Frequence - Inversed Document Frequency matrix when a word does not occur within a document. A dimensionality reducing technique can be used such as Principal Component Analysis (PCA). This decreases the number of dimensions of the data by transforming a large set of variables into

a smaller set while maintaining as much information from the original data as possible. More information about transforming the data can be found in the next paragraph. Another way to cluster text documents is by applying a spherical k-means algorithm instead of a classic k-means algorithm, which looks at the cosine similarity (i.e., the angle) between vectors (Buchta et al., 2012). In this research a vector would be a case-law article (Buchta et al., 2012, pp. 2-3). Empirical evidence shows that various types of spherical k-means perform better than classic k-means for clustering text documents (Buchta et al., 2012, p. 19). The difference lies in the fact that k-means clustering looks at the distance between two observations by using a distance function for all variables, where spherical k-means uses a different type of distance measure. Particularly, spherical k-means looks at the difference between the angles of vectors. Vectors with similar angles are subsequently clustered. Spherical k-means is frequently used for clustering based on the occurrence of words in a document (Buchta et al., 2012, p. 19).

An important point to clarify is the representation of the case-law articles before applying clustering like earlier mentioned through the PCA technique. The starting point for text classification usually is a document-term matrix with the documents in rows and the variables (i.e., words) in columns. This data structure results in high dimensionality as described earlier. Transforming the data into a smaller number of dimensions results in less required computational power and it can also result in a simpler retrieval of important information from the data (Bengio et al., 2013, pp. 3-4). The transformation of data before applying clustering in order to make it more suitable for a cluster algorithm is called representation learning in the paper from Bengio et al. (2013). In representation learning a model extracts information from data objects and uses this information to create a new representation of an object which makes it easier to apply clustering (Bengio et al., 2013, p. 1). Examples of this are the previously mentioned PCA technique for document representations and word embeddings for word representations (explained in Section 2.2.3). Another example of representation learning (for image clustering) is provided by Tao et al. (2021), who develop an approach that clusters images by using extractions of certain variables of the data and thereby capturing similarity. The

clustering is for example based on images that contain the same colours in the same places. It is interesting to see if it is also possible to create useful representations containing important extracted information of case-law articles before applying clustering. Tao et al. (2021, p. 9) expect that this could also be effective because extracting important variables (i.e., words or word combinations) from texts is easier than extracting relevant variables from raw pictures.

A frequently used method for the representation of text documents is Latent Dirichlet Allocation (Blei et al., 2003). This is an unsupervised topic modelling method that uses Dirichlet distributions to build a topics per document model and a words per topic model. The idea behind this method is that a document is about a number of topics and those topics are all covered by a number of words. Given the fact that this method is designed specifically for the unsupervised classification of text documents, it is useful for this research. In Section 2.2.2 a more detailed description of Latent Dirichlet Allocation is given.

2.1.2 Extracting labels from characteristics

A second option to move away from an entirely unlabeled dataset is to provide labels for a part of the case-law articles based on certain characteristics of a document. Named Entity Recognition can extract information from a document by detecting names of organisations, persons, locations and more. An example which is applicable for this research is to use a Named Entity Recognition model which labels a case-law article that contains references to certain legal articles to a sub-area of taxation (e.g., 'article 1 of the Income tax chapter'). So, the entities where the Named Entity Recognition model searches for are references to (fiscal) legal articles. Obviously, only a part of the dataset is labeled by this solution because not all case-law articles contain a reference to a legal article. The part of case-law articles that is labeled by this method can form a training dataset for a subsequent supervised model that labels the rest of the articles.

When running the specific Named Entity Recognition model (or any other model which generates labels based on specific parts of the data), certain entities (i.e., references

to legal articles) indicate that an article belongs to one sub-area of taxation. In this way labels are automatically generated for some case-law articles: these are called pseudo-labels. Pseudo-labels are different than human-annotated labels because they are only based on specific properties of a document, while human-annotated labels are carefully issued based on the whole context (Jing and Tian, 2020, pp. 2-3). This means that pseudo-labels can in principle be wrong because they are focused only on one or a few aspects of a document. Similarly, a case-law article may contain a reference to a legal article from another sub-area of taxation. This can possibly result in an incorrect label. Still, it is likely that a reference to a legal article within a case-law article points to the correctly corresponding sub-area of taxation.

The process of automatically providing pseudo-labels based on specific characteristics of the data is an example of a self-supervised learning process. Self-supervised learning refers to methods trained with automatically generated labels (i.e., without a human providing labeled data) (Jing and Tian, 2020, p. 2). The process usually consists of two parts. First, a predefined text task (like Named Entity Recognition) is solved resulting in the generation of pseudo-labels by the model based on some specific characteristics of the data. After this, the pseudo-labeled dataset is forwarded to a subsequent model where it serves as training set (Jing and Tian, 2020, p. 2). The subsequent model then labels the rest of the data. It is common to compare this to human-annotated labels (for a subset of the data) to estimate the performance. The performance of self-supervised learning methods is highly influenced by the accuracy of the generated pseudo-labels on which the model trains. Self-supervised learning has already been successfully introduced to the NLP field, for example by Mikolov et al. (2013b) to predict context words within a window and by Devlin et al. (2018) in the training process of the Bidirectional Encoder Representations from Transformers (BERT). Both of these models are explained later in Sections 2.2.3 and 2.2.9.

A Named Entity Recognition model that searches for references to legal articles in documents is called the Extendable Legal Link Extractor (van Opijnen et al., 2015). This Dutch framework is able to detect references for both national and European

legislation (van Opijnen et al., 2015, p. 8). It uses Named Entity Recognition and Pattern Recognition to detect references to legislation within documents. The Extractor can generate pseudo-labels by searching for references in case-law articles to Dutch fiscal legislation and automatically label the document as the specific type of taxation. For example, if the Legal Link Extractor finds a reference to a tax law which is covered by the income tax chapter it labels the case-law article as income tax. Like earlier mentioned, only a part of the case-law articles will be labeled by the Legal Link Extractor because not all articles contain a reference to a specific tax law. In Figure 2 this approach is visually illustrated.

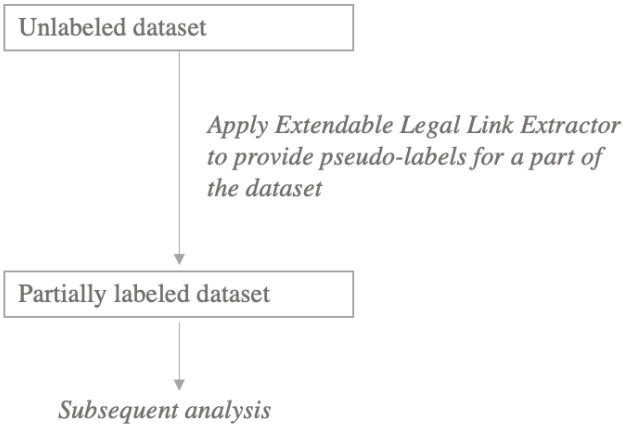

Figure 2: Assigning labels for a part of the data.

2.1.3 Extracting labels from the occurrence of specific contextual words

A third option for providing labels for an unlabeled dataset is to look directly at keywords that occur in a case-law article. A similar example for this is provided by Turney (2002), who directly looks at the sentimental orientation of words within a text. To classify reviews to particular sentiments an algorithm is used that searches for synonyms for positive and negative words (Turney, 2001). The algorithm identifies and recognises synonyms for positive and negative words (because this information is provided beforehand) and calculates the corresponding sentimental value (which is also

provided beforehand) (Turney, 2001, p. 491). The score of a review is then defined by comparing its similarity to a positive word (like 'nice') with its similarity to a negative word (like 'poor') (Turney, 2002, p. 2). The goal of this research was to analyse sentiment and classify reviews into two categories (positive/negative), but the general idea of this research design can be extended to this research's task of classifying case-law articles. It can be interesting to look for specific contextual keywords in the data and label (a part of the) data on that basis. For example, one would expect that in fiscal case-law articles about income tax the Dutch word "inkomstenbelasting" (income tax) is frequently mentioned. Other words that have a strong link with a specific type of taxation can be added to a list of contextual keywords for each tax type. In this way pseudo-labels can be assigned to a part of the data and this part can serve as a training dataset for a subsequent model, similar to the Extendable Legal Link Extractor from Figure 2.

2.1.4 Discussion

A supervised classification model needs labels assigned to a part of the data to train on that and subsequently make predictions on unseen data. In the previous sections three possible solutions are discussed for labeling an unlabeled dataset. Clustering (or: unsupervised topic modelling by Latent Dirichlet Allocation), extracting labels based on references to legal articles and extracting labels based on the occurrence of contextual keywords are three applicable methods. Where types of self-supervised learning look at a few specific characteristics to generate pseudo-labels for only a part of the data, clustering aims to classify every data observation into a group based on (dis)similarity. Before applying a clustering algorithm the data can be transformed to reduce the number of dimensions and to make the data more compatible for clustering. In addition, clustering is fully unsupervised which may cause the performance to be less accurate because a model was never trained on labeled data. A suitable clustering method to apply is Latent Dirichlet Allocation because it is especially designed for unsupervised topic modelling of documents.

Looking at references to legal articles (using the Extendable Legal Link Extractor)

and looking at contextual keywords both generate pseudo-labels which do not necessarily have to be correct because they are only based on a few specific aspects of a case-law article. In this research these approaches are compared. Intuitively one would expect that a reference to a legal article within a case-law article points to the correct sub-area of taxation where the article is about. Looking at contextual keywords can also result in many correct labels, but keywords do not always give the certainty that a case-law article is really about the specific fiscal subject. Clustering and Latent Dirichlet Allocation label all case-law articles in the data, so labels provided by these methods can be directly compared with a subset of the data which contains manually assigned labels. Searching for references to legal articles or searching for contextual keywords results in labels for only part of the dataset. A supervised classification model is subsequently needed to label the rest of the data and to compare it with the manually labeled subset to evaluate performance. The supervised models will be explained in the following sections.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a part of machine learning that concerns the processing of human language by computers. The specific goal of NLP is to process and analyse human language (or: natural language) to perform tasks such as translation or information retrieval. This area of artificial intelligence originates from the second part of the 80s with the arrival of machine learning algorithms and extended computational power (Foote, 2019). Initially, machine translation was the most important area of NLP research and is in fact the origin of this field (Chowdhury, 2003). However, more areas have been added to the field of NLP. The speed of development of new NLP-techniques has been increasing in recent years with consistently producing new state-of-the art (i.e., the highest level of development at a specific time) methods.[2] In this section, a brief historical overview of several methods within the field of NLP is given with focus on the most recent period. First, some methods will be explained that are the foundation of

[2]See: (Super)Glue Leaderboard - https://gluebenchmark.com/leaderboard for the most recent state-of-the art methods.

currently used models. After that these modern models are clarified. In the end, some models are selected to compare in performance of classifying fiscal case-law articles into type of taxation. These models are trained on the data gathered with the methods from Section 2.1.

2.2.1 The early stage of Natural Language Processing

Before and during the first half of the 80s most computer systems used 'if-then' models to make decisions (Foote, 2019). These systems used specific rules when a word occurred. In the example of machine translation from English to German a system would just use two lists of words: when a word from the English list was given as input it looked at the same row in the German words list and give this as output. A problem arose when an input word was not present in the list or when a word had multiple meanings (homonym). In the latter case the system would just give all possible meanings as output. It was not yet able to account for the context of the word. These if-then rules provide a specific and clear framework that can help with some tasks but are generally limited. In the paper by Moore (1981), the problem is clearly described as *"Unrestricted dialogue remains as elusive as ever. Even if we know how to provide a computer with all the knowledge it needed in a form it could use (which we do not), the task of putting it all in would be so great as to be totally impractical."* (Moore, 1981, p. 5). From this quote, both the lack of knowledge of appropriate methods as well as the lack of computational power clearly emerges. Automated machine translation for long pieces of text and information retrieval systems were impractical before the late 1980s while if-then systems were limited and full of weaknesses (Moore, 1981, p. 6).

In the 90s, there was a shift from previously discussed limited if-then approaches to methods taking probabilities into account such as decision trees (Bahl et al., 1989), statistical (probabilistic) language models that used N-grams to model semi-long range correlations (Lafferty et al., 1992; Della Pietra et al., 1994; Berger et al., 1994) and other approaches that took conditional probabilities (i.e., the probability of the occurrence of a word given the occurrence of another word) into account (Berger et al., 1996). Other

methods in the field of Information Retrieval (Lewis and Jones, 1996; Smeaton, 1999; Deerwester et al., 1990) show a slow but increasing performance compared to earlier techniques from the 80s.

2.2.2 Topic modelling

As from 2000, topic modelling and topic detection became a popular field of research within NLP. Topic modelling is an unsupervised technique where a topic is a collection of words that often occur together within documents (Steyvers et al., 2004). In this way different topics can be linked to a document with a corresponding percentage for each topic. So, topic modelling identifies topics within a text and assigns topics for each document, where a topic is a collection of words that often appear jointly (Steyvers et al., 2004, p. 308). One needs to interpret a topic manually by looking at the corresponding collection of words. For example, when a topic contains the words "football", "baseball", "basketball" and "game" one can define this topic as "sports". Before applying topic modelling it is important to remove stopwords (i.e., frequently used words with little unique information like "and", "the" and "I") from the text, because otherwise these words belong to topics as well.

The goal of topic modelling is to extract information from a body of text using unsupervised learning techniques (Steyvers et al., 2004). The introduction of Latent Dirichlet Allocation extended the possibility to extract topics from documents by looking at (the frequency of) words in the documents (Blei et al., 2003, p. 1007). In Latent Dirichlet Allocation, a document is represented by a probability distribution of topics and these topics are represented by a probability distribution of words (Steyvers et al., 2004, p. 308). Latent Dirichlet Allocation uses one-hot encoding for word representation. This means that a word is represented by a vector of the length of the total different words in the corpus (i.e., the complete dataset containing all documents), with a 1 corresponding to the relevant word and zeros in all other places. Therefore, the representation of a word is a vector of the size of the full corpus (i.e., all different words in the complete dataset of documents). Latent Dirichlet Allocation also uses the bag-of-words assumption, which

implicitly assumes that the order of words in a document is not of relevance. So, the words within a document are not displayed in chronological order while processed by the model.

2.2.3 Word vectors

To solve the problem of increasing dimensionality with a bigger vocabulary and reduce the computational complexity by using another alternative for word representation than one-hot encoding, Mikolov et al. (2013a) propose two methods for computing continuous vector representations of words. These methods are part of the Word2Vec (i.e., word to vector) embedding technique which creates vectors for the representation of words to try to capture the meaning of words inside these vectors. The vector representations of words, named word embeddings, give initial meaning and similarity of and about words, usually in 50-100 dimensions (but this can be more) which is far less than one-hot-encoding for medium and large datasets. A vector for a word representation is created by looking at how frequently a word occurs in combination with other words. The general idea is that words that have similar context and appear in similar sentences have equivalent meanings (distributional hypothesis), and thus have similar word embeddings (Young et al., 2018, p. 2). A similar word embedding implies that two words have similar values in most dimensions (e.g., an approximately equal positive or negative value in the same dimension). Results show that the proposed model architectures significantly improve performance of NLP-tasks at much lower computational cost (Mikolov et al., 2013a, p. 10). Because of the lower computational complexity with respect to one-hot-encoding, the models can compute very accurate high dimensional word embeddings from much larger datasets. Within Natural Language Processing, more training data usually equals higher performance because a model can train on more sentences and see more language structures. In this research, word embeddings are used within the models to have a better representation of words. The next paragraph explains the two word embedding types from the paper of Mikolov et al. (2013a), who were the first to successfully introduce this new form of word representation in Natural Language Processing. An explanation is

given so that it is easier to understand the word representations which are used in this research.

In the paper by Mikolov et al. (2013a) two different model architectures for word embeddings are introduced. Both work with the conditional probability of the occurrence of a word given the occurrence of another word. Firstly, the Continuous Bag-of-Words model creates the embedding of words by predicting the current word (projection) based on its context (the words before and after). The embedding of the current word is created by averaging the embeddings of the context words (Mikolov et al., 2013a, p. 4). This is still a bag-of-words model, since the order of the context words does not influence the projection. Secondly, the Continuous Skip-gram model predicts context words based on the current word (Mikolov et al., 2013a, p. 4). So, both models are the opposite of each other because one predicts the current word based on multiple context words and the other one predicts multiple context words based on the current word. The range of context words (i.e., context window) that the model predicts or takes into account needs to be pre-specified, where higher context windows result in better quality word embeddings (because it can account for longer range dependencies) but also increases computational complexity. In Figure 3 both models are visually defined. The result (or output) of both models is a vector in case of the Continuous Bag-of-Words model or multiple vectors in case of the Continuous Skip-gram model (Mikolov et al., 2013a, p. 5). Word embeddings are defined in such a way that addition and subtraction of words is possible. For example, the result of the word embeddings of "Madrid" – "Spain" + "France" is very close to the word embedding of "Paris" and the word embeddings of "Germany" + "capital" is very close to the word embedding of "Berlin", while training is done without any supervised information (Mikolov et al., 2013b, p. 4).

Many other types of word embedding models are currently used within Natural Language Processing. Commonly used word embedding models are GloVe (Pennington et al., 2014), FastText (Joulin et al., 2016), Skip-Thoughts which focuses on sentences rather than words (Kiros et al., 2015) and Poincaré embeddings (Nickel and Kiela, 2017). Word embeddings are an appropriate and popular starting point for most NLP-tasks.

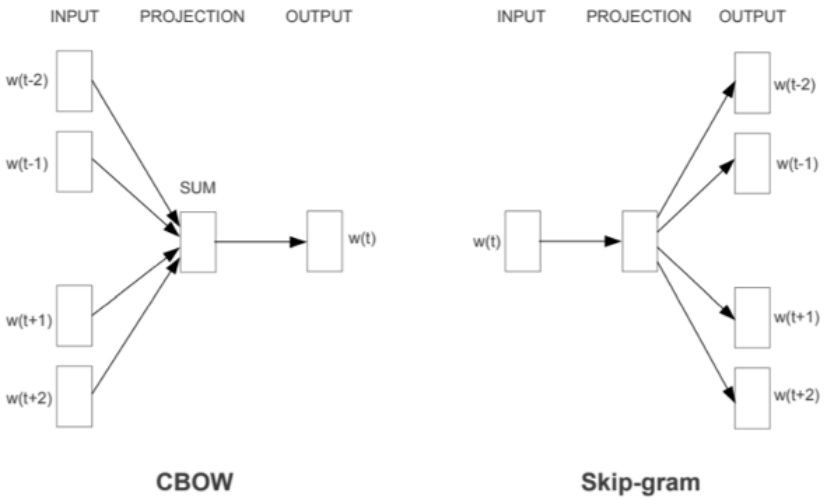

Figure 3: Continuous Bag-of-Words and Continuous Skip-gram model architecture for word embeddings (Mikolov et al., 2013a, p. 5).

Nowadays one can use pre-trained word embeddings that are trained on large corpora of text, so that one does not have to create word embeddings him or herself. These pre-trained embeddings are accurate word representations that are directly ready for use and are included in this research as well.

2.2.4 Deep learning

The methods covered so far process a piece of text under the bag-of-words assumption like explained in previous sections. Intuitively, this is not an ideal approach since meaning can be derived from the order of words. To provide an alternative for the bag-of-words assumption, deep learning methods are able to take other words into account while processing a particular word. For this, deep learning methods employ multiple layers to process textual data and are therefore able to analyse it sequentially (Young et al., 2018, p. 1). This means that the layers can store information from other parts of the sentence. The structure of multiple layers which are all connected to each other is the concept of deep learning. The specific methods used in deep learning are called neural networks. Neural networks contain multiple layers consisting of neurons that are all connected with

each other over the entire neural network (Hecht-Nielsen, 1992, p. 593). The neurons within a layer process data from inputs and pass it on to the next layer. This process is explained in detail in the following paragraph. Before 2006 training deep learning methods on textual data was mostly unsuccessful. However, when computational power increased, more training data became available and word embeddings were introduced, deep learning methods achieved state-of-the art (i.e., the highest level of development) performances in various NLP-tasks from 2010 onwards (Socher et al., 2012, p. 18). In the next sections two types of neural networks are briefly explained to demonstrate the method of using multiple layers to process text sequentially.

2.2.5 Recurrent neural networks

A recurrent neural network (RNN) is a neural network with feedback connections and has been applied for a long time in a wide number of areas such as wind turbine power estimation where the generated power of electric wind turbines is predicted (Li et al., 2001), face recognition (Lawrence et al., 1997) and financial predictions (Giles et al., 1997; Pradeepkumar and Ravi, 2017). In the field of Natural Language Processing, RNN models take one word at a time and process this word, while still being able to account for words that are previously seen. In a simple form, a RNN consists out of three layers: an input layer, a hidden (context) layer and an output layer which all contain multiple neurons (Mikolov et al., 2010, p. 6). Neurons are points in each layer of a neural network that store information and are connected with each other. Intuitively, it is helpful to see the neural network structure like a solution to a problem. The input layer represents the input of a particular problem, so it collects all information regarding it. The output layer provides the solution for the problem and the calculations that need to be done in order to achieve the solution take place in the hidden layer. In other words, the hidden layer transforms the data from the input layer into a form that can be used as solution in the output layer. This is also visually illustrated in Figure 4. The neurons within the layers all store and process a part of the information from the problem.

In a basic RNN in case of a text application, the input layer is formed by the word

embedding representing the current word in a sentence. This layer is taken into account together with the hidden layer from the previous words in a sentence (such that it can process previous words at the current time) and is then passed on to the hidden layer at the current moment. The hidden layer now has information about the current word and about previous words to pass on information to the output layer. The output can be a specific word (for example in translation tasks) or a binary outcome (for example in basic question answering tasks where the result is correct or false). A schematic overview of this process is given in Figure 4. In this figure there is one current hidden layer, but usually a neural network consists of multiple hidden layers.

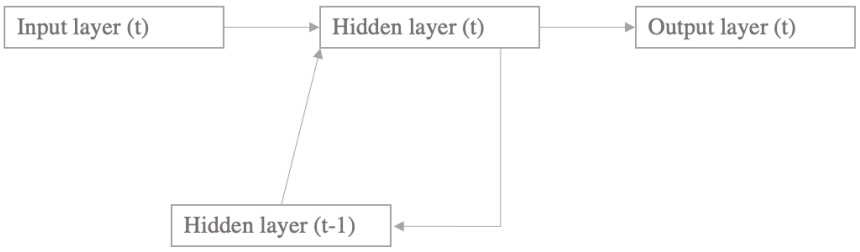

Figure 4: RNN simplified structure at time t.

A neural network needs to train on labeled data so that it can recognise associations in the data and make predictions about new observations. All neurons in the hidden layer of a neural network have a certain weight. This weight determines how data is transformed and passed on to other neurons. To define the weights of the neurons in the hidden layers, the neural network trains through backpropagation (Medsker and Jain, 1999, Ch. 1). Three major steps are executed during the training-phase of the model. Firstly, the RNN predicts the output by using an activation function (for example Tanh, ReLu or Sigmoid) within a hidden layer and by using a softmax function in the output layer that transforms values into probabilities that add up to one for all possible outcomes (Mikolov et al., 2010, p. 8). After that, a loss function is used to compare the prediction with the actual label for an observation. The outcome of the loss function represents the error made by the model. Finally, this error value is used to perform backpropagation which calculates the adjustments of the weights of the neurons in the

hidden layers (internal weights) (Medsker and Jain, 1999; Werbos, 1990). The directions and magnitudes of those adjustments to the weights of the neurons are called gradients: the larger this gradient, the bigger the error made by the model thus the bigger the adjustment needs to be in the neuron's weight. The goal of the training-phase is to minimise the error made by the model.

2.2.6 Vanishing gradient

Due to the structure of the backpropagation process recurrent neural networks experience trouble dealing with information from previous inputs of the data several steps ago (Pascanu et al., 2013, pp. 1310-1311). The correlation between information that was processed by the neural network several steps ago and the information processed at the current step is referred to as a long-term dependency in the data. Like visualised in Figure 4 a RNN is able to take into account long-term dependencies by concatenating the hidden layer of one step ago with the hidden layer of the current step. As these steps go further back a recurrent neural network experiences more difficulties taking the information into account. Training RNNs becomes more difficult and earlier experiments show that the parameters of a model (i.e., the weights of the neurons) result in sub-optimal values that only take short-term dependencies (i.e., less steps back) into account (Bengio et al., 1994, p. 157). Bengio et al. (1994) find that when the length of the input becomes larger it is almost impossible to achieve convergence to certain values of weights of the model. Even for simple RNNs with a few hidden layers the gradients vanish exponentially, or the model is unable to cache long-term information about its inputs (Hochreiter et al., 2001, p. 2). The problem lies in the way the model handles the gradients when backpropagating: every neuron within a hidden layer calculates its gradient based on the gradient in the previous layer. If the value of that gradient in the previous layer is already really small, the change to the next layer will be even smaller. This is called a vanishing gradient, where the gradients become too small for a RNN to train effectively.

2.2.7 Long Short-Term Memory neural networks

A popular way to solve the vanishing gradient problem that RNNs face is a Long Short-Term Memory neural network (LSTM). The architecture of a LSTM ensures a more appropriate gradient-based backpropagating method to overcome vanishing gradients as described in Section 2.2.6 (Hochreiter and Schmidhuber, 1997, p. 1735). The big difference to a RNN is the presence of memory cells with gate units in each neuron that control the information-flow within a hidden layer. The gate units regulate information-flow from previous memory cells (Hochreiter and Schmidhuber, 1997, p. 1741). A memory cell has an input gate that protects the memory contents from previous neurons with irrelevant interference from current inputs or ignores information from previous neurons (Hochreiter and Schmidhuber, 1997, p. 1742). Similarly, a memory cell contains an output gate which combines the current cell state with relevant previous hidden states to create a new hidden state that is passed on to the next memory cell (Hochreiter and Schmidhuber, 1997, p. 1743). In summary, a neuron in a LSTM contains a memory cell that controls information via the input gate to decide what information to keep or override from the previous memory cell(s) and the current input, and a memory cell uses the output gate to decide when to access and when to ignore the current cell's information. This mechanism ensures a LSTM properly handles long-term dependencies, reducing the problem of vanishing gradients and it is therefore able to achieve higher performances than RNNs when longer-term dependencies play a role. The training time of a LSTM is still relatively long because the backpropagating method is computationally expensive.

2.2.8 Transformer-based models

Before 2018 the most popular Natural Language Processing models for processing pieces of text were neural networks which required a lot of training time. A new type of neural network architecture developed by Google, the 'Transformer', successfully uses attention mechanisms to achieve state-of-the art (i.e., the highest level of development) results on multiple NLP-tasks in 2017 (Vaswani et al., 2017). The Transformer-based models

perform better in the field of Natural Language Processing than RNNs and LSTMs (Vaswani et al., 2017; Devlin et al., 2018; Liu et al., 2019; Wang et al., 2018, 2019). The attention mechanism allows the model to take relevant parts of the input data into account when it is needed. Instead of focusing on only the last hidden state as visualised in Figure 4 for RNNs, a Transformer-based model can take all hidden states from all parts of the input into account. Where the architecture of previously mentioned neural networks like RNN and LSTM results in sequential (i.e., piece after piece in logical order) processing of data, a Transformer is able to apply parallelization during the training process. This strategy comes down to handling multiple inputs simultaneously and boosts the training speed of the model. In the next paragraphs the structure of a Transformer-based model and the way it uses attention to link input words are explained. In Appendix A in Section 8 a more detailed and mathematical explanation about the attention mechanism of Transformer-based models is given. Two different Transformer-based models are applied to the classification task of this research. A row of the dataset in this research which is processed by a Transformer model consists of one case-law article. A Transformer model then splits an article into tokens using a tokenizer, where a token is equal to a word or a part of a word (for longer words). These tokens are then represented by word embeddings as described in Section 2.2.3.

It is important to make a distinction between encoders and decoders within a Transformer-based model. An encoder processes input words, whereas a decoder produces the output of the model. This is similar to the input and output layer in a RNN. In the example of a machine translation application from one language to another a Transformer model consists of a set of n encoders, n decoders, connections between them, a linear layer and a softmax layer as is visualized in Figure 5 (Vaswani et al., 2017, pp. 2-3). So, in this case the encoder processes the words of the language of the input sentence and the decoder processes the words of the language requested for the output. In other text-processing tasks such as classification, only one of the encoder or decoder part of the model is used because the output of the model is just one value instead of a complete sentence. So, in this research a Transformer model only consists of an encoder or a

decoder because it is applied on a document classification task. Before entering the model, a word embedding is prepared that represents the words in vectors. Because the model does not handle data sequentially, a positional vector is added to each word embedding vector before a sentence enters the encoder and decoder layers in order to account for the order of an input sentence (Vaswani et al., 2017, pp. 5-6). In Natural Language Processing, this positional vector indicates the position of a word in a sentence and is of the same dimension as the word embedding in order to be able to sum up the two.

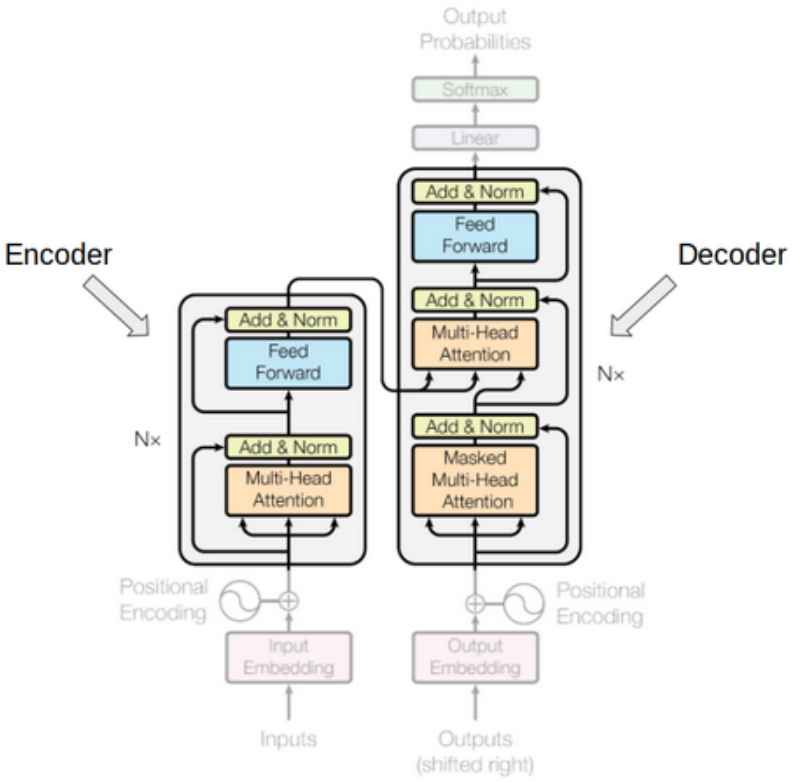

Figure 5: Transformer model structure (Vaswani et al., 2017, p. 3 - customized).

The Transformer model in Figure 5 contains one encoder and one decoder, but a Transformer model typically contains several encoders and decoders (denoted as N in the figure). The structure of the encoders is identical. The encoders split up an input sentence

to a sequence of tokens, covering a part of the information of the sentence each (Vaswani et al., 2017, p. 2). An encoder contains two parts: a multi-head attention mechanism and a feed-forward neural network. For textual data the self-attention mechanism takes other parts of the input into account while processing a particular token (Wu et al., 2021, pp. 2-3). So, the multi-head attention mechanism allows the Transformer model to direct its focus to other parts of the input. In Appendix A in Section 8 the concept of self-attention for Transformer-based models is illustrated in detail. After the data is processed by the self-attention mechanism within an encoder the output is passed on to a feed-forward layer that transforms the data such that it can forward the information to the next encoder (Vaswani et al., 2017, p. 5). The feed-forward layer ensures that it is possible to execute the transformations to the data in the layer in parallel, which increases the speed of the training phase of the model (compared with RNNs and LSTMs, where transformations and calculations can not be executed in parallel and therefore generally have longer training times).

Decoders work in a similar way as encoders. A Transformer usually contains multiple decoders (in Figure 5 denoted as N) which all have the same structure. The only difference with encoders is that decoders have an additional layer at the beginning (in Figure 5 denoted as masked multi-head attention) that directs its focus to important parts of the encoder's output. It does that by processing the output of the top encoder, which indicates what the important parts of the input are (see last paragraph of Appendix A in Section 8 for a detailed explanation of masked multi-head attention). Finally, in the example of a translation task the output of the decoders for a single word in the original language is a vector that indicates the translation of that word in the desired language while taking the context of the word into account. The output of a Transformer model in this translation example is transformed by a linear layer and a softmax layer. The linear layer transforms the output into a large logits vector with each cell in the vector corresponding to a word in the vocabulary in this example of a translation task (Vaswani et al., 2017, p. 5). This vector is the input of the final softmax layer that transforms every cell of the logits vector to a positive number, adding all up to 1. This can be seen

as probabilities for a certain outcome (i.e., a word in this example). The cell containing the highest value is chosen by the model and the word (or category, in case of text classification) corresponding to this cell is produced as output. So, in the example of text translation, this is a word in the desired outcome language. In the example of classification of case-law articles, which is done for this research, the potential outcomes are different sub-areas of taxation types. The input for the Transformer-based model in this research is the complete text of a case-law article. In the next two paragraphs an example of a translation task is given to clarify the practical functioning of a Transformer-based model.

Consider the sentence "The dog was unfortunately hit by the car because it did not have its lights on" that needs to be translated into the German language by using a Transformer-based model. First, the input sentence is split up by a tokenizer, such that all words are separated. The word embeddings of all words are retrieved and the positional encoding is added to those embeddings. Now the Transformer-based model is able to recognise the word embeddings and the place of the words in the input sentence. This is the input for the model. Each piece of the input (i.e., a word) enters the first encoder at the same time. The first encoder focuses not only on the currently processed piece of input, but also on other parts of the input through the multi-head attention layer. Each piece of input has its own path in the multi-head attention layer. There are dependencies (i.e., crosses) between these paths when words have a strong connection like the words 'car' and 'it' in the input sentence. In this way the Transformer-based model is able to derive connections within an input. For the translation it is important for words like 'it' in the example sentence to determine which word in the sentence it refers to. This is less important for the word 'dog', because it is usually just translated with the same German word (the Transformer-based model was able to learn this during the training phase). After flowing through the multi-head attention layer, all inputs are separated again and follow their own independent path through the feed forward layer of the encoder. In the feed-forward layer there are no dependencies between pieces of input, so all paths can be executed in parallel. This process repeats itself in each encoder of the

Transformer-based model. The multi-head attention between pieces of input is different in each encoder, because each encoder focuses on a slightly different part of the input sentence. The combined multi-head attention mechanism in every encoder ensures that the model recognises that the word 'it' in the example sentence relates to 'car' and not to 'dog', for example.

The outputs of the last encoder are multiple vectors for each piece of input. These vectors are passed on to the masked multi-head attention layer in each decoder. These layers are responsible for sending information about relevant parts for each piece of input to the decoders. This ensures that the decoders are now able to recognise that the word 'it' from the input sentence relates to the word 'car' when creating the translation. So, the masked multi-head attention layer is basically transmitting the relevant information from the encoders to the decoders. The other two layers in the decoder are again a multi-head attention layer and a feed-forward layer. These two layers work exactly the same as in the encoders, except that they operate in the desired language the input sentence needs to be translated in (in this case German). The input words flow through all decoders and the decoders use the attention-scores for each word provided by the encoders and the masked multi-head attention layer to translate the words. The output of the last decoder are multiple vectors for each word from the input. Finally, a linear and a softmax layer transform these vectors into the specific word in the desired language which has the highest probability to be the translation. In this way the Transformer-based model recognises that the word 'it' in the input sentence relates to 'car' and can be translated into the German language accordingly.

In the paper by Vaswani et al. (2017) the Transformer-model structure is introduced. The model is trained on an English-German dataset that contains more than 4 million sentence pairs and an English-French dataset containing about 36 million sentences (Vaswani et al., 2017, p. 7). Shortly after the publication of this original paper, Radford et al. (2018) invent a new training method named Generative Pre-Training for Transformers, where a model is trained through a procedure of two stages. The first stage is to pre-train a language model on a large corpus of text. In this stage a variant

of the Transformer model is applied where only the decoder part is used for processing textual data (Liu et al., 2018). Before the training procedure starts the text is divided into tokens by a tokenizer. A tokenizer splits up the text into words (by splitting on spaces) and splits up some words into pieces (longer words into smaller words). This is done because otherwise many (longer) words will be extremely rare. Radford et al. (2018, p. 3) want to predict a token based on previous tokens in a sentence in this training phase. So, the likelihood of the sum of conditional probabilities for a target token (i.e., token to predict) given one or multiple context tokens (previous tokens) is maximised during training. The context tokens which are taken into account by the model all occur before the target token in the sentence. After this pre-training stage the supervised fine-tuning stage follows. In contrast to the first stage, this stage is carried out using a labeled dataset. An example for this stage is document classification or question answering. The input, a sequence of tokens along with a provided label, is forwarded through the pre-trained model and this is fed to the final two layers of the Transformer model (the linear and softmax layer, see Figure 5) to predict a label (Radford et al., 2018, p. 3). Similarly, the input of the model during the training phase in this research is a complete case-law article with corresponding sub-area of taxation during the fine-tuning phase.

The process of using a pre-trained model trained on huge amounts of text and fine-tuning it for a specific supervised downstream task (like document classification in this research) is called transfer learning (Torrey and Shavlik, 2010; Pan and Yang, 2009). It is called transfer learning because with such a knowledge "transfer" a model can learn language structures from unsupervised pre-training on a prediction task which does not necessarily have to be the same as the task where the model is fine-tuned on. This implies that a pre-trained model can provide information for several different target tasks for which the model is fine-tuned (Pan and Yang, 2009, p. 1). In the field of NLP, an abundance of unlabeled data and a shortage of labeled data is very common. Transfer learning therefore offers possibilities within this area. The structure of this process ensures that the fine-tuning procedure is relatively computationally inexpensive because a model can use the information collected from the unsupervised pre-training phase (like

common language structures etc.) (Devlin et al., 2018, p. 5).

2.2.9 Bidirectional Encoder Representations from Transformers

After the general Transformer model combined with Generative Pre-Training for Transformers achieved state-of-the art (i.e., the highest level of development) results on multiple Natural Language Processing tasks in 2017, various adjustments were made in the next years to improve the model. Bidirectional Encoder Representations from Transformers (BERT) was developed by researchers at Google and was the best-performing model within NLP for a wide range of tasks during 2018 and partly in 2019 (Devlin et al., 2018). As the name of the model suggests, a BERT model is able to process texts and provide word representations (i.e., word embeddings) from unlabeled pieces of text by jointly analysing context left and right from the target word (or: token) during the pre-training phase (Devlin et al., 2018, pp. 1-2). While the Generative Pre-Training method only used a unidirectional model (i.e., accounting for context to the left of the target word) BERT can pre-train bidirectional representations. Devlin et al. (2018, p. 1) argue that it is crucial to take context from both sides of the target token into account, especially for tasks on the sentence-level, paragraph-level and document-level. Intuitively it makes sense when processing pieces of text to not only look at words occurring before a word but also after a word, since this is what humans tend to do when processing texts as well. This applies to the English language and also to the Dutch language. The improvement achieved by BERT is therefore not due to a change in the Transformer structure itself, but rather to changes in the training phase.

The way BERT changes the pre-training phase is by applying two new unsupervised tasks. Firstly, a masked language model is applied (Devlin et al., 2018, p. 4). The text is again split up into tokens by applying a tokenizer. The word 'masked' implies that some tokens are hidden from the model and it needs to predict these tokens based on the context (i.e., other tokens occurring before and after the token). In each input sequence a randomly selected 15% of the tokens is masked. A sequence can be one sentence or two sentences packed together and a sentence does not necessarily have to be an actual

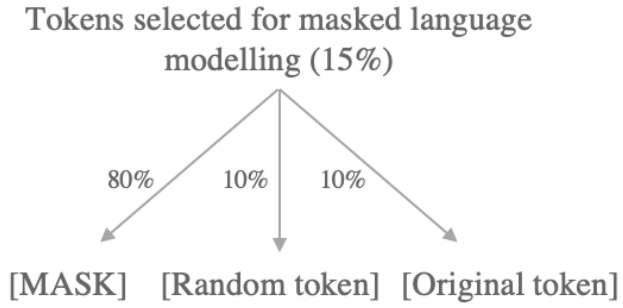

Figure 6: Masked language model design from BERT.

linguistic sentence but can also be just a span of text (Devlin et al., 2018, p. 4). From the randomly chosen 15%, 80% of the tokens are replaced with the [MASK] token (i.e., the words are hidden/masked), 10% with a random token and the other 10% of the tokens remain the same. In Figure 6 the selection of tokens for the masked language model procedure is illustrated. The original token that is replaced by these tokens is then predicted by the model during this pre-training phase. A big advantage of this structure is that it is unknown which words have been replaced by random words, so the model takes every token into account for representation (Devlin et al., 2018, p. 12). So, the general idea behind the combination of masked tokens, randomly selected other tokens and unchanged tokens is as follows:

- If each token from the randomly chosen 15% was masked with the [MASK] token, the model would only take these words into account. The model does not produce token representations for the other words that are not masked in this case;
- If each token from the randomly chosen 15% was either masked or replaced by a random token, the model would learn during the training phase that the observed token is never the correct one;
- If each token from the randomly chosen 15% was either masked or kept the same, the model could just copy the tokens that are not masked.

In the second stage of pre-training BERT the procedure of next sentence prediction is applied, because the relationship between sentences is important for downstream tasks

such as question answering and information retrieval (Devlin et al., 2018, pp. 4-5). Next sentence prediction is a binary classification task and asks the model to predict if two sentences from a corpus are consecutive or not. In each training example the probability that a sentence comes consecutively after a target sentence is 50% (Devlin et al., 2018, p. 4). In the other 50% of the time the following sentence is a random other sentence from the corpus.

There exist many tokenize-methods to create tokens from text. BERT uses WordPiece embeddings to tokenize words (Devlin et al., 2018, p. 4). WordPiece embedding is a way to improve the handling of rare words by a model by dividing words into a limited set of sub-words (i.e., word-pieces or tokens) (Wu et al., 2016, p. 2). For example, the word "arrangement" can be split up into "arr", "ange" and "ment" to ensure that there are more of the same tokens. The word "arrangement" would probably not occur many times in the full dataset, but when it is split up the parts occur more frequently. The first token of every input sequence is a special classification token (named as CLS-token) that is used as the aggregate sequence representation, mainly relevant for classification tasks (Devlin et al., 2018, p. 4). This token contains information about a complete input sequence (which consists of a pair of sentences). The next sentence prediction phase trains the special classification token, such that it contains global information about the two sentences within a sequence. The sentences are separated with a special separator token (named as SEP-token) that indicates where one linguistic sentence stops and another one begins. Each token within a sequence (so, each sub-word) is calculated as the sum of the learned token embedding (using WordPiece embedding, covering only the sub-word itself), the segment embedding (indicating to which sentence a token belongs) and a positional embedding (indicating the exact position of a token within the overall input sequence) (Devlin et al., 2018, p. 5).

2.2.10 Robustly optimized BERT approach

After BERT achieved state-of-the art (i.e., highest level of development) results on multiple NLP-tasks in 2018, Facebook designed the "robustly optimized BERT approach"

(named as RoBERTa). This method revealed opportunities to improve BERT by changing both pre-training objectives as well as the pre-training structure and adjusting the model's hyperparameters. Liu et al. (2019) find that the original BERT method was under-trained (i.e., training the model was not sufficiently dynamic and large-scaled trained, see next paragraphs) and that adjusting the model's key hyperparameters leads to improved performance on the majority of NLP-tasks.[3]

The masked language model pre-training phase of BERT was to randomly select 15% of the tokens in each input sequence once (see Figure 6). RoBERTa adjusts this process from static masking to dynamic masking to avoid using the same mask for each training instance in every epoch (i.e., a complete iteration through the training dataset) (Liu et al., 2019, p. 4). The dynamic masking process implies that the training data is duplicated 10 times, in order to make sure every input sequence is masked in 10 different ways (Liu et al., 2019, p. 4). The model follows this pre-training phase over 40 epochs such that each input sequence is analysed four times by the model with the same masked tokens. Dynamic masking achieved similar or slightly better results in all Natural Language Processing tasks than BERT's static masking.

Another adjustment in the pre-training phase of RoBERTa is made for the next sentence prediction objective. Liu et al. (2019, pp. 4-5) find that when the next sentence prediction task is removed during pre-training and the focus lies only on the masked language modelling task, the performance of downstream tasks remains the same or even improves. They additionally find that restricting input sequences to come from the same document achieves slightly better performance. However, in this case input sequences for the model can differ in length because the end of a document aborts an input sequence. When this happens, Liu et al. (2019, p. 5) increase the batch size (i.e., the number of samples before the model's parameters are updated) to get the same number of tokens again for every sequence. Although handling the restriction that an input sequence cannot cross document boundaries slightly improves performance, the format automatically leads to variable batch sizes which does not outweigh the better

[3]See: (Super)Glue Leaderboard - https://gluebenchmark.com/leaderboard.

results according to the authors (Liu et al., 2019, p. 5). Therefore, the next sentence prediction task is completely removed from pre-training and the input sequence is allowed to cross document boundaries in the RoBERTa model.

Finally, RoBERTa achieves higher performance in most Natural Language Processing tasks[4] than BERT because of a significantly longer training process with larger batches and more pretraining steps (Liu et al., 2019, pp. 5-6). The model also uses a larger vocabulary size containing more subword units and is pre-trained on 10 times more textual data than the BERT model (Liu et al., 2019, pp. 2-3). Like mentioned earlier, longer training time usually equals better results for Natural Language Processing tasks.

While there are multiple other Transformer-based models used within Natural Language Processing, this research focuses on the models described in Sections 2.2.9 and 2.2.10. The main reason for this is that these models have a pre-trained version on only Dutch text and are therefore applicable within this research, since the case-law articles are all in Dutch. The models specifically designed for Dutch texts are explained later in the methodology section. It would not make sense to apply a model that is pre-trained on the English language in this research, because this model would only be familiar with English words and would not recognise Dutch words because it is not trained on Dutch text. It is noticed by the author that the new Transformer-based model "Text-to-Text Transfer Transformer (T5)" is currently the state-of-the art (i.e., the highest level of development) for multiple Natural Language Processing tasks (Wang et al., 2019; Raffel et al., 2019). However, this model is not pre-trained on Dutch text and is therefore not suitable for this research.

2.2.11 Discussion

With the introduction of word embeddings which are able to capture the meaning of words in vectors, and neural networks which can account for the order of words within input sequences, the number of dimensions of the vectors representing words is reduced and the assumption that a document is just a bag-of-words is addressed and (partially)

[4]See: (Super)Glue Leaderboard - https://gluebenchmark.com/leaderboard.

solved. Since 2017 Transformer-based models that use word embeddings as starting point have completely taken over the power from recurrent neural networks and Long Short-Term Memory neural networks in the area of Natural Language Processing. The self-attention mechanism makes a difference compared to previous methods, because it provides the ability to account for longer-term dependencies (i.e., connections between words and sentences) within textual data. The successful application of a knowledge transfer between a pre-trained model on big amounts of textual data and a model that can be fine-tuned for a specific downstream task like document classification or question answering is another improvement in both performance and additional training time. The training time is also reduced through the possibility of parallelization.

With the various introductions of Transformer-based models recently it is difficult to decide which model suits the best for a specific task. Subsequently, it is also hard to determine if higher performances on Natural Language Processing tasks by a model come from longer training time on more data or from enhanced design choices in the pre-training phase. There are various leaderboards like GLUE and SuperGLUE where Natural Language Processing models are compared along (Wang et al., 2018, 2019). However, downstream tasks evaluated by these leaderboards differ significantly in essence to this research's classification task and the leaderboards only compare methods on English language tasks as well.

It is expected that the new Transformed-based models achieve higher performances in classifying case-law articles into sub-areas of taxation compared to recurrent neural networks and Long Short-Term Memory neural networks structures, since these new models dominate leaderboards on Natural Language Processing tasks. A pre-trained Transformer-based model can be used directly on the dataset containing the fiscal case-law articles or can be further pre-trained on the data with the pre-training objectives described in Sections 2.2.9 and 2.2.10. One would expect that a model performs better when it is further pre-trained on the particular data from this research, because legal texts are different in jargon (i.e., use of words) compared to regular texts. However, this is a computationally expensive task. In addition, one would expect that Transformer-based

models pre-trained on Dutch texts strongly defeat models pre-trained on English texts in performance of classifying Dutch case-law articles into sub-areas of taxation because these models are more familiar with Dutch language structures and common combinations of words (Delobelle et al., 2020). It is interesting to see how the models perform in combination with the methods from Section 2.1 on a Dutch multi-category classification task with lengthy and relatively complicated legal documents.

3 Data

In this section the data used for this research is outlined. First, the source and context of the data is described. After that the data cleaning process is described and the variables of the dataset are listed and explained, including some descriptive statistics to explore the data. Finally, the process of creating the test set to evaluate the performance of the methods is explained.

3.1 Data source

This research is conducted at Bluetick. Bluetick is a startup in the LegalTech field and has created an Artificial Intelligence-based search engine for lawyers and tax advisers who are looking for Dutch or European case-law articles. The search engine can find similar documents, similar textual passages within documents and is able to provide personal recommendations based on the searching behaviour of the user (similar to, for example, recommended videos on Youtube). The data used in this research comes from one of the databases of Bluetick. This database contains all case-law articles available on the open and free website www.rechtspraak.nl. From all case-law articles only the ones that have the label "Tax law" as legal field are selected. This is the data used in this research to classify the fiscal case-law articles into sub-areas of taxation. After adding labels for specific sub-areas of taxation, Bluetick can add an extra filter within their search-engine that indicates to which type of tax a document belongs.

3.2 Data preparation

A fiscal case-law article in the dataset contains metadata that describes the characteristics of the document. In Table 9 in Appendix B in Section 9 an overview of the variables in the metadata is given. Like earlier mentioned, only the case-law articles with subject "Tax Law" are selected for this research. In total, this concerns 59671 fiscal case-law articles by 1 September 2021. The variables in the metadata are not used in this research, because they do not add useful information for the classification of fiscal case-law articles

to specific type of taxation and the goal is to classify the articles based on the text only.

Some functions are prepared to extract the raw text from the documents. This ensures that there is no extra information in the case-law articles besides the actual raw text of a document and thus removes unwanted pieces of text such as automatically included HTML-code. The selected case-law articles are placed in a dataframe with three columns: a row identifier number (from 1 to 59671), the ECLI number (a unique number for every court-case) and the text of the case-law article.

After performing additional analysis, a fourth column is added that indicates the number of words of a case-law article. The average number of words for a case-law article in the initial dataset is 2605. The maximum number of words for a case-law article is 74774 and the minimum is 0. Empty case-law articles (i.e., without words) cannot be classified into a specific type of taxation. These documents are removed from the data. After additional exploration of the data it is detected that 131 case-law articles contain exactly 4 words. The 4 words indicate that a judgement for the particular case is not published. So, these documents are also removed from the data since they cannot be classified. As a general rule of thumb, it is decided that case-law articles containing less than or exactly 100 words are removed because they do not contain relevant information about the subject of a case-law article. Case-law articles of more than 100 words generally contain relevant information about the particular case and are therefore classifiable. This eventually removes 2239 case-law articles from the dataset.

Besides removing case-law articles containing less than or exactly 100 words other case-law articles need to be removed from the data for this research as well. This concerns case-law articles that only indicate that a particular appeal has been rejected. These case-law articles do not indicate the content of a court case, but only contain the information that an appeal by a defendant has been rejected. According to article 81 of the Dutch Judicial Organisation Act (in Dutch: "Wet op de Rechtelijke Organisatie"), a court does not have to argue this decision (but sometimes still does). Hence, there is no indication of content in most of these case law articles. In Figure 7 a histogram is created of the case-law articles that contain article 81 of the Dutch Judicial Organisation Act

and the number of words they consist. In total there are 3317 case-law articles containing the specific article of law.

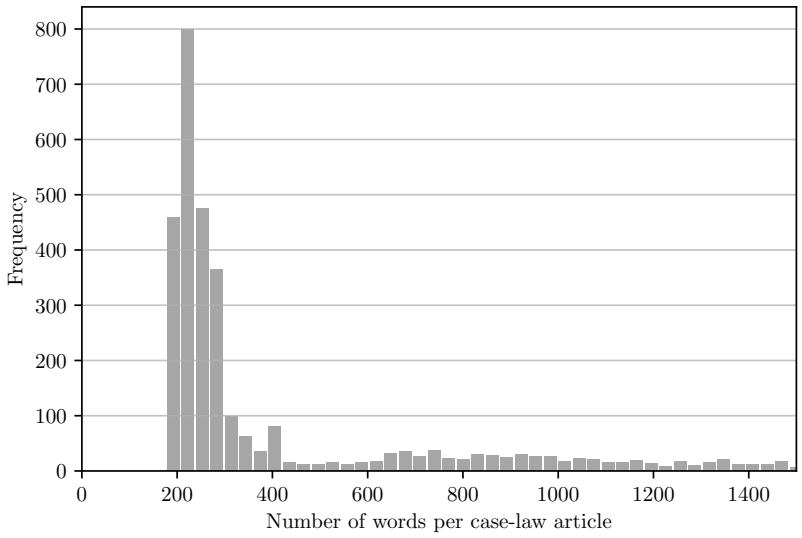

Figure 7: Histogram of the number of words per case-law article which contain article 81 of the Judicial Organisation Act.

The decision that an appeal is rejected without providing arguments by the court is usually published on a maximum of one page. If the case-law article is longer than one page it means that the judges did discuss the content of a case. Therefore, the case-law articles that contain article 81 of the Dutch Judicial Organisation Act and consist less than or exactly 400 words (which is approximately one page) are deleted from the dataset because they do not contain relevant content. In Figure 7 it becomes clear that this concerns most of the case-law articles in which the specific article of law occurs. Articles over 400 words remain preserved in the dataset because they mostly do contain specific content about the subject of the legal case. In total this removes 2310 case-law articles. Similarly, case-law articles which contain article 80a of the Dutch Judicial Organisation Act do not indicate information about a relevant subject as well. These articles only indicate that a particular appeal has been rejected, just like case-law articles that contain article 81 of the Judicial Organisation Act. The same approach is applied for the case-law articles containing article 80a of the specific Act. So, if these articles contain less than or

exactly 400 words they are removed from the data. This removes 1135 case-law articles. Finally, some case-law articles of the Supreme Court (in Dutch: "Hoge Raad") indicate that an appeal in cassation is not admissible because certain required aspects of the appeal are missing. These case-law articles do not indicate information about the subject of the court case and are therefore removed from the data by filtering on the text "The Supreme Court declares the appeal in cassation inadmissible" (in Dutch: "De Hoge Raad verklaart het beroep in cassatie niet-ontvankelijk") for case-law articles of below or exactly 400 words. This removes 590 articles, leaving 53397 case-law articles to be classified for this research.

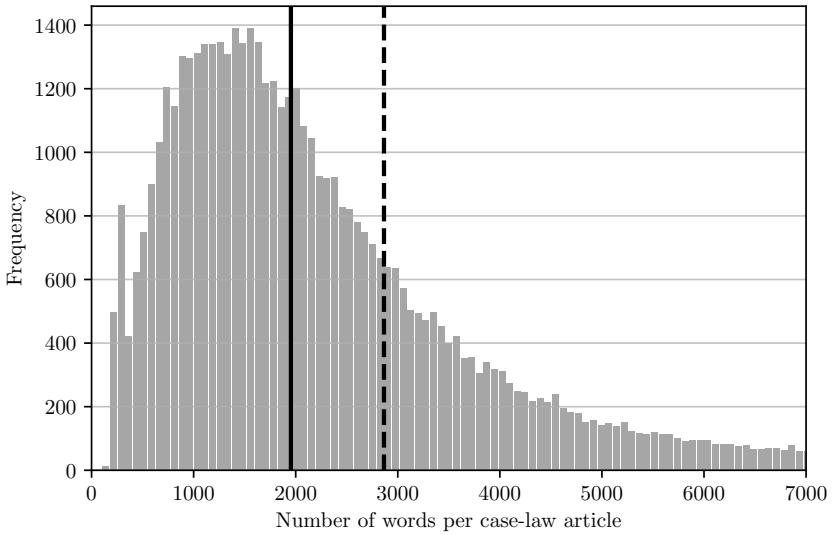

Figure 8: Histogram of the number of words per case-law article in final data.

In Figure 8, a histogram is created that shows the number of words per case-law article for the final data used in this research. The average number of words per case-law article in the final cleaned data used in this research is 2893 (dashed black line in Figure 8) and the median is 1971 (solid black line in Figure 8). This indicates that there is a right-skewed distribution, which means that there are several relatively long case-law articles. In Figure 8, case-law articles above 7000 words are not shown because there are some articles with an extremely high number of words (maximum number of words for a case-law article is 74774). Showing these outliers makes the histogram unreadable.

3.2.1 Sub-areas of taxation

In this research fiscal case-law articles are classified into sub-areas of taxation. In Table 2 a schematic overview of these sub-areas (i.e., types of Tax Law) is given in English and Dutch. The Dutch translation of the sub-areas of taxation is given because these are typical names that will be searched for later in this research when labeling a part of the dataset. The sub-areas considered for this research are directly obtained from the chapters of the Dutch Tax Law Book 2021. Because some sub-areas of taxation are relatively similar and contain few legal articles in the book of Tax Law they are joined together. This is specifically the case for the sub-areas concerning international taxes and for the general state taxes. The grouping of these sub-areas results in a total of 15 categories to classify case-law articles, given in the last column of Table 2.

Table 2: Final sub-areas of taxation in Dutch Tax Law.

Sub-area of taxation (NL)	**Sub-area of taxation (translation)**	**Final sub-areas of taxation for this research**
Inkomstenbelasting	Income tax	Income tax
Vennootschapsbelasting	Corporate tax	Corporate and dividend tax
Dividendbelasting	Dividend tax	
Erf- en schenkbelasting (successiewet)	Inheritance and gift tax	Inheritance and gift tax
Omzetbelasting	Turnover/Sales tax	Turnover/Sales tax
Overdrachtsbelasting (Wet op belastingen van rechtsverkeer)	Transfer Tax (Legal Transactions Tax Act)	Transfer Tax (Legal Transactions Tax Act)
Loonbelasting	Wage/Payroll tax	Wage/Payroll tax
Autobelastingen	Car taxes	Car taxes
Milieuheffingen	Environmental taxes	Environmental taxes
Douanebelasting	Customs tax	Customs tax
Diverse belastingen (kansspelbelasting, bankenbelasting, verhuurderheffing)	Various taxes (gambling tax, bank tax, landlord levy)	Various taxes (gambling tax, bank tax, landlord levy
Algemeen bestuursrecht	General administrative law	General state taxes and administrative law
Algemene wet inzake rijksbelastingen	General act on state taxes	
Invorderingswet	Collection act	Collection act
Belastingen lagere overheden	Local government taxes	Local government taxes
Waardering onroerende zaken	Valuation of real estate	
Toeslagen	Allowances	Allowances
Internationale bijstandsverlening bij de heffing van belastingen	International tax assistance	
Voorkoming dubbele belasting	Prevention of double taxation	International tax law
Mensenrechtenverdragen	Human rights conventions	
EU-recht	EU law	

3.2.2 Contextual keywords

For the method of providing labels as described in Section 2.1.3 specific contextual keywords for each sub-area of taxation need to be formulated. The occurrence of contextual keywords within a case-law article provides a sub-area of taxation label for that specific case-law article in this method. Only a part of the case-law articles in the data are classified by this method, because not all articles contain a listed keyword. The keywords are established based on frequently occurring words within the legal text of the Tax Law Book of the specific sub-area of taxation, or based on my own knowledge[5] of the specific sub-area. It is important that these keywords occur exclusively in one sub-area of taxation, because otherwise a case-law article could be labeled incorrectly. In Table 3 the contextual keywords for each of the 15 sub-areas of taxation are given.

Table 3: Contextual keywords for the sub-areas of taxation.

Sub-area of taxation	**Keywords**
Income tax	Inkomstenbelasting (IB), fiscale partner(s), winst uit onderneming, ondernemersaftrek, belastbaar/belastbare inkomen uit werk en woning, belastbaar/belastbare inkomen uit sparen en beleggen
Corporate and dividend tax	Vennootschapsbelasting (VPB), dividendbelasting, dividend, deelneming, deelnemingsvrijstelling
Inheritance and gift tax	Erfbelasting, schenkbelasting, erfrecht, schenkingsrecht, successiewet
Turnover/Sales tax	Omzetbelasting (OB), verlaagd tarief BTW, normaal tarief BTW, BTW-aftrek/aftrek van btw, nul(-)tarief
Transfer Tax (Legal Transactions Tax Act)	Overdrachtsbelasting, assurantiebelasting, fictieve onroerende zaak, wet op belastingen van rechtsverkeer, wet BRV/wet BVR/WBRV
Wage/Payroll tax	Loonbelasting (LB), loonheffing, gebruikelijkloonregeling
Car taxes	Autobelasting(en), personenauto, motorrijtuigen, BPM
Environmental taxes	Milieuheffing(en), wet milieubeheer, afvalstoffenbelasting, afvalstoffenheffing, afvalstoffenwet, kolenbelasting, energiebelasting, leidingwater
Customs tax	Douane, (anti)dumpingrechten, (wet op de) accijns, communautair douanewetboek
Various taxes (gambling tax, bank tax, landlord levy)	Kansspelbelasting (wkb/ksb), bankenbelasting, verhuurderheffing (wvh)
General state taxes and administrative law	Algemene Wet Bestuursrecht (AWB), Algemene Wet inzake Rijksbelastingen (AWR), proforma, WOB(-verzoek), dwangsom, vergrijpboete, verzuimboete
Collection act	Invorderingswet (IW), invorderingsrente, bestuurdersaansprakelijkheid
Local government taxes	Grondwet, provinciewet, gemeentewet, waterschapswet, waterschapsbelasting, wet waardering onroerende zaken (wet WOZ), waardepeildatum, onroerende-zaakbelasting(en), hondenbelasting
Allowances	Algemene wet inkomensafhankelijke regelingen (awir), zorgtoeslag, wet op de zorgtoeslag (Wzt), zorgverzekeringswet, huurtoeslag, wet op de huurtoeslag, kinderopvangtoeslag, wet kinderopvang
International tax law	Internationale bijstandsverlening, voorkoming (van) dubbele belasting, OESO-modelverdrag, Verdrag, belastingverdrag, internationaal verdrag inzake burgerrechten en politieke rechten (IVBPR), EVRM, VWEU, (EU-)Fusierichtlijn, Moeder-dochterrichtlijn

[5] A combination of using my own fiscal knowledge and by investigating the Dutch Fiscal Code Taxonomy, see: https://fiscaletaxonomie.pleio.nl.

3.2.3 Test set

For the evaluation of the methods for this research a test set is created. This test set contains manually labeled case-law articles to a specific sub-area of taxation. The test set is a subset of the data with randomly selected case-law articles from the data. It contains 100 case-law articles with each sub-area of taxation represented by a minimum of five articles, because for this research it is relevant to have results for each individual category (to analyse if a model performs better or worse for specific sub-areas of taxation). The selection of case-law articles for the test set proceeds as follows: first, 80 case-law articles are randomly selected from the data. The categories that now contain 5 or more case-law articles are set apart. Second, 20 more case-law articles are randomly selected. If, out of these 20 case-law articles, an article belongs to a category that already has 5 articles (and is thus set apart earlier) that specific case-law article is replaced by a randomly selected other article. This process is repeated until each sub-area of taxation contains a minimum of five case-law articles. In Table 4 the distribution of case-law articles in the test set for each sub-area of taxation is given.

Table 4: Test set.

Sub-area of taxation	**Number of case-law articles**
Allowances	5
Car taxes	8
Collection act	5
Corporate and dividend tax	9
Customs tax	5
Environmental taxes	6
General state taxes and administrative law	6
Income tax	10
Inheritance and gift tax	6
International tax law	5
Local government taxes	6
Transfer tax	7
Turnover/Sales tax	9
Various taxes (gambling tax, bank tax, landlord levy)	5
Wage/Payroll tax	8
Total	100

4 Methodology

In this section the specific methods to classify the case-law articles into sub-areas of taxation are explained. The methods applied in this research are given in Table 5. First, the unsupervised method of Latent Dirichlet Allocation is explained. This method is the baseline of performance for this research and assigns topics to all case-law articles. After that, the methods of searching for references to legal articles and searching for contextual keywords in context of this research are briefly clarified. These two methods label a part of the case-law articles that can be used for training supervised models. Next, the Dutch BERT and RoBERTa Transformer-based models are explained in detail. These supervised models label the case-law articles that are not labeled by the above mentioned methods that search for references to legal articles and contextual keywords. Finally, a model modification is explained to solve the high computational requirements and limits that Transformer-based models face when processing relatively long texts. Through this model modification a Transformer-based model is able to process up to eight times longer documents.

Table 5: Combinations of research methods.

One-step method	
Latent Dirichlet Allocation (Clustering)	
Two-step methods	
Train labeling	**Full labeling**
Searching for references to legal articles	BERT
Searching for references to legal articles	RoBERTa
Searching for specific contextual keywords	BERT
Searching for specific contextual keywords	RoBERTa

4.1 Latent Dirichlet Allocation

In this research the unsupervised Latent Dirichlet Allocation technique is used as a baseline to compare more advanced supervised methods against. Latent Dirichlet Allocation is a

topic modelling method that uses the intuition that every document corresponds to a distribution of topics and every topic can be described by a distribution of words. If two documents contain the same set of words, it is likely that they describe the same topic. In this research stopwords within the text are removed and stemming is applied. Stemming transforms words to their root form, such that "install" and "installs" are considered as the same word. Words containing less than 4 characters are removed (because relevant words in the tax domain are usually longer) and some general words occurring frequently in most fiscal case-law articles (see Appendix C in Section 10 for these words) are removed as well. Before applying Latent Dirichlet Allocation the number of different topics k (i.e., the categories) for the dataset needs to be pre-specified and a Term Frequency - Inversed Document Frequency matrix is created. This matrix is created such that the the relative frequencies of words in a case-law article (comparing to word frequencies in other case-law articles) is covered. In this research the number of topics k is 15, because the fiscal case-law articles are classified into 15 sub-areas of taxation (see Table 2).

The Latent Dirichlet Allocation method is applied to the data after the procedures described in the previous paragraph are executed. A word w is denoted as an index p by one-hot encoding, which means that the index is as long as the full vocabulary of the dataset (i.e., all unique words) and equals one at the place of the specific word and zero for all other words (Blei et al., 2003, p. 995). A document is then represented by a vector of N words denoted as $\boldsymbol{w} = (w_1, w_2, ..., w_N)$ where w_N is the nth word of a document. The full corpus D (i.e., the complete dataset) consists of all documents, such that $D = (\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_Z})$ where $\boldsymbol{w_Z}$ is the Zth document in the corpus. Eventually, for each word w in all documents $\boldsymbol{w}$ in corpus D a topic is assigned through a fixed process.[6]

The Latent Dirichlet Allocation technique can be summarised in the following conditional probability (Blei et al., 2003, p. 996):

$$P(\theta : Z, \boldsymbol{v} : Z, \beta : k | D; \boldsymbol{\alpha} : Z, \boldsymbol{\eta} : k)$$

where:

[6]See Blei et al. (2003, p. 996-997) for a detailed description of this process.

- θ is a matrix containing the distribution of topics for each of the Z documents. $\theta_{i,j}$ is the probability that the document in row i is about the topic in column j;
- $\boldsymbol{v}$ is a vector with the number of topics k for each of the Z documents in the dataset;
- β is a matrix containing the distribution of words for each topic for the pre-specified total number of topics k. $\beta_{i,j}$ is the probability that the topic in row i contains the word in column j.

Given:

- corpus D which includes all documents in the dataset;

using the following two parameter vectors that can be fine-tuned:

- $\boldsymbol{\alpha}$ is a vector containing the distribution of topic-parameters for each of the Z documents (i.e., the document-topic distribution);
- $\boldsymbol{\eta}$ is a vector containing the distribution of word-parameters for each of the k topics (i.e., the topic-word distribution).

One needs to define the parameter-values that approximate the matrices θ and β and the vector $\boldsymbol{v}$ such that documents with similar words belong to the same topic(s) before running the method. A higher α parameter results in more topics to be represented by a document and a higher η parameter results in more words to be represented by a topic. Similarly, low values of α mean that it is likely that a document corresponds only to one or a few topics and low values of η mean that a topic is only covered by a mixture of a few words.[7] In this research there are 10 words per topic and a case-law article is classified as the sub-area corresponding with the highest probability. The Latent Dirichlet Allocation method is applied to identify topics on a sample from the dataset which is of the same size of the training datasets where the supervised Transformer-based models train on (for reasons of comparability). The detailed model specifications are given in Appendix C in Section 10.

[7]For a detailed explanation of the parameter estimation see Blei et al. (2003, p. 1003-1006).

4.2 Labeling a part of the dataset

To move from an entirely unlabeled dataset of fiscal case-law articles to a labeled dataset, the first step is to label a part of the data using specific characteristics. This part can later be used as a training set for supervised models. In the following sections two methods are explained that label a part of the case-law articles by looking at either references to legal articles or by looking at contextual keywords. This concerns the first column of the last four rows in Table 5. The two different training datasets created by these methods are kept the same size, such that the quality of both datasets can be optimally compared (and that the size of the datasets therefore does not play a role).

4.2.1 Extendable Legal Link Extractor

The first method to generate labels for a part of the dataset is by looking at references to legal articles. Fiscal case-law articles sometimes refer to fiscal law articles from the Tax Law Book. It is likely that a case-law article is about a certain subject when a specific sub-area of taxation occurs multiple times within a case-law article as a legal reference. The algorithm that is able to automatically extract references to legal articles from Dutch text is named as the Extendable Legal Link Extractor (van Opijnen et al., 2015). A case-law article is usually processed within 3 seconds making the Extendable Legal Link Extractor suitable for this research. In the next paragraph a brief explanation of the operation of the algorithm is given.

The algorithm applies a combination of Named Entity Recognition and Pattern Recognition to extract legal references from the text (van Opijnen et al., 2015, p. 3-5). On the one hand Named Entity Recognition is applied to select the names of areas of law from the text. On the other hand Pattern Recognition is used to see whether there are specific legal references to laws before or after the selected area of law. In this way the algorithm distinguishes between a piece of text where only the words "income tax" occur and a piece of text where the words "article 1 of Income Tax" occur. The first piece of text is not a reference to a legal article, whereas the second piece of text is a legal reference. Next to the combination of Named Entity Recognition and Pattern

Recognition, van Opijnen et al. (2015, p. 5-6) create a list of special examples which follow specific procedures. This list is applied in content specific components which occur frequently in legal texts. The algorithm has a recall rate (proportion of actual positives identified correctly) of 98% and a precision rate (correct proportion of positive identifications) of 96% on national legislation (van Opijnen et al., 2015, p. 8).

Based on the references extracted by the Extendable Legal Link Extractor, a new variable is created that indicates the legal references of a case-law article. If a case-law article contains references to a specific sub-area of taxation more than three times the article is classified as that specific sub-area of taxation. The rule that a reference to a sub-area needs to occur more than three times is created because there are standard references to legal articles in many case-law articles that are not necessarily about the actual subject of the case-law article.

4.2.2 Contextual keywords

The second method to generate labels for a part of the dataset is by looking at specific contextual keywords for each sub-area of taxation. The keywords corresponding to the sub-areas are given in Table 3. A keyword needs to occur more than three times in a case-law article for it to be labeled as the specific sub-area of taxation. In this way it is ensured that a specific keyword is really relevant to the case-law article and does not just occur once by chance or as a side issue. An example for this is the occurrence of the keyword "Algemene Wet Bestuursrecht (AWB)" (which can be translated as "General administrative law") for the sub-area "General state taxes and administrative law". This keyword occurs at the end of many case-law articles because it is usual to relate to it in the allocation of the costs of the court case (e.g., lawyers fees, travel and accommodation costs) to a participating party. It is not desired to classify each case-law article to the specific sub-area of taxation when that keyword is mentioned once, so keywords need to occur more than three times.

4.3 Supervised methods

The two methods described in Sections 4.2.1 and 4.2.2 classify a part of the case-law articles in the dataset. This classified part is used as a subset of the data where supervised algorithms can train on. The supervised models applied in this research are the Dutch versions of the Transformer-based models "BERT" (see Section 2.2.9) and "RoBERTa" (see Section 2.2.10) and are described in the next two sections. Only the encoder part of the Transformer-based models is used for this research, since it concerns a classification task instead of a sequence-to-sequence task such as translation or summarisation (where a decoder is used to formulate texts as output).

4.3.1 BERTje

A Dutch pre-trained version of the Transformer-based model BERT is applied in this research. This means that the model has the exact same structure as the regular BERT model designed by Devlin et al. (2018), but is trained on Dutch text only such that it is only able to successfully process the Dutch language. The particular model is pre-trained by de Vries et al. (2019) and is named BERTje. It is pre-trained on five different datasets containing Dutch text (12GB in total) by applying the two pre-training phases described in Section 2.2.9. Since the textual data for this research is completely in Dutch (because it only concerns Dutch case-law articles), this pre-trained model is used instead of the general BERT model to classify the case-law articles into sub-areas of taxation. It is assumed that this generates better results because the general BERT model is mostly pre-trained on English text, while the BERTje model is already familiar with Dutch language. Besides, de Vries et al. (2019, p. 5) find that their model achieves better results for all Dutch Natural Language Processing tasks considered in their paper. So, BERTje is applied in this research to classify case-law articles into type of taxation. Because the model is pre-trained on Dutch texts, it is already familiar with the Dutch language in terms of frequently occurring combinations of words and thus with common language structures. In this research the model is further fine-tuned for the specific task of classifying fiscal case-law articles into sub-area of taxation by training on the

labeled data generated by the methods from Sections 4.2.1 and 4.2.2. For a detailed technical explanation of the training procedure and the corresponding parameter values, see Appendix C in Section 11.

4.3.2 RobBERT

Similar as for the BERTje model, there is a pre-trained version on datasets containing Dutch text available for the Transformer-based model RoBERTa as well. The model is called RobBERT and is trained as described in Section 2.2.10, but on Dutch text. This pre-trained model of RoBERTa on Dutch text is applied in this research to classify case-law articles into sub-area of taxation. Delobelle et al. (2020) have pre-trained a Robustly optimized BERT approach (RoBERTa) on a 39GB corpus of Dutch text using the pre-training structure described in Section 2.2.10, such that the model is able to recognise the Dutch language in terms of frequently occurring combinations of words and language structures. RobBERT outperforms both the RoBERTa model (which is mostly trained on English text) as well as other Transformer-based models (including BERT) for various Dutch Natural Language Processing tasks (Delobelle et al., 2020, p. 9). For the same reason as the choice for the BERTje model, the Dutch model RobBERT is applied in this research to classify case-law articles into sub-area of taxation and is further fine-tuned on the labeled data generated by the methods from Sections 4.2.1 and 4.2.2. For a detailed technical explanation of the training procedure and the corresponding parameter values of the model, see Appendix C in Section 11.

4.3.3 Long document attention

Transformer-based models like BERTje (BERT) and RobBERT (RoBERTa) face computational difficulties and limits when they process relatively long documents. When the amount of text to be processed becomes larger, the self-attention mechanism increases exponentially in complexity (Beltagy et al., 2020, p. 1). This makes it not possible (or computationally very expensive) to process long documents of text such as case-law articles. Beltagy et al. (2020) have developed a solution which is called The Long-

Document Transformer (Longformer). This is a modification to the Transformer-based model structure that adjusts the self-attention process. Because of this modification, the computational power required for the self-attention process increases linearly (instead of exponentially) when the amount of text to be processed becomes larger. In the next two paragraphs this model modification is explained.

The maximum amount of tokens the BERT and RoBERTa models can process in one input is 512. Given that each word is usually divided into about two or three tokens, the maximum amount of text these models can process at once is less than one page. Case-law articles are usually much longer than one page, so this makes the processing by the regular BERT and RoBERTa models difficult. It means that only the first 512 tokens of a case-law article are taken into account by the model. To take more content of a case-law article into account for classification the structure of the BERTje and RobBERT models is transformed to a Long-Document Transformer format. The original Transformer model developed by Vaswani et al. (2017) has a self-attention component that is equal to a time and memory complexity of $O(n^2)$, where n is the length of the input (i.e., the amount of tokens entering the model) and O is a constant (Beltagy et al., 2020, p. 3). This is the case because each token in the input calculates its attention for every other token in the original Transformer model. The attention mechanism in the Long-Document Transformer uses a local fixed-size attention window $w_{attention}$ instead. This ensures that a token from an input does not calculate attention for all other tokens, but only inside a pre-specified window. For a pre-specified window $w_{attention}$ a token calculates attention for $\frac{1}{2}w$ tokens on each side. This changes the time and memory complexity to the function $O(w * n)$, which increases linearly with longer inputs n (Beltagy et al., 2020, p. 4).

Furthermore, the attention window is "dilated" such that it reaches more tokens on both sides of the relevant token (Beltagy et al., 2020, p. 4). A dilated attention window means that there are gaps in the attention window for each token. An example of this is that for the token directly next to the specific token the attention is calculated, but for the token next to this direct neighbour the attention is not calculated. This process is

repeated until the fixed size of the attention window is reached. In this example the size of the gap of the dilated attention window is one, because one token is always skipped. The size of the gaps in the attention window d needs to be pre-specified beforehand, just like the attention window w itself. Dilated attention windows have a wider reach while having the same computational complexity as non-dilated attention windows. Next to the dilated attention window the Long-Document Transformer applies global attention for some specific tokens (Beltagy et al., 2020, p. 4). Global attention is the same attention mechanism that is used in the original Transformer model (Vaswani et al., 2017). For text classification tasks the Long-Document Transformer applies global attention to the CLS-token (classification token), which is the first token of each input as described in Section 2.2.9. Global attention means that the relevant token calculates attention for every other token within the input. This attention operation is symmetric: all other tokens also calculate attention to the CLS-token (Beltagy et al., 2020, p. 4). The combination of local fixed-size, dilated and global attention windows in the Long-Document Transformer decreases the computational complexity of processing long documents. This makes it useful for this research to handle fiscal case-law articles containing more than 512 tokens (almost all case-law articles in the data contain more than this number of tokens). The Long-Document Transformer can process inputs of at most 4096 tokens (which equals about 1638 words or 4,5 pages) and the attention pattern of the Long-Document Transformer can be plugged into any pre-trained Transformer model (Beltagy et al., 2020, p. 6). The Long-Document Transformer is therefore applied in combination with the pre-trained Dutch BERTje and RobBERT models in this research. Because of computational limits, the Transformer-based models take 2048 tokens into account in one input (instead of 4096 tokens). This means that only the first 2048 tokens of a case-law article are processed by the Transformer-based models for the classification.

An alternative to the Long-Document Transformer could be to use the standard Transformer-based models and split a lengthy case-law article into multiple batches of 512 tokens and combine or average the predictions for each part of the document. The Long-Document Transformer would not be necessary to apply in that case. However,

long-range dependencies (i.e., dependencies between words or phrases within a case-law article that are more than 512 tokens apart) are not taken into account by this approach. Therefore the usage of the Long-Document Transformer is preferred to process the case-law articles.

4.4 Training procedure

For this research the Transformer-based models are only trained on the labeled datasets created by the methods in Section 4.2.1 and 4.2.2, so that the models are able to analyse relevant features (i.e., words or word combinations) for each sub-area of taxation. The models are trained on the graphics processing unit (GPU) of Google Cloud Service via Bluetick's paid subscription. The reason for this is the required computational power for the models, which is too much for a local device (i.e., a central processing unit or CPU). The matrix multiplications can be executed more quickly on a GPU and the GPU handles tensor operations in a faster way. The models are trained on a NVIDIA Tesla T4 GPU with a 15GB Random Access Memory (RAM). This GPU speeds up the training process up to 40 times. The training datasets are divided into a train and a validation set, following a 80-20% split.

4.5 Performance evaluation

For evaluating the performance of the methods a test set is created as described in Section 3.2.3. The metric used for comparing the performance of the methods and eventually selecting the best one is the accuracy. This is calculated by applying the following formula on the articles within the test set:

$$\frac{\text{Number of correctly predicted articles}}{\text{Total number of articles}} \times 100.$$

The accuracy is selected as metric in this research because it is important that the best model reaches the highest overall performance. A specific sub-area of taxation is not more important for the classification than another sub-area. The articles in the test set

are not processed or trained on by any method and are seperated from the data from the beginning.

5 Results

In this section the results of the methods are given in the order specified in Table 5. First, the two datasets (based on searching for references to legal articles and searching for contextual keywords) are created where the supervised Transformer models can train on. Secondly, the results of the unsupervised Latent Dirichlet Allocation method are illustrated. Thirdly, the results for the method of searching for references to legal articles combined with respectively the Transformer-based models BERTje and RobBERT are given. After that the results for the method of searching for contextual keywords in combination with respectively BERTje and RobBERT are formulated. Finally, an overall overview of the results on the test set from the applied methods in this research is given in Figure 14.

5.1 Creating training datasets

In this research two methods are applied for creating a dataset where the Transformer-based models can train on. In this paragraph the outcome of both approaches is given. The first method to create a training dataset is to search for references to legal articles by using the Extendable Legal Link Extractor, as described in Section 4.2.1. This dataset is referred to as the 'references to legal articles dataset'. The second method to create a training dataset is to search for contextual keywords that characterise specific sub-areas of taxation, as described in Section 4.2.2. This dataset is referred to as the 'contextual keywords dataset'. A reference to a legal article or a contextual keyword needs to occur more than three times in a case-law article for it to be labeled to a particular sub-area of taxation. The methods label a part of the case-law articles in the dataset. The distribution of case-law articles per sub-area of taxation labeled by the methods is presented in Table 6. Because the contextual keywords dataset contains more articles than the references to legal articles dataset a random sample is taken from the contextual keywords dataset, such that the number of case-law articles in both datasets is the same (i.e., 9601). In this way the quality of the datasets can be optimally compared because

the size of both datasets is the same. For Transformer-based models (and for machine learning methods in general) more training data usually leads to better results. This factor should be excluded in order to increase comparability between models for academic purposes. It is already noticeable that the contextual keywords dataset is more balanced (i.e., more equal distribution of case-law articles over the sub-areas of taxation). From the datasets 80% is used for training the models and 20% is used for validation.

Table 6: Training datasets distribution.

Sub-area of taxation	**Number of articles in references to legal articles dataset**	**Number of articles in contextual keywords dataset**	**Number of articles in contextual keywords dataset after sampling**
Allowances	77	162	56
Car taxes	184	1596	532
Collection act	149	376	133
Corporate and dividend tax	247	2536	829
Customs tax	55	727	243
Environmental taxes	215	329	117
General state taxes and administrative law	5039	4564	1407
Income tax	983	7665	2434
Inheritance and gift tax	116	361	117
International tax law	141	1078	357
Local government taxes	1794	3394	1138
Transfer tax (Legal Transactions Tax Act)	12	567	196
Turnover/Sales tax	317	4668	1414
Various taxes (gambling tax, bank tax, landlord levy)	17	225	71
Wage/Payroll tax	255	1689	557
Total	**9601**	**29937**	**9601**

5.2 Results for Latent Dirichlet Allocation

The Latent Dirichlet Allocation method is applied on a random sample of the total dataset to create the 15 topics (i.e., for each of the 15 sub-areas of taxation). This sample is of the same size as the training datasets in Table 6 (i.e., 80% of 9601 articles which is 7680). The size of the datasets is kept the same such that all methods can be optimally compared. After removing general fiscal words (see Appendix C in Section 10), short

words (less than 4 characters) and stopwords and after applying stemming the Latent Dirichlet Allocation method is applied to create topics. In addition, words occurring in less than 15 case-law articles or in more than 50% of all case-law articles are removed to create relatively specified topics. Latent Dirichlet Allocation is trained on the Term Frequency - Inversed Document Frequency matrix of the data to take into account the relative importance of words in a case-law article. The technical training details can be found in Appendix C in Section 10. The 10 most relevant words per topic are given in Table 7. In the last column the sub-area of taxation derived from the top 10 words per topic is given. Not all topics can be clearly defined into a specific sub-area of taxation. Therefore, not all 15 sub-areas of taxation are included in Table 7 and some sub-areas are included more than once.

In Table 8 an overview is given for the corresponding label number for each sub-area of taxation. This is relevant to interpret the confusion matrices that are prepared for each method in this research. In Figure 9 the classifications for each individual sub-area of taxation for the Latent Dirichlet Allocation method are given in the format of a confusion matrix. The values on the diagonal of the confusion matrix are the correct classifications. The Latent Dirichlet Allocation method achieves 20% accuracy on the test set. In Appendix C in Section 10 there is also a classification report available which indicates the precision, recall and f1-score for each sub-area of taxation in the test dataset (see Table 11). In the confusion matrix it becomes clear that the majority of case-law articles (66 out of 100) is assigned to the Turnover/Sales tax category (corresponding with topic 7 in Table 7), while there are only 9 case-law articles in the test set that really belong to this category. This explains the high recall rate and low precision rate for the Turnover/Sales tax category in Table 11 in Appendix C. Overall, Latent Dirichlet Allocation does not classify most of the case-law articles correctly because the created topics by the method do not correspond sufficiently with the actual sub-areas of taxation. The method is unsupervised and is therefore not able to recognize on what specific content the classification should be made (i.e., the specified sub-areas of taxation). A supervised method can possibly be more accurate in classifying the fiscal case-law articles

Table 7: Generated topics with corresponding sub-area of taxation.

Topic	**Most relevant (stemmed) words**	**Assigned sub-area of taxation**
1	Ontvanger, ontvank, aftrek, naheffingsaanslag, woning, uitgav, aandel, betalingsonmacht, pol, aanprakelijkstell	Collection act
2	Aandel, verzet, that, heffingsambtenar, certificat, waard, onroer, verdrag, verzetschrift, ontvank	Corporate and dividend tax
3	Woning, vervuilingsen, verorden, onroer, schip, waard, inkom, navorderingsaanslag, woondoeleind, middel	Local government taxes
4	Auto, naheffingsaanslag, dividendbelast, personenauto, termijn, sigaret, schad, woning, koerslijst, ontvank	Car taxes
5	Woning, waard, onroer, heffingsambtenar, strat, vergelijkingsobject, waardepeildatum, object, taxatierapport, inkom	Local government taxes
6	Herinvesteringsreserv, woning, bedrijfsmiddel, prostituees, onroer, waard, partner, naheffingsaanslag, strat, vervangingsreserv	Income tax
7	Naheffingsaanslag, omzetbelast, navorderingsaanslag, boet, inkom, lening, aandel, ondernem, ontvanger, winst	Turnover/Sales tax
8	Navorderingsaanslag, bankreken, rekeninghouder, informatiebeschik, gerecht, buitenland, ontvank, proceder, goeder, gegeven	General state taxes and administrative law
9	Verzoekster, heffingsambtenar, lijfrent, dwangbevel, klacht, voorlop, heffingsrente, kerk, naheffingsaanslag, gemeent	General state taxes and administrative law
10	Ontvank, track, trac, postnl, afgeleverd, raadsher, gewez, opgegev, treur, wortel	Unknown
11	Gerecht, winstbelast, aruba, werknemer, inkom, naheffingsaanslag, aftrek, woning, verblijf, landsverorden	International tax law
12	Onroer, heffingsambtenar, gemeent, waard, verorden, leges, woning, apellant, object, geraamd	Local government taxes
13	Gemeent, leges, personenauto, auto, verorden, colleg, naheffingsaanslag, heffingsambtenar, vergunn, precariobelasting	Local government taxes
14	Heffingsambtenar, auto, naheffingsaanslag, parkeerbelast, onroer, waard, parker, pand, gemeent, object	Car taxes
15	Boet, auto, navorderingsaanslag, inkom, verhog, naheffingsaanslag, immateri, priv, redelijk, werknemer	Income tax

because it is able to learn relevant features from the data with corresponding sub-area of taxation labels during the training process. In the next sections the results for the supervised Transformer-based methods are described.

Table 8: Sub-areas of taxation with corresponding label number.

Sub-area of taxation	**Label number**
Allowances	0
Car taxes	1
Collection act	2
Corporate and dividend tax	3
Customs tax	4
Environmental taxes	5
General state taxes and administrative law	6
Income tax	7
Inheritance and gift tax	8
International tax law	9
Local government taxes	10
Transfer tax (Legal Transactions Tax Act)	11
Turnover/Sales tax	12
Various taxes (gambling tax, bank tax, landlord levy)	13
Wage/Payroll tax	14

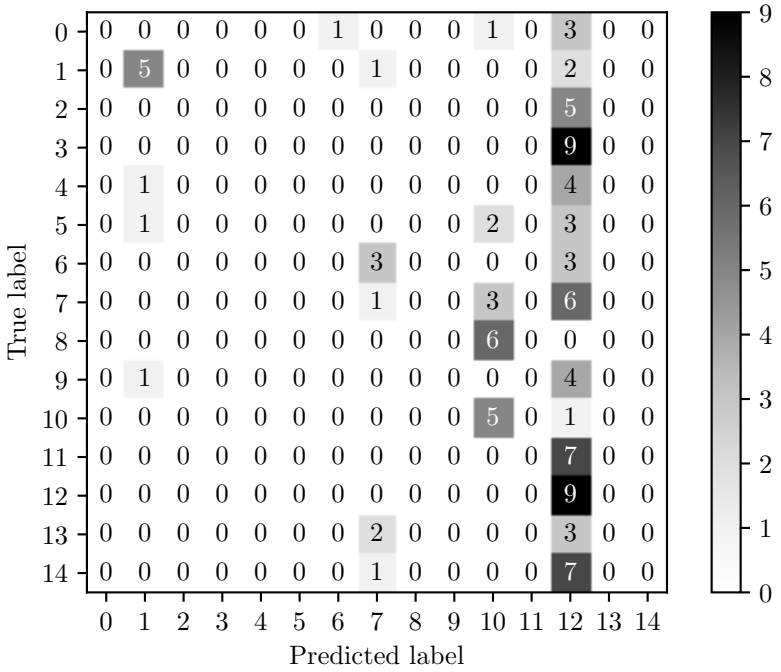

Figure 9: Confusion matrix for Latent Dirichlet Allocation method.

5.3 Results for BERTje trained on references to legal articles dataset

The first Transformer-based model applied in this research is the BERTje model trained on the references to legal articles dataset (second column of Table 6). In Appendix D in Section 11.1 a detailed overview of the training process is given. The duration of the training process is 5 hours and 50 minutes on a graphics processing unit device (NVIDIA Tesla T4, 15 GB RAM). The increasing accuracy at each epoch, decreasing training loss and decreasing learning rate (see Figures 15, 16 and 17 in Section 11.1) indicate that the model is improving during the training process and moving towards convergence. After this process the trained model is applied on the test set to analyse the performance. The model achieves 61% accuracy on the test set. In Figure 10 the classifications for each case-law article in the test set are visualised in the format of a confusion matrix. In Appendix E in Section 12.1 a detailed classification report for each individual sub-area of taxation is given as well. The model classifies the case-law articles in the test set within 15 minutes on a local device. This process can be speed up more than 10 times on a graphics processing unit (GPU) device.

The results show the problem of the unbalanced training dataset. In the references to legal articles training dataset the categories 'Transfer tax', 'Allowances', 'Various taxes', 'Customs tax', 'International tax law' and 'Inheritance and gift tax' are underrepresented. From these categories the model only achieves a high accuracy percentage on 'Allowances' and 'Inheritance and gift tax'. For the other four categories the performance of the model is moderate to poor, as is shown in Figure 10 and in the classification report (Table 20) in Section 12.1 in Appendix E. This indicates the importance of a balanced training dataset. The model needs to process sufficient examples for each sub-category during training to be able to make successful classifications. In the next section the results for the RobBERT Transformer model are described, which is trained on the same training dataset.

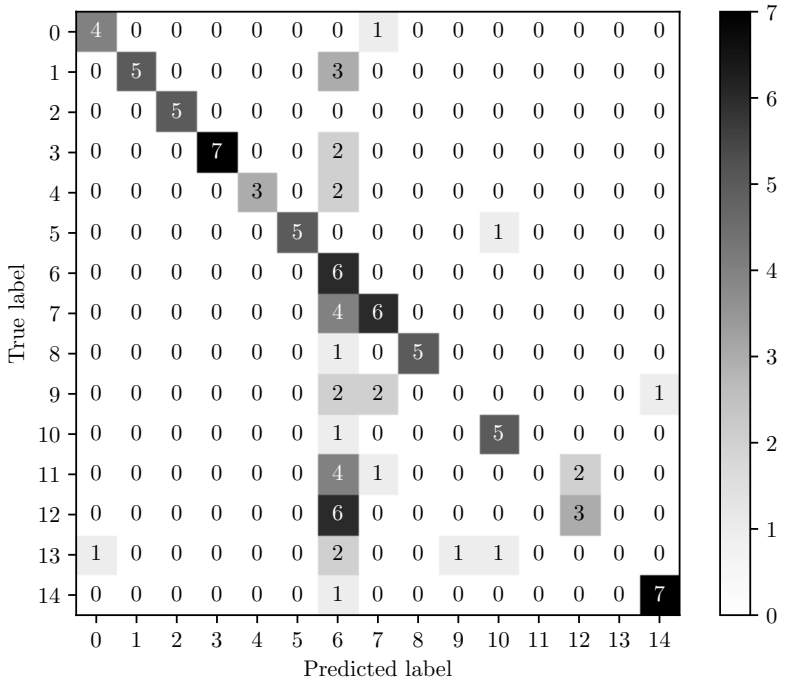

Figure 10: Confusion matrix for BERTje - references to legal articles dataset.

5.4 Results for RobBERT trained on references to legal articles dataset

The next Transformer-based method is the RobBERT model trained on the references to legal articles dataset. In Appendix D in Section 11.2 a detailed overview of the training process is given. The duration of the training process is 5 hours and 44 minutes. The increasing accuracy at each epoch, decreasing training loss and decreasing learning rate (see Figure 19, 20 and 21 in Section 11.2) again indicate that the model is improving during the training process and moving towards convergence. The trained model achieves 69% accuracy on the test set. In Figure 11 a confusion matrix of the classifications on the test set is given and in Appendix E in Section 12.2 a detailed classification report for this model is prepared. The confusion matrix and the classification report describe the specific classifications for each sub-area of taxation and provide the opportunity to analyse in which category the most mistakes are made.

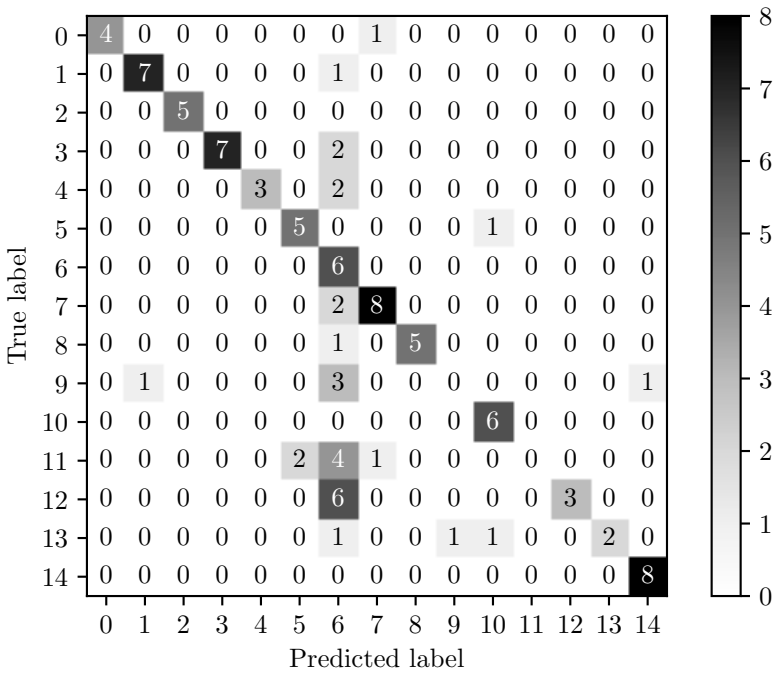

Figure 11: Confusion matrix for RobBERT - references to legal articles dataset.

Despite that this model is trained on the same unbalanced dataset it achieves better performance than the BERTje model. The model still performs the worst on the categories that are underrepresented in the training set as is shown in Table 21 in Section 12.2 (Appendix E). The recall rate (proportion of correct classifications in relation to the actual number of articles in a specific category) for the underrepresented categories 'International tax law', 'Transfer tax' and 'Various taxes' is low, as well as the recall rate for the 'Turnover/Sales' tax category. It is interesting to note that the model classifies many articles incorrectly as 'General state taxes and administrative law' (see Figure 11: label number 6). This specific category is overrepresented in the training dataset, where more than half of the case-law articles belong to this sub-area of taxation. It is therefore explicable that the model classifies 28 case-law articles from the test set to this category, while only 6 articles actually belong to this category. This explains the low precision rate (proportion of total predicted classifications for a specific category that was actually correct) of 21% for the 'General state taxes and administrative law' sub-area of taxation.

In the next two sections the results for the Transformer-based models trained on the contextual keywords dataset are described, which is a better balanced training dataset.

5.5 Results for BERTje trained on contextual keywords dataset

The third Transformer-based model is the BERTje model trained on the contextual keywords dataset. As is shown in Table 6 the contextual keywords dataset is more balanced than the references to legal articles dataset (i.e., the case-law articles are more equally distributed over all sub-areas of taxation). This implies that sub-areas of taxation are less under or overrepresented. In Appendix D in Section 11.3 an overview of the training process and detailed model specifications is given. The training time for this model is 5 hours and 58 minutes. The increasing accuracy at each epoch, decreasing training loss and decreasing learning rate (see Figure 23, 24 and 25 in Section 11.3) show that this model improves performance during training and moves towards convergence. After training the model is applied to classify the case-law articles in the test set and achieves 86% accuracy. In Figure 12 a confusion matrix is prepared. In Appendix E in Section 12.3 the classification report for this specific model is produced. These two reports provide more insight into the classifications of the individual sub-areas of taxation.

It is clear that the BERTje model trained on the contextual keywords dataset achieves higher classification accuracy on the unseen case-law articles in the test set than the previous Transformer-based models that are trained on the less-balanced references to legal articles dataset. In Figure 12 this can be seen from the fact that there are many high values on the diagonal. Six of the fifteen sub-areas of taxation are identified completely correct. The model performs the worst on the 'International tax law' and 'General state taxes and administrative law' sub-areas of taxation. A possible explanation for this is that these two categories are relatively widespread (i.e., have a wide range of subjects) and often combined with other sub-areas of taxation within a case-law article. Except for these two categories all sub-areas of taxation have a recall rate of 80% or higher (see Table 22 in Section 12.3). The better balanced contextual keywords dataset where the model trains on clearly improves the performance of this model.

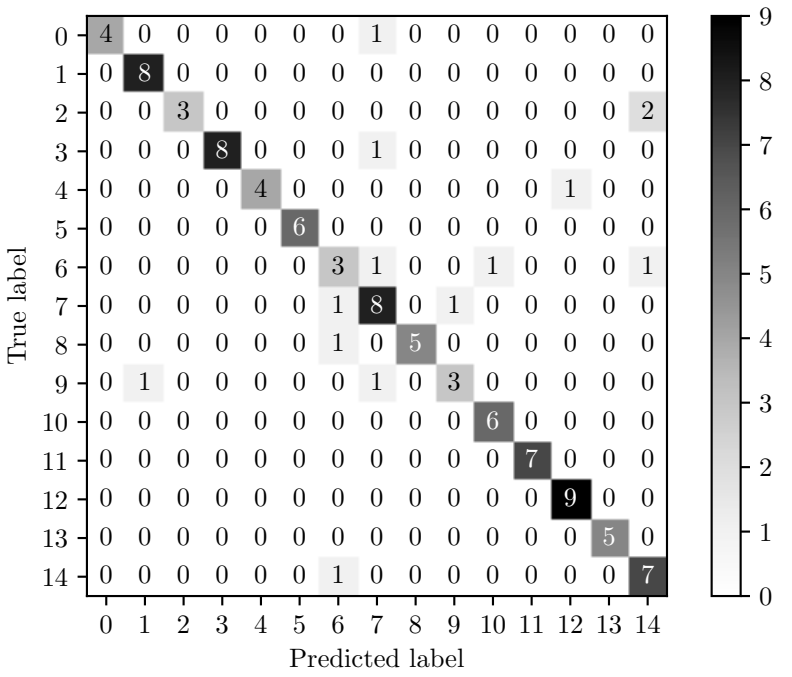

Figure 12: Confusion matrix for BERTje - contextual keywords dataset.

5.6 Results for RobBERT trained on contextual keywords dataset

The last Transformer-based model applied in this research is the RobBERT model which is trained on the contextual keywords dataset. The RobBERT model achieves higher accuracy on the test set than the BERTje model when they are both trained on the references to legal articles dataset. It is therefore interesting to see if this model also achieves better results when trained on the contextual keywords dataset. In Appendix D in Section 11.4 a detailed overview of the training process, which has a duration of 6 hours and 17 minutes, is given. The model improves during the training process and moves towards convergence, given the fact that the accuracy increases at each epoch, the training loss decreases and the learning rate decreases at each step (see Figure 27, 28 and 29 in Section 11.4). The RobBERT model achieves 87% accuracy on the test set. In Figure 13 a confusion matrix for the classifications on the test set is produced and in Appendix E in Section 12.4 a classification report for this model is prepared to provide

detailed insights for the classifications for each individual sub-area of taxation.

The RobBERT model trained on the contextual keywords dataset achieves a high performance on the test set, just like the BERTje model trained on the same dataset. The importance of a relatively balanced training dataset is hereby made clear for this classification task. In the confusion matrix in Figure 13 and in the classification report in Table 23 in Section 12.4 (Appendix E) it is shown that the model classifies six sub-areas of taxation completely correct. The model performs again the worst for the 'International tax law' (recall rate of 60%) and 'General state taxes and administrative law' (recall rate of 83%, but precision rate of 62%) categories. The recall and precision rates for each individual sub-area of taxation are never lower than 60%. With more training data and an even better balanced training dataset performance is likely to increase further.

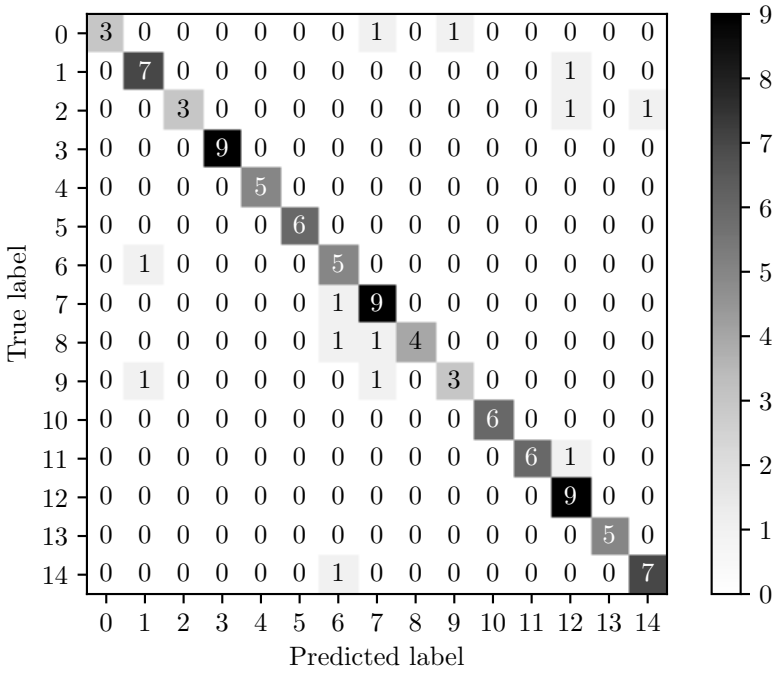

Figure 13: Confusion matrix for RobBERT - contextual keywords dataset.

5.7 Method comparison

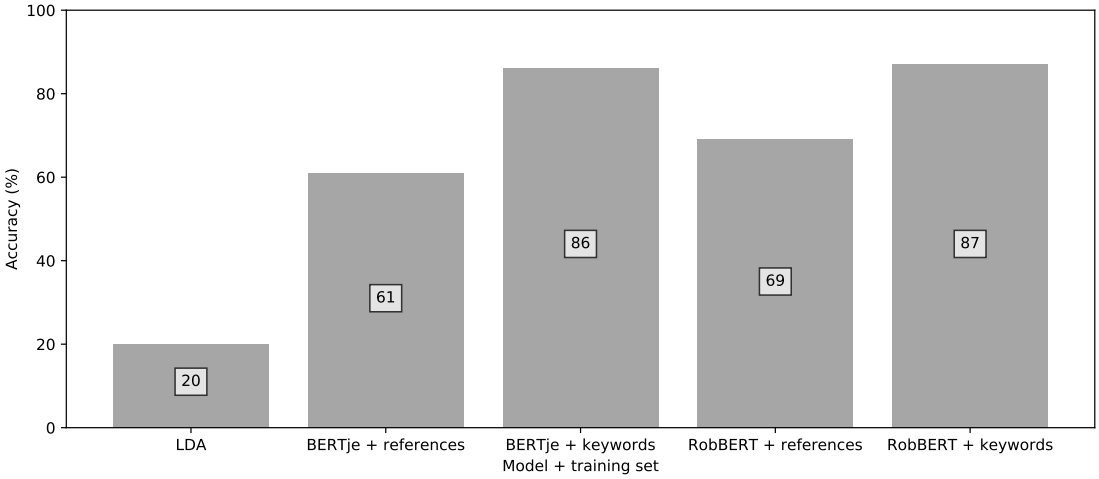

Figure 14: Method comparison on test set.

In the previous paragraphs of this section the results of the five applied methods in this research are described. Firstly, it is clear that the unsupervised topic modelling method (Latent Dirichlet Allocation, or LDA) does not create high-quality topics that correspond with the 15 sub-areas of taxation in which the case-law articles are classified. Some sub-areas of taxation are covered in multiple topics, whereas some topics do not cover any sub-area of taxation at all or are ambiguous. Secondly, the Transformer-based methods that are trained on the references to legal articles dataset only achieve a moderate performance. This is mainly because of the fact that this training dataset is highly unbalanced, with some sub-areas of taxation being underrepresented and some being overrepresented.

The RobBERT model performs better than the BERTje model when both trained on the same training dataset. This confirms the findings in existing literature that RoBERTa-based Transformer models achieve better results on Natural Language Processing tasks than BERT-based Transformer models (see Section 2.2.10).[8] Finally, the Transformer-based models trained on the more balanced contextual keywords dataset achieve the

[8]See also: (Super)Glue Leaderboard - https://gluebenchmark.com/leaderboard.

best performance as is visualised in Figure 14 above. The RobBERT model achieves the highest accuracy on the test set (87%), but there is no significant difference between this model and the BERTje model trained on the contextual keywords dataset (86% accuracy). A larger test set may clarify any significant differences between these two models.

6 Conclusion

In this research a method of classifying (legal) documents is proposed. The starting point of this new method is a fully unlabeled dataset consisting of fiscal case-law articles. By applying a method that searches for specific characteristics of a document a part of the case-law articles is labeled. This is the first step of the proposed method. The subset of labeled case-law articles created by the first step can be used as a training dataset for supervised Transformer-based models. This is the second step of the proposed method. In this section the main results are summarised as an answer to the research question. After that, the implications for the literature and to business practitioners (specifically in the field of marketing) are mentioned. Finally, the limitations of this research are given. Recommendations for further research are given based on the limitations and insights of this study.

6.1 Main results

The question *"Which combination of methods is most successful in classifying Dutch fiscal case-law articles into type of taxation?"* was the focus of this research. The first step of the methods applied in this research is to search for specific contextual keywords or references to legal articles within a case-law article. The second step is to train a Dutch pre-trained Transformer-based model (BERTje or RobBERT) on the datasets created by the methods in the first step. The BERTje and RobBERT models are already pre-trained on big amounts of Dutch text such that they are familiar with frequently occurring combinations of words and thus with common language structures before they are used in this research. The quality of the four combinations of methods (two different methods in the first step and two different models in the second step) is evaluated by applying the trained models on the independent test set consisting of 100 manually labeled case-law articles.

The best performing proposed method is the combination of searching for specific contextual keywords to create labels for a part of the dataset and then train the RobBERT

Transformer-based model on this subset. This method classifies 87% of the case-law articles in the test set correctly. The main reason why this method achieves high accuracy is the relatively balanced training set. Each category (i.e., sub-area of taxation) is sufficiently covered (i.e., consists of a sufficient number of case-law articles) in the contextual keywords dataset, such that the model is able to learn important features about each sub-area of taxation during training. The results in this research show the importance of a balanced training dataset. The searching for references to legal articles training dataset is less balanced and methods trained on this dataset therefore achieve poorer results. The RobBERT model achieves higher performance than the BERTje model when trained on the same dataset. This is in line with contemporary literature. Still, the result of the best performing RobBERT model (87%, trained on the contextual keywords dataset) is only one percentage point higher on the test set than the best performing BERTje model (86%, also trained on the contextual keywords dataset). A larger test set may lead to further clarification of this difference.

Overall, the proposed method structure that first creates labels for a part of the data and then trains a Transformer-based model on it to label the rest of the data is a promising research method for three main reasons. Firstly, no manual labeling efforts (except for evaluation purposes) have been necessary for this method structure. Secondly, the method structure is highly flexible. In this research the categories are the sub-areas of the Dutch tax system. However, the same research structure can easily be applied on other document classification tasks. The only requirement is the presence of specific contextual keywords that characterise the categories to be formed. Finally, this research structure achieves significantly better results than the unsupervised topic modelling method of Latent Dirichlet Allocation. Latent Dirichlet Allocation only achieves 20% accuracy on the test set, whereas the four combinations of methods of the proposed method structure achieve a minimum of 61% and a maximum of 87% (which can probably be increased through more training data).

6.2 Implications for the literature

In the previous paragraph the strength of the proposed method structure is explained. This is the first main contribution to the existing literature. This research shows that the previously described method structure achieves better results than the unsupervised topic modelling method of Latent Dirichlet Allocation. Transformer-based models achieve high results on the Natural Language Processing task in this research (i.e., document classification), which is in line with current findings in the literature. This research demonstrates that the Transformer-based models are also successful when the training dataset is created through a relatively simple and automated approach (i.e., by just looking at specific contextual keywords) instead of manual labeling.

The second main contribution to the literature is the comparison between Dutch pre-trained Transformer-based models in the field of document classification. The language used in the documents in this research is relatively complex and domain-specific. The Dutch Transformer-based models BERTje and RobBERT are as far as this author knows never been compared before in the field of document classification. Only the English or multilingual Transformer-based models BERT and RoBERTa are compared extensively. This research was carried out in such a way that the Dutch Transformer-based models can be optimally compared with each other. All parameters are kept as constant as possible as well as the training instructions and characteristics for all applied models. The comparison of results between the Dutch Transformer models (RobBERT performs better than BERTje) is in line with the comparison between the English or multilingual Transformer models (RoBERTa performs better than BERT) in the literature.

6.3 Implications for practitioners

This research shows that it is possible to classify fiscal case-law articles successfully by applying the proposed method structure. An implication from a business perspective is the possibility to easily add extra filters in a search engine. In time Bluetick (the company for which this research is conducted) will include the results and the proposed method structure of this research in their search engine to add an extra filter with sub-areas of

taxation. Big legal literature publishers currently hire several working students to classify all newly published literature into desired categories. The proposed method structure can be a useful, accurate and quick alternative to reduce these labour costs.

In other domains document classification can also be used in, amongst others, classifying news articles, detecting spam messages or evaluating cover letters. The proposed research structure is highly flexible, achieves promising results and is not time-consuming because it requires no manual labeling efforts. These characteristics make it attractive for all kinds of companies to use this structure with the associated Transformer-based models to perform document classification. The models also give estimated probabilities for each category for a classified document, such that the end user can see how certain the model is of a particular classification and between which categories this uncertainty plays.

An example of the application of the proposed method structure in the field of marketing is to classify customers into categories (i.e., customer profiles) in an innovative way. Think of a company that wants to classify its users into specific profile types based on a personal description written by the user. This can be the case, for example, for a dating application where each new user must provide a description of him/herself and his/her preferences. Through the proposed method structure such a company can classify its users into specific customer profiles. These customer profiles can subsequently be used for personal content generation, personalised offers (in the context of a dating application: similar people from a relevant customer profile) and other types of personalisation (e.g., customised homepages). Other examples where the proposed method structure can be applied in a marketing context are for the classification of customer complaints, product reviews and strategies of competitors (by analysing the text from advertisements).

6.4 Limitations

There are four major limitations to this research. First, the goal of document classification is to classify documents into desired categories as accurate as possible. The training parameters and structure in this research is kept the same for all Transformer models

in order to keep comparability as high as possible. However, if one wants to classify the fiscal case-law articles with the highest accuracy the training process likely needs to be adjusted. In this research the training procedure of the Transformer models is implemented with default parameter values. The effect of changes in parameter values such as batch size and learning rate is therefore unknown, but this will likely result in better results since there are many other combinations of values. Besides, the training dataset size can be increased or the training dataset can be made more balanced (for example via stratified sampling, which samples the dataset so that categories are of the same size) to achieve better results. The effect of changes to the (structure of) the training dataset is mostly unknown. It is also very likely that performance will increase when the Transformer-based models are pre-trained further on the Dutch legal texts from this research. Legal texts contain domain-specific language which is not often used in other texts. The applied Transformer-based models in this research are only pre-trained on Dutch Wikipedia text, so further pre-training on legal texts could boost the performance.

The second major limitation in this research is that only the first part (i.e., the first 2048 tokens) of a case-law article is processed by the Transformer-based models, due to restrictions in computational power. The choice for only processing the first part of a case-law article is arbitrary. It may well be the case that processing only the middle part or the last part of a case-law article leads to improved results. Another option is to split a case-law article in pieces of 2048 tokens, separately classify each piece with a Transformer-based model and average the classifications for each case-law article. Results probably increase as well when more tokens are being processed by the models (the models can process up to 4096 tokens if computational power allows it). Furthermore, the choice for labeling a fiscal case-law article to a sub-area of taxation when a specific contextual keyword or a specific reference to a legal article occurs more than three times is also arbitrary. The number three can easily be adapted and can have consequences for the results. Still, it was outside the scope of this research to adjust these arbitrary choices and check the performances of the methods accordingly.

The third limitation is that this research does not show the relation between the quality of the prepared contextual keywords for each sub-area of taxation and the performance of the methods. The contextual keywords (and especially the exclusivity to one particular category) determine the quality of the training dataset where the Transformer-based models train on. The labels derived from the occurrence of contextual keywords in case-law articles need to be of a high quality (i.e., as correct as possible), otherwise the Transformer-based models learn false or ambiguous information during training. The contextual keywords are self-created, so they need to be created by someone with domain-specific knowledge. The quality of the contextual keywords and the relation with the performance of the methods is not covered in this research.

The last limitation of this research is the structure and size of the test set. The test set only contains 100 manually labeled case-law articles. Given the number of categories in this research (15) this is a small size. Therefore, it is harder to evaluate the methods with respect to determining whether a particular method is significantly better than another, especially if the difference in results is small. This problem can be solved by increasing the size of the test set. However, this is infeasible in the given time frame of this research. In addition, the test set is created in a way that all categories are sufficiently represented. Therefore it is not completely random which case-law article is in the test set. Nevertheless, the choice was made to create a balanced test set so that statements can be made about individual sub-areas of taxation. If the test set was completely randomly created it would result in some sub-areas of taxation being underrepresented or not present at all, because particular sub-areas of taxation are much more common than others.

6.5 Future work

As described in the previous sections results can certainly be improved. Performance can be boosted via adjustments in the training parameters, method structure and in the Transformer-based models (e.g., by pre-training them further on legal documents). Results can also be improved through increasing the training dataset, making the

training dataset better balanced or through processing more tokens in a document. This, however, is the rough way of boosting performance because more training data generally equals higher results. A recommendation for future work therefore is to change the hyperparameters during the training process, processing other parts of the document (or by splitting a document into parts and averaging classifications for all parts) and by applying pre-training on Dutch legal texts for the Transformer-based models. These gentle adjustments can increase the quality of the document classification without the need of more computational power or longer training time with more data.

In future work the relation between the quality of contextual keywords for each category and the performance of the methods needs to be established quantitatively. For new document classification tasks outside the Dutch fiscal domain it is important to be informed about the optimal number of contextual keywords per category and the properties (e.g., exclusivity for a single category, frequency of occurring within a category) that a keyword must meet to achieve high performance. In other words, it is important to know how many times a contextual keyword should appear in a specific category, how important the exclusivity of that keyword is for the specific category, and whether creating more keywords per category is always better.

Other interesting directions for future research are twofold. On the one hand it is interesting to compare other Transformer-based models against the Dutch Transformer-based models applied in this research. It can be interesting to see for example how the newer Text-To-Text Transfer Transformer (T5) performs on the method structure used in this research, since the authors of that model claim it achieves state-of-the-art results on text classification tasks.[9] Other multilingual Transformer-based models can be applied and compared as well. On the other hand it can be interesting to conduct a study on the output of the Transformer-based model. For each document the model gives probabilities that it belongs to a particular category. The analysis of these probabilities is important to determine whether the model is able to estimate its own uncertainty. A model is more powerful if it can realistically indicate with how much certainty a classification is made.

[9]See Raffel et al. (2019) for the Text-To-Text Transfer Transformer (T5) paper.

In other words, the probability for a selected category ideally needs to be lower for false classifications than for correct classifications. This makes the use of such a model in practice even more attractive because it is possible to estimate when the model is likely to make a mistake.

All in all, it can be concluded that the proposed method structure and the use of a Transformer-based model in this research is a solid foundation for document classification tasks. The first step is to create contextual keywords for each category, such that a training dataset is prepared. A Transformer-based model, in this research the Dutch RobBERT model, is then trained on the dataset and later applied on unseen documents to perform classification. The most effective method in the context of classifying Dutch fiscal case-law articles achieves 87% accuracy. The method structure is highly flexible and can be applied in other domains like marketing. It is also fast because no manual labeling efforts are required. More work should be invested in adjusting key hyperparameters, pre-training on texts which are comparable to the texts that need to be classified, changing the structure of the training process and processing other parts of the documents to analyse the change in performance. Several other Transformer-based models which are not applied in this research also appear to be promising according to the literature.

7 Bibliography

Bafna, P., Pramod, D., and Vaidya, A. (2016). Document clustering: Tf-idf approach. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 61–66. IEEE.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Berger, A., Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Gillett, J. R., Lafferty, J., Mercer, R. L., Printz, H., and Ures, L. (1994). The candide system for machine translation. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Berger, A., Della Pietra, S. A., and Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of Machine Learning research*, 3:993–1022.

Buchta, C., Kober, M., Feinerer, I., and Hornik, K. (2012). Spherical k-means clustering. *Journal of statistical software*, 50(10):1–22.

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., and Nissim, M. (2019). Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H., and Ureš, L. (1994). Inference and estimation of a long-range trigram model. In *International Colloquium on Grammatical Inference*, pages 78–92. Springer.

Delobelle, P., Winters, T., and Berendt, B. (2020). Robbert: a dutch roberta-based language model. *arXiv preprint arXiv:2001.06286*.

Deng, L. and Liu, Y. (2018). A joint introduction to natural language processing and to deep learning. In *Deep learning in natural language processing*, pages 1–22. Springer.

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

Foote, K. D. (2019). A Brief History of Natural Language Processing (NLP). https://www.dataversity.net/a-brief-history-of-natural-language-processing-nlp/. [Online; Accessed 01-05-2021].

Giles, C. L., Lawrence, S., and Tsoi, A. C. (1997). Rule inference for financial prediction using recurrent neural networks. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pages 253–259. IEEE.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. (2018). Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651.

Jing, L. and Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Lafferty, J., Sleator, D., and Temperley, D. (1992). *Grammatical trigrams: A probabilistic model of link grammar*, volume 56. School of Computer Science, Carnegie Mellon University.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

Lewis, D. D. and Jones, K. S. (1996). Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101.

Li, S., Wunsch, D. C., O'Hair, E. A., and Giesselmann, M. G. (2001). Using neural networks to estimate wind turbine power generation. *IEEE Transactions on energy conversion*, 16(3):276–282.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Medsker, L. and Jain, L. C. (1999). *Recurrent neural networks: design and applications*. CRC press.

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

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Moore, R. C. (1981). Practical natural-language processing by computer. Technical report, Sri International Menlo Park CA Artificial Intelligence Center.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

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

Pradeepkumar, D. and Ravi, V. (2017). Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. *Applied Soft Computing*, 58:35–52.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Smeaton, A. F. (1999). Using nlp or nlp resources for information retrieval tasks. In *Natural language information retrieval*, pages 99–111. Springer.

Socher, R., Bengio, Y., and Manning, C. D. (2012). Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pages 5–5.

Steinbach, M., Ertöz, L., and Kumar, V. (2004). The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer.

Steyvers, M., Smyth, P., Rosen-Zvi, M., and Griffiths, T. (2004). Probabilistic author-topic models for information discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315.

Tao, Y., Takagi, K., and Nakata, K. (2021). Clustering-friendly representation learning via instance discrimination and feature decorrelation. pages 1–15.

Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European conference on machine learning*, pages 491–502. Springer.

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032*.

van Opijnen, M., Verwer, N., and Meijer, J. (2015). Beyond the experiment: the extendable legal link extractor. In *Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts, held in conjunction with the 2015 International Conference on Artificial Intelligence and Law (ICAIL)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vig, J. (2019). A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wu, H., Zhao, H., and Zhang, M. (2021). Not all attention is all you need. *arXiv preprint arXiv:2104.04692*.

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

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 13(3):55–75.

8 Appendix A - Self-attention

In this section the self-attention mechanism that Transformer-based models use is described in detail. Since the way Transformers use self-attention to process input sequences is key to the model's performance, it is useful to take a closer look at how this is done intuitively and mathematically. The self-attention mechanism is partially comparable to how a recurrent neural network uses the previous hidden state for the processing of the current input. First, three matrices are created: a Query matrix $\boldsymbol{Q}$, Key matrix $\boldsymbol{K}$ and Value matrix $\boldsymbol{V}$ (Vaswani et al., 2017, p. 4). There is also an embedding matrix $\boldsymbol{X}$ that contains the embedding of every word in the input sequence in its rows. Matrices $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are calculated respectively by multiplying matrix $\boldsymbol{X}$ by the weight matrices $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$ and $\boldsymbol{W_V}$ that are retrieved during the training-phase of the model. Matrices $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are of the same size having one row for each word and mostly have lower dimensions (columns) than the embedding matrix $\boldsymbol{X}$. So, if $\boldsymbol{X}$ is a $m * r_x$ matrix with m words represented in r_x dimensions, $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are $m * r_y$ matrices where $r_y \leq r_x$.

Secondly, for every word in an input sequence a score is calculated that represents how much attention is targeted to other words in that sequence (Vaswani et al., 2017, p. 4). This score is calculated by multiplying the Query matrix and the transpose of the Key matrix (transpose, because otherwise matrix multiplication is not possible) for a specific word. So, when calculating the attention for the first word of the input sequence on the second word the dot product is taken from the first row of the Query matrix and the second column of the Key matrix. Here it becomes clear that the rows of the Query matrix represent the words for which the self-attention is calculated (i.e., focal word) and the rows of the Key matrix represent the words on which positions the focal word is expressed (i.e., where the attention of the focal word goes to/which other words contribute to the description of the focal word). This score is divided by the square root of the dimensions of the Key matrix r_y to make the gradients better balanced, which is also shown in Equation 1 below (Vaswani et al., 2017, p. 4). From this score a softmax

transformation is executed to let all scores be positive and add up to 1. Softmax scores closer to 1 mean these words are relevant in the self-attention mechanism. After this, the softmax scores for each word in the input sequence are multiplied by the corresponding rows of the Value matrix. Irrelevant words with low softmax scores are thereby multiplied by a small number in order to fade away their Value vectors. Subsequently, all scores concerning the focal word are summed up. The result is a vector for each word which can be placed in a matrix $\boldsymbol{S}$ that contains the self-attention scores for every word in the input sequence. This is the output of a self-attention layer that is transmitted to the feed forward layer. In summary, the formula for the full self-attention matrix looks like this:

$$\boldsymbol{S} = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{r_y}})\boldsymbol{V} \ . \tag{1}$$

Now that the general concept of self-attention is clear, the mechanism of multi-head attention can be defined (Vaswani et al., 2017, pp. 4-5). Instead of using a single attention function with just a single matrix $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$, the authors found it was better to linearly project h different learned matrices of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$. The self-attention function is then performed on each of the projected matrices (this can be done parallelized) resulting in h different $\boldsymbol{S}$ matrices (Vaswani et al., 2017, pp. 4-5). The matrices are then concatenated into one matrix and multiplied by weight matrix $\boldsymbol{W_S}$ that is trained jointly with the model. The outcome is transmitted to the feed-forward layer within the encoder. The feed-forward networks are present within each of the n encoders and decoders and execute two linear transformations and a linear activation function that outputs the input directly if it is positive and otherwise it will output a zero (Vaswani et al., 2017, p. 5). The multi-head attention mechanism makes sure the model is able to focus on all positions of the input sequence, regardless of the distance between the words (for which a recurrent neural network is limited). Besides, it gives the model the possibility to use multiple representation spaces to define self-attention (because it uses multiple trained matrices) (Vig, 2019, p. 4).[10]

[10]See also: Llion Jones. 2017. Tensor2tensor transformer visualization. https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/visualization.

A decoder works in a similar way as an encoder, however it has one extra sub-layer at the beginning. This layer performs masked multi-head attention over the output of the encoders (Vaswani et al., 2017, p. 3). In this way, the decoder is not able to see subsequent positions of the input sequence when processing a certain word. The multi-head attention layer or 'encoder-decoder attention' layer of a decoder is slightly different from the multi-head attention layer in the encoder, because the matrices $\boldsymbol{K}$ and $\boldsymbol{V}$ used in calculating the self-attention come from the output of the encoder (Vaswani et al., 2017, p. 5). Matrix $\boldsymbol{Q}$ comes from the previous layer just like in the process of the encoders. So, the masked multi-head attention mechanism works by looking at the matrices $\boldsymbol{K}$ and $\boldsymbol{V}$ obtained from the top encoder.

9 Appendix B - Metadata

In this section an overview of the metadata for the case-law articles is given in Table 9.

Table 9: Metadata for case-law articles.

Variable	**Description**
Identifier (ECLI)	ECLI case number*
Format	Format of the document (text/xml)
Access Rights	Indicates if the document is publicly available
Issued	Time the document enters the database
Modified	Time the document is modified in the database
Publisher	Organisation that publishes the document
Language	Language of the document
Creator	Court dealing with the case
Date	Time of judgment of the case
Type	The type of court case
Coverage	Country concerned by the case
Subject	Area of law of the case**

*Each case-law article corresponds with a unique identifier named an ECLI number. This ECLI number has the following structure:

1. The abbreviation ECLI;
2. A country code, NL in all cases for this research;
3. A court code, depending on which court deals with the case;
4. Year of judgment of the case;
5. A unique identification number.

An example of an ECLI number is "ECLI:NL:GHARN:1949:5".

** The subject of the case-law articles for this research is Tax Law.

10 Appendix C - Latent Dirichlet Allocation details

In this section the details for the Latent Dirichlet Allocation method are described.

Table 10: General fiscal words deleted from the data.

ECLI	Verweerders	Verweerder
NTFR	Eiser	Eiseres
Eisers	Arrest	Rechtbank

10.1 Train details Latent Dirichlet Allocation

- Number of topics: 15
- Number of workers (number of workers processes to be used for parallelization): 2
- Number of passes (number of passes through the full corpus of documents during training, similar to epochs for Transformer models): 2
- Chunksize (number of documents used in each training chunk): 2000
- Alpha (α) (parameter that defines the document-topic distribution): 1/15
- Eta (η) (parameter that defines the topic-word distribution): 1/15
- Decay (value that defines what percentage of information is forgotten when a new document is examined): 0.5
- Offset (value that controls the slow down of the first steps for the first few iterations): 1
- Evaluation per update (how many times perplexity is estimated per update): 10
- Number of iterations (Maximum number of iterations through the corpus when inferring the topic distribution for new documents): 50
- Random state (useful for reproducibility): 1

10.2 Results Latent Dirichlet Allocation

Table 11: Classification report for Latent Dirichlet Allocation method.

	precision	recall	f1-score	support
Allowances	0.00	0.00	0.00	5
Car taxes	0.62	0.62	0.62	8
Collection act	0.00	0.00	0.00	5
Corporate and dividend tax	0.00	0.00	0.00	9
Customs tax	0.00	0.00	0.00	5
Environmental taxes	0.00	0.00	0.00	6
General state taxes and administrative law	0.00	0.00	0.00	6
Income tax	0.12	0.10	0.11	10
Inheritance and gift tax	0.00	0.00	0.00	6
International tax law	0.00	0.00	0.00	5
Local government taxes	0.29	0.83	0.43	6
Transfer Tax (Legal Transactions Tax Act)	0.00	0.00	0.00	7
Turnover/Sales tax	0.14	1.00	0.24	9
Various taxes (gambling tax, bank tax, landlord...	0.00	0.00	0.00	5
Wage/Payroll tax	0.00	0.00	0.00	8
Accuracy			0.20	100
Macro avg	0.08	0.17	0.09	100
Weighted avg	0.09	0.20	0.11	100

11 Appendix D - Specifications and training details for Transformer-based models

In this section the technical details of the specifications and training procedures of the BERTje and RobBERT models are given.

11.1 Technical details for BERTje trained on references to legal articles dataset

Table 12: Training details BERTje - references to legal articles.

Specification	**Value**
Training time	5h 50min
Epochs	3
Training dataset size	7681
Testing dataset size	1920
Train batch size	4
Test batch size	4
Warmup steps	200
Weight decay	0.01
Total optimization steps during training	5760
Accuracy on test set	0.839
Train loss	0.403
Test loss	0.676
Evaluation documents per second	2
Learning rate (start)	0.001
Learning rate (end)	1.17e-7
Maximum number of processed tokens	2048

Table 13: Model specifications BERTje - references to legal articles.

Specification	**Value**
Hidden layers	12
Hidden size	768
Maximum position embeddings	2048
Intermediate size	3072
Vocabulary size	30073
Attention window dropout probability	0.1
Output features	15

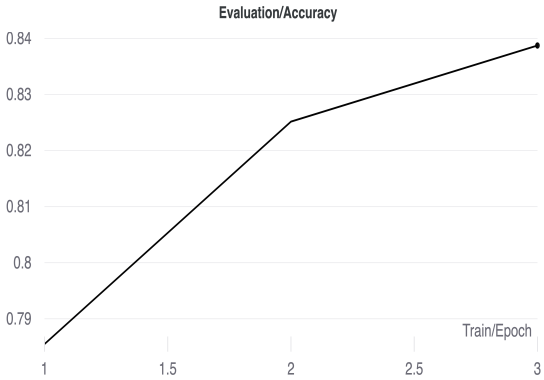

Figure 15: Accuracy per epoch on test set.

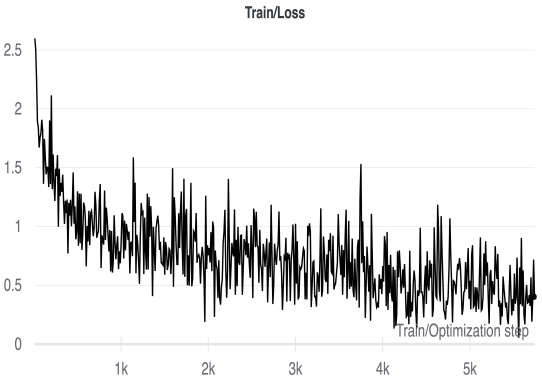

Figure 16: Train loss per optimization step.

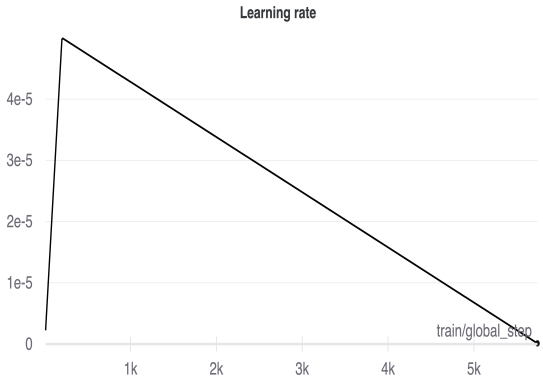

Figure 17: Learning rate during training.

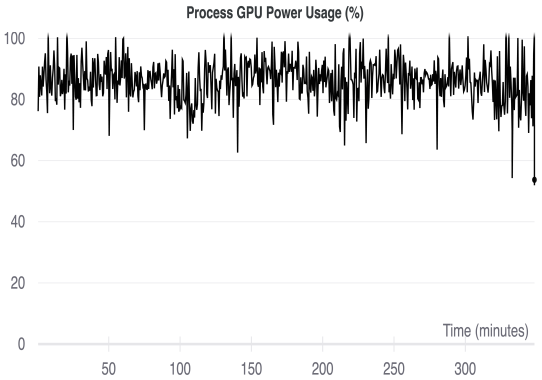

Figure 18: GPU percentage used.

11.2 Technical details for RobBERT trained on references to legal articles dataset

Table 14: Training details RobBERT - references to legal articles.

Specification	Value
Training time	5h 44min
Epochs	3
Training dataset size	7681
Testing dataset size	1920
Train batch size	4
Test batch size	4
Warmup steps	200
Weight decay	0.01
Total optimization steps during training	5760
Accuracy on test set	0.838
Train loss	0.204
Test loss	0.745
Evaluation documents per second	2
Learning rate (start)	0.001
Learning rate (end)	1.26e-7
Maximum number of processed tokens	2048

Table 15: Model specifications RobBERT - references to legal articles.

Specification	Value
Hidden layers	12
Hidden size	768
Maximum position embeddings	2048
Intermediate size	3072
Vocabulary size	40000
Attention window dropout probability	0.1
Output features	15

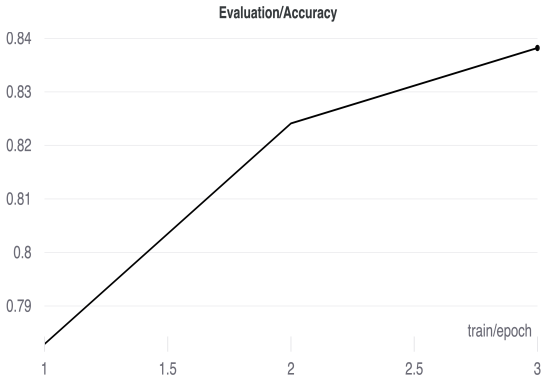

Figure 19: Accuracy per epoch on test set.

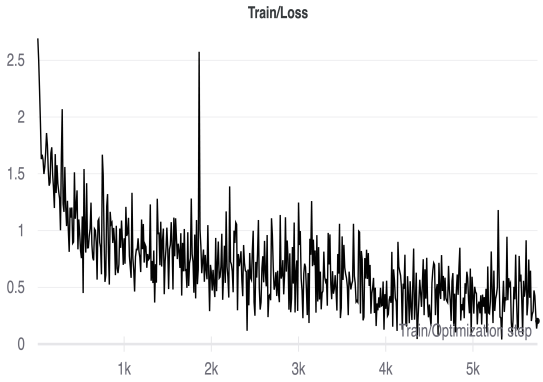

Figure 20: Train loss per optimization step.

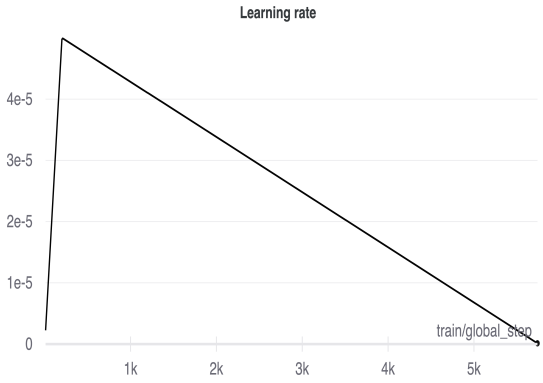

Figure 21: Learning rate during training.

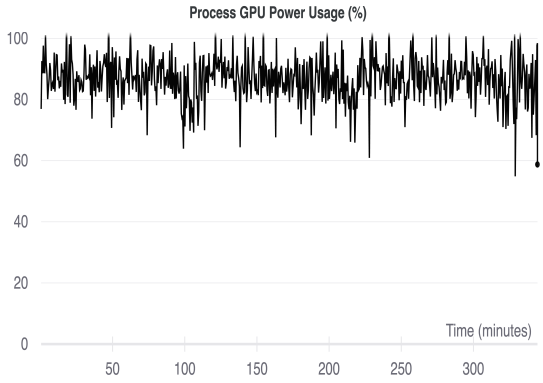

Figure 22: GPU percentage used.

11.3 Technical details for BERTje trained on contextual keywords dataset

Table 16: Training details BERTje - contextual keywords.

Specification	Value
Training time	5h 58min
Epochs	3
Training dataset size	7681
Testing dataset size	1920
Train batch size	4
Test batch size	4
Warmup steps	200
Weight decay	0.01
Total optimization steps during training	5760
Accuracy on test set	0.789
Train loss	0.438
Test loss	0.947
Evaluation documents per second	2
Learning rate (start)	0.001
Learning rate (end)	5.17e-8
Maximum number of processed tokens	2048

Table 17: Model specifications BERTje - contextual keywords.

Specification	Value
Hidden layers	12
Hidden size	768
Maximum position embeddings	2048
Intermediate size	3072
Vocabulary size	30073
Attention window dropout probability	0.1
Output features	15

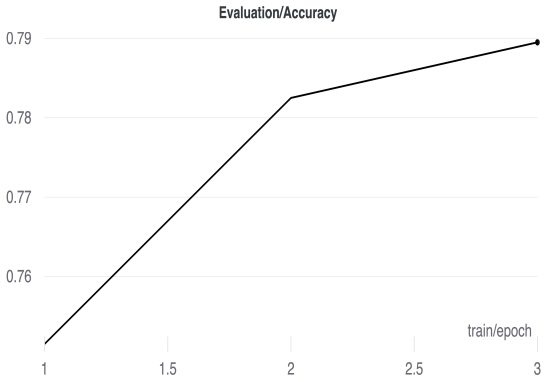

Figure 23: Accuracy per epoch on test set.

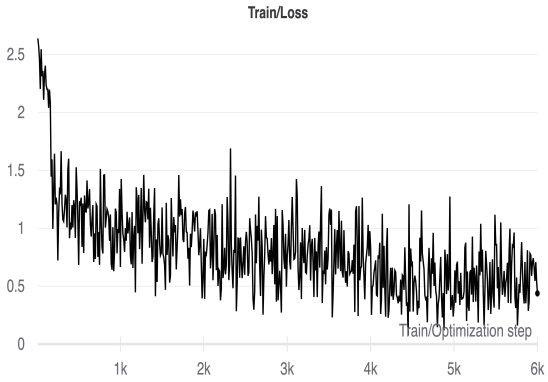

Figure 24: Train loss per optimization step.

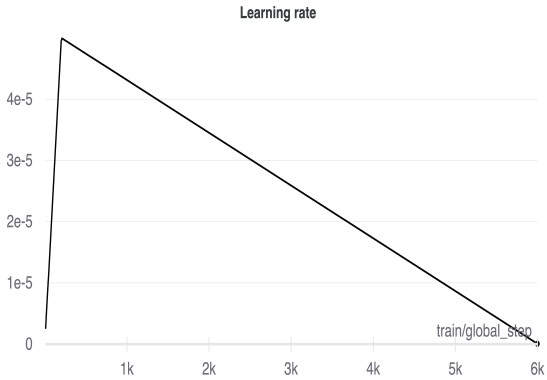

Figure 25: Learning rate during training.

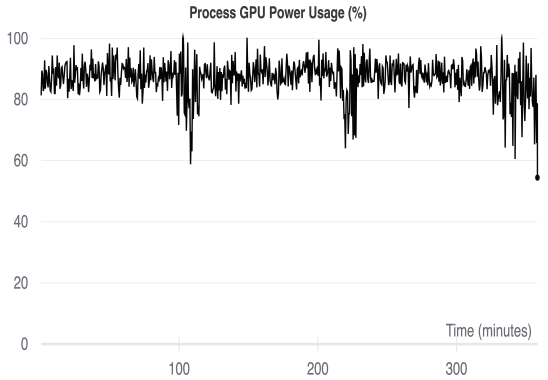

Figure 26: GPU percentage used.

11.4 Technical details for RobBERT trained on contextual keywords dataset

Table 18: Training details RobBERT - contextual keywords.

Specification	**Value**
Training time	6h 17min
Epochs	3
Training dataset size	7681
Testing dataset size	1920
Train batch size	4
Test batch size	4
Warmup steps	200
Weight decay	0.01
Total optimization steps during training	5760
Accuracy on test set	0.825
Train loss	0.508
Test loss	0.776
Evaluation documents per second	2
Learning rate (start)	0.001
Learning rate (end)	4.31e-8
Maximum number of processed tokens	2048

Table 19: Model specifications RobBERT - contextual keywords.

Specification	**Value**
Hidden layers	12
Hidden size	768
Maximum position embeddings	2048
Intermediate size	3072
Vocabulary size	40000
Attention window dropout probability	0.1
Output features	15

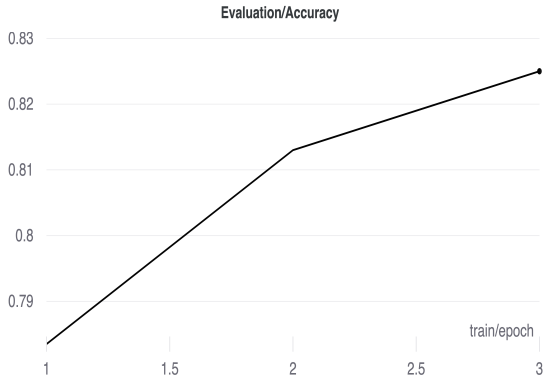

Figure 27: Accuracy per epoch on test set.

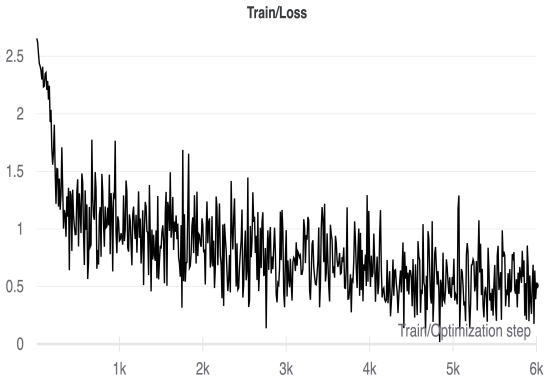

Figure 28: Train loss per optimization step.

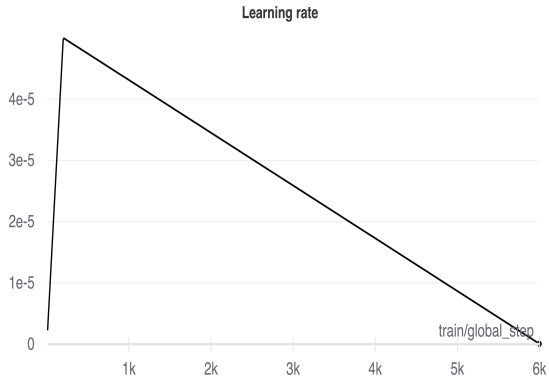

Figure 29: Learning rate during training.

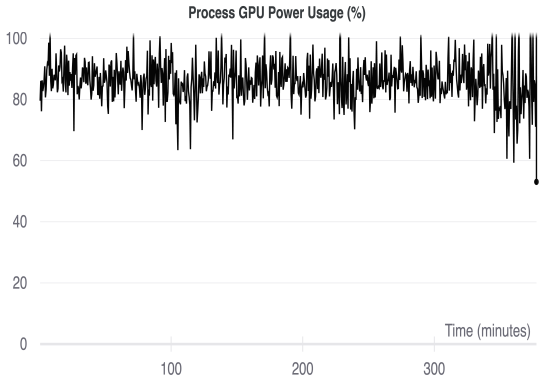

Figure 30: GPU percentage used.

12 Appendix E - Results for Transformer-based models

In this section the results for the Transformer-based methods are given in the format of detailed classification reports.

12.1 Results for BERTje trained on references to legal articles dataset

Table 20: Classification report for BERTje trained on references to legal articles dataset.

	precision	recall	f1-score	support
Allowances	0.80	0.80	0.80	5
Car taxes	1.00	0.62	0.77	8
Collection act	1.00	1.00	1.00	5
Corporate and dividend tax	1.00	0.78	0.88	9
Customs tax	1.00	0.60	0.75	5
Environmental taxes	1.00	0.83	0.91	6
General state taxes and administrative law	0.18	1.00	0.30	6
Income tax	0.60	0.60	0.60	10
Inheritance and gift tax	1.00	0.83	0.91	6
International tax law	0.00	0.00	0.00	5
Local government taxes	0.71	0.83	0.77	6
Transfer Tax (Legal Transactions Tax Act)	0.00	0.00	0.00	7
Turnover/Sales tax	0.60	0.33	0.43	9
Various taxes (gambling tax, bank tax, landlord...	0.00	0.00	0.00	5
Wage/Payroll tax	0.88	0.88	0.88	8
Accuracy			0.61	100
Macro avg	0.65	0.61	0.60	100
Weighted avg	0.67	0.61	0.61	100

12.2 Results for RobBERT trained on references to legal articles dataset

Table 21: Classification report for RobBERT trained on references to legal articles dataset.

	precision	recall	f1-score	support
Allowances	1.00	0.80	0.89	5
Car taxes	0.88	0.88	0.88	8
Collection act	1.00	1.00	1.00	5
Corporate and dividend tax	1.00	0.78	0.88	9
Customs tax	1.00	0.60	0.75	5
Environmental taxes	0.71	0.83	0.77	6
General state taxes and administrative law	0.21	1.00	0.35	6
Income tax	0.80	0.80	0.80	10
Inheritance and gift tax	1.00	0.83	0.91	6
International tax law	0.00	0.00	0.00	5
Local government taxes	0.75	1.00	0.86	6
Transfer Tax (Legal Transactions Tax Act)	0.00	0.00	0.00	7
Turnover/Sales tax	1.00	0.33	0.50	9
Various taxes (gambling tax, bank tax, landlord...	1.00	0.40	0.57	5
Wage/Payroll tax	0.89	1.00	0.94	8
Accuracy			0.69	100
Macro avg	0.75	0.68	0.67	100
Weighted avg	0.76	0.69	0.68	100

12.3 Results for BERTje trained on contextual keywords dataset

Table 22: Classification report for BERTje trained on contextual keywords dataset.

	precision	recall	f1-score	support
Allowances	1.00	0.80	0.89	5
Car taxes	0.89	1.00	0.94	8
Collection act	1.00	0.60	0.75	5
Corporate and dividend tax	1.00	0.89	0.94	9
Customs tax	1.00	0.80	0.89	5
Environmental taxes	1.00	1.00	1.00	6
General state taxes and administrative law	0.50	0.50	0.50	6
Income tax	0.67	0.80	0.73	10
Inheritance and gift tax	1.00	0.83	0.91	6
International tax law	0.75	0.60	0.67	5
Local government taxes	0.86	1.00	0.92	6
Transfer Tax (Legal Transactions Tax Act)	1.00	1.00	1.00	7
Turnover/Sales tax	0.90	1.00	0.95	9
Various taxes (gambling tax, bank tax, landlord...	1.00	1.00	1.00	5
Wage/Payroll tax	0.70	0.88	0.78	8
Accuracy			0.86	100
Macro avg	0.88	0.85	0.86	100
Weighted avg	0.87	0.86	0.86	100

12.4 Results for RobBERT trained on contextual keywords dataset

Table 23: Classification report for RobBERT trained on contextual keywords dataset.

	precision	recall	f1-score	support
Allowances	1.00	0.60	0.75	5
Car taxes	0.78	0.88	0.82	8
Collection act	1.00	0.60	0.75	5
Corporate and dividend tax	1.00	1.00	1.00	9
Customs tax	1.00	1.00	1.00	5
Environmental taxes	1.00	1.00	1.00	6
General state taxes and administrative law	0.62	0.83	0.71	6
Income tax	0.75	0.90	0.82	10
Inheritance and gift tax	1.00	0.67	0.80	6
International tax law	0.75	0.60	0.67	5
Local government taxes	1.00	1.00	1.00	6
Transfer Tax (Legal Transactions Tax Act)	1.00	0.86	0.92	7
Turnover/Sales tax	0.75	1.00	0.86	9
Various taxes (gambling tax, bank tax, landlord...	1.00	1.00	1.00	5
Wage/Payroll tax	0.88	0.88	0.88	8
Accuracy			0.87	100
Macro avg	0.90	0.85	0.87	100
Weighted avg	0.89	0.87	0.87	100