Hierarchical Segment Classification on Search Queries using Machine Learning

by

Nicole Schrieks (ANR: 595868)

A thesis submitted in partial fulfillment of the requirements for the degree of

Master in Econometrics and Mathematical Economics

Tilburg School of Economics and Management

Tilburg University

Supervised by:

prof. Dr. T. (Tobias) Klein — Tilburg University

Dr. C. (Christoph) Walsh — Second reader, Tilburg University

T. (Tim) Butterbrod, MSc. — Greenhouse Group

P. (Peer) van Kemenade, MSc. — Greenhouse Group

Date: March 30, 2023

Abstract

Hierarchical text classification is used to classify text into hierarchical categories. In search queries, hierarchical classification is useful to give organizations more helpful insight into interesting categories to focus on and where the potential is present. In recent years neural networks have been used often to hierarchically classify text data, which led to good results. This problem is challenging because of the hierarchy, but also because in this problem, the text contains little information, and the amount of data is limited. Two new loss functions are proposed to solve the problem in this context. The first loss function ensures that the model considers the hierarchy during training. The second loss function aims to minimize the mean absolute percentage error of search volume per segment; this function works as an addition to the first one. In addition, $BERT$-like models are used to improve the model's context understanding. Overall, the usage of hierarchical text classification in combination with these loss functions can help improve the average F_1-score, and the models classify small keywords well, thus providing more valuable insights into search behavior.

Contents

Abstract **1**

1 Introduction **4**

2 Background **7**

2.1 Search Engine Marketing 7

2.1.1 Search Engine Advertising 7

2.1.2 Search Engine Optimization 9

2.1.3 Google Ads 10

2.1.4 Keyword Research 13

2.2 Recent Developments 14

3 Data **16**

3.1 Example 16

3.2 Datasets 18

3.2.1 Data Preparation 18

3.2.2 Training, Validation, and Test set 19

3.3 Data Statistics 20

3.4 Keywords and Segments 21

3.4.1 Tokenization 22

3.4.2 Vectorization 23

4 Empirical Approach **24**

4.1 Hierarchical Classification 24

4.1.1 Classification Types 25

4.1.2 Types Hierarchical Classification 26

4.2 Neural Network 29

4.2.1 Layers 30

4.2.2 Multi-Output Neural Network 31

4.2.3 Loss Functions . . . 32
4.2.4 Transformers . . . 37
4.2.5 BERT . . . 38
4.2.6 BERT-like Models . . . 42
4.3 Evaluation Metrics . . . 43
4.3.1 Precision, Recall and F_1-score . . . 43
4.3.2 Hierarchical Evaluation Metrics . . . 44
4.4 Model Evaluation . . . 45
4.4.1 Baseline and Benchmark models . . . 45
4.4.2 Evaluation Metrics . . . 46

5 Results **48**
5.1 Results per Model . . . 48
5.2 Costs and Benefits . . . 52
5.3 Misclassifications . . . 53

6 Discussion **55**

7 Conclusion **57**
7.1 Summary and Conclusion . . . 57
7.2 Recommendations and Further Research . . . 58

References **61**

A Appendix **66**

1 Introduction

In the last few years, internet users have increased yearly. In 2022, internet users rose to 5.07 billion, which equals 63.5 percent of the world population (Kemp, 2022). These internet users are most of the time not directly searching for information and getting to an internet site instantly. According to Yang et al. (2015), approximately 80 percent of internet users searched for information using a search engine in 2015.

At the moment, Google is by far the most used search engine (Statcounter, 2022). More than 8.5 billion searches are done on Google daily (Stats, 2022). As the usage of Google and e-commerce, in general, is this prominent, online advertising is becoming increasingly important. Hence, a crucial part of online advertising is marketing on search engines, which shows advertisements on search engine result pages (Yang et al., 2015). From the revenue of Google follows this rise in search engine usage continues. The revenue of Google in 2021 was equal to 256.74 billion US dollars, an increase of 41 percent compared to 2020 (Bianchi, 2022b). Eighty-two percent of the total revenue comes from advertising, 43 percent more than the previous year (Bianchi, 2022a). Hence, the revenue from advertising is increasing together with the total revenue and the number of users.

These high search engine numbers cannot be separated from the fact that e-commerce is still increasing rapidly. The amount of e-commerce has been increasing rapidly and during COVID-19 even more. Moreover, it is predicted that from 2022 onwards, e-commerce sales in Europe will increase by 47 percent in three years (Statista, 2022). In addition, it is predicted that in 2040 about 95 percent of all purchases will be made through e-commerce (Deniz et al., 2022).

The high number of searches on search engines highlights the importance of search engine optimization (SEO) for organizations that want to increase their visibility and audience. By improving the content used by search engines and the organization's website, the ranking of the websites can be improved, which will increase the organization's visibility. Part of

SEO involves keyword research; this includes investigating which keywords and segments are being searched for on search engines like Google. Doing this can help an organization find interesting keywords and segments for their advertising campaigns. Keyword research consists of approximately 70 percent of classifying the keywords into segments, subsegments, and sometimes subsubsegments. These segments help organizations find their smallest and largest segments, which may not have been noted before but are relevant due to a high search volume. Furthermore, the segments help to reduce many keywords to a few relevant categories for the organization. At the moment, a marketeer is performing this classification manually, and afterward, the classification is checked by several other specialists to avoid mistakes. However, as classifications are performed on files with thousands of keywords, it takes much time to accomplish this classification manually. Automating this process will save time resulting in more keyword research in the same amount of time.

The aim of this paper is thus to design a segment classifier that predicts the segments corresponding to the keywords as accurately as possible, such that time is saved. Nevertheless, there can be many segments and subsegments with only a few corresponding keywords, and the model should select those segments and subsegments correctly with a corresponding hierarchy. Furthermore, the amount of data is limited, as a keyword consists of only four words on average. Hence, there is a minimal amount of context to a keyword. In addition, it should be able to work with English and Dutch data.

The innovation of this paper lies in several points. One point is that the models used to perform the classification were never used on this type of data. Usually, longer keywords or whole sentences are present for classification methods considering hierarchy. In contrast, in this paper, short keywords are used with an average length of four and contain little information per keyword and, thus, are less straightforward to classify correctly. In addition, most of the existing models are trained in English or other common languages, but in this paper, the models are trained in Dutch or sometimes English. Furthermore, during this paper, a new loss function considering hierarchy is designed, and so-called BERT-like models are used for hierarchical classification, which has not been done before, as far as aware. Considering

the hierarchy, these models are combined with a newly designed custom loss function. The F_1-score is used to determine if a model works well. Still, a new metric is also designed based on how close the predicted search volumes are to the actual values in combination with a newly designed corresponding loss function to perform better on this new metric.

The previous paragraph explained that two new loss functions are designed in this paper. One is to learn the model hierarchy, and the other is an addition to the first new loss function designed to reduce the difference between the search volume per segment and the predicted search volume per segment. These loss functions are used within $BERT$-like models. The tests in this paper show that the newly designed custom loss functions perform better on F_1-score than the hierarchical text classification model designed by Gao et al. (2020).

The following sections provide more information about the problem. First, more background on the problem is given. Second, the data and tasks that should be performed on the data before using a model are explained. Thirdly, the approach used to find a good performing classification model and more on the models is explained together with how it will be evaluated. Afterward, the results are displayed and further explained. Lastly, the discussion and conclusion are given.

2 Background

In this section, more information on the central question of this paper is provided to enhance the knowledge of the problem. This information will lead to a better understanding of the problem and what influence a classification model could have. The background information can be split into two sections. First, how search engine marketing works and different types of marketing in search engines. Second, the recent developments in text classification and hierarchical text classification.

2.1 Search Engine Marketing

Digital marketing focuses on every kind of marketing on electronic devices and objects connected to the internet. One part of digital marketing is Search Engine Marketing, also known as SEM. SEM is about increasing a business's visibility on a Search Engine Results Page (SERP) using digital marketing strategies (Sen, 2005). The ability to put their business in front of likely interested customers, and filter the non-interested customers out, is an uncommon ability for other advertising options. SEM consists of two methods, SEO and SEA; both will be discussed in the following sections.

Moreover, the worldwide search engine market share has been dominated by Google during the last year, as displayed in figure 1 (Statcounter, 2022). Hence, the main focus in descriptions during this paper will be on Google, but the idea is the same for the other search engines as Bing and YAHOO!.

2.1.1 Search Engine Advertising

When searching on Google, most of the time, at the top or bottom of the SERP, "Sponsored links" are displayed. The top SERP of "smartphone" is shown in figure 2, where the blue boxes represent the "Sponsored links". These advertisements are based on the keywords entered in the search bar, also known as the search query, which in this case was "smartphone". These "Sponsored links" are used as advertising options in Search Engine Advertising (SEA).

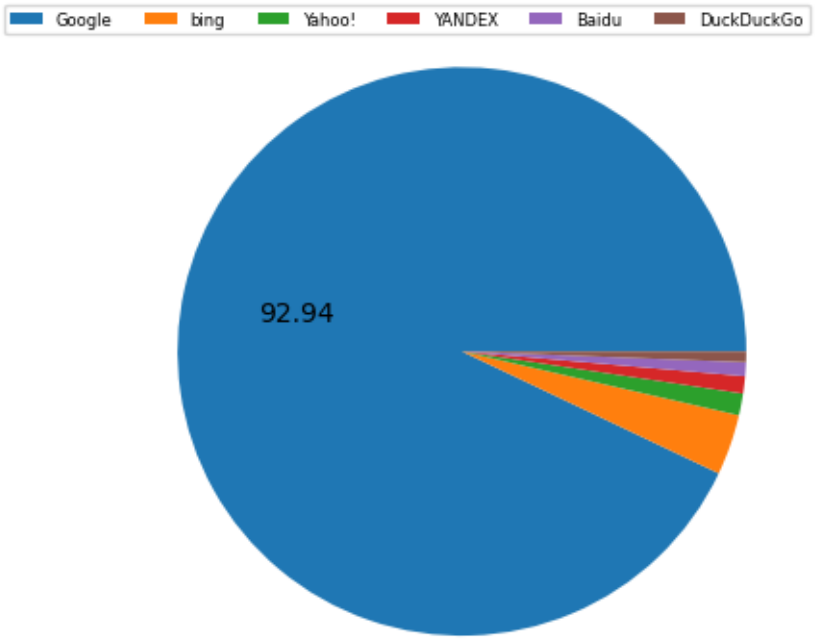

Figure 1: Market share search engines in percentages (Statcounter, 2022)

Some advertisements can be for free, but in Google, the advertisements on SERP are paid, and the prices for the advertisements are dependent on bidding and thus not stable. Additional information on the bidding part and the allocation of advertisements is discussed in the Google Ads section 2.1.3 (Decarolis et al., 2020). The price depends on three main factors (Van Looy, 2022).

First, the location of the advertisement on the SERP. Most of the time, advertisements are on the top and bottom of a SERP. Since most searchers look with more attention at the top of the page, these locations have a higher price than the advertising spaces on the bottom of the pages. Second, whether the demand for advertisement space is high. When more businesses like to show an advertisement at the same time and place, the price will increase due to high demand. Third, the keyword where the advertisement should be shown on. The price will increase if the keyword has high competition, meaning more organizations are willing to advertise on that specific keyword.

2.1.2 Search Engine Optimization

When searching with a search engine, the first links on the SERP are the ones looked at most, and the lower on the SERP, the lower the chance of a customer clicking on the link. Hence, using search engines, one of the main goals is to ensure the page is ranked high on the SERP such that the probability of visiting the advertisers' page increases. This process is done using Search Engine Optimization (SEO). This includes optimizing the landing page, which is the page of arrival when clicking on a link from a search engine. The goal is to make sure the landing page is as relevant as possible to the search query of the user, such that the probability of the search engine algorithm ranking the page higher is increased. The organic results are displayed below the paid ads at the top most of the time, as in figure 2. The result within the green box is the organic result that SEO can improve. SEO was defined as "the art and the science of getting a website to appear prominently in organic search engine results when a search submits a query relevant to that website" (Lieb, 2009).

As the most significant number of clicks is not on the sponsored links but on the organic links, it is recommended to focus on the organic ranking of a website and not only on the "Sponsored links" using SEA (Van Looy, 2022). Through proper application of SEO, it can be made sure that the landing page is ranked high on specific keywords, as a result of which budget can be saved on these particular keywords with SEA.

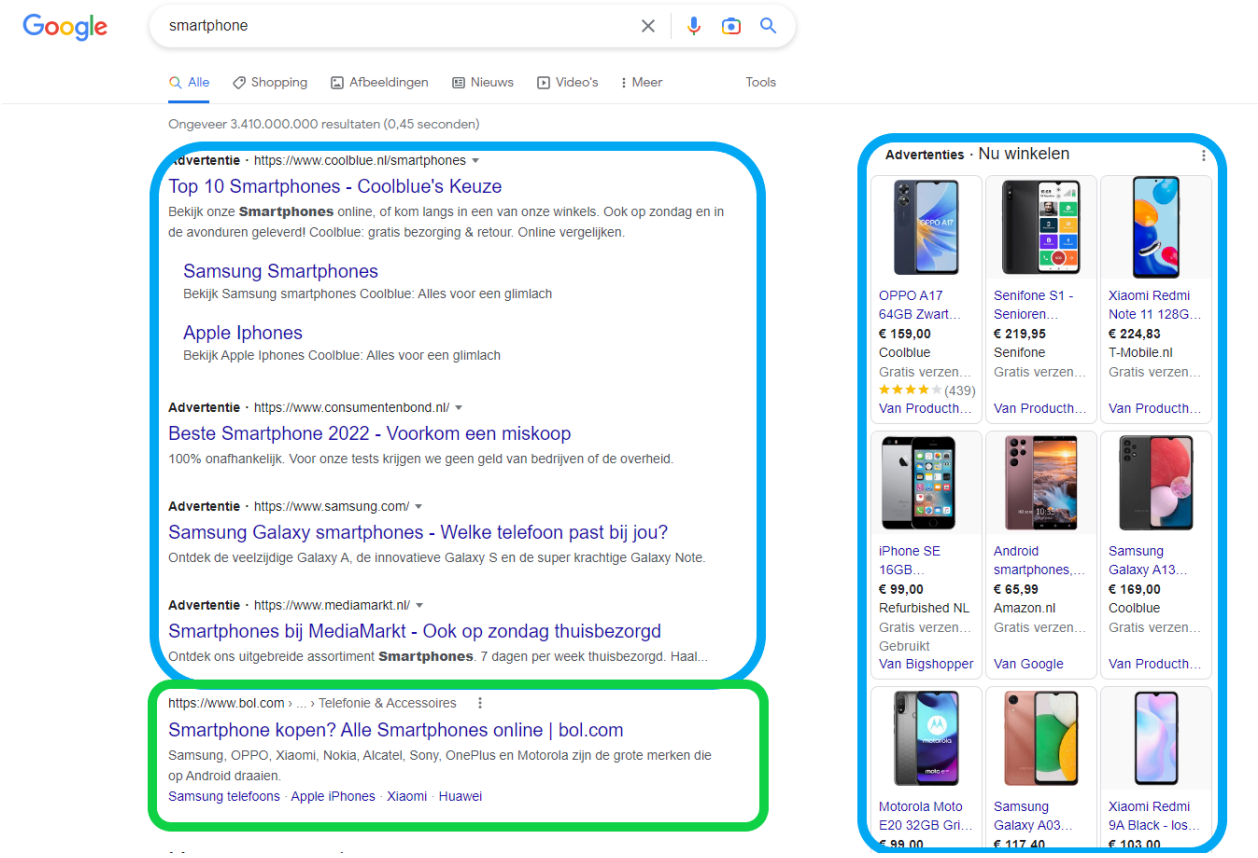

Figure 2: Google SERP "smartphone". Blue: paid results SEA. Green: organic results SEO.

2.1.3 Google Ads

Google has an advertising program based on keywords names, Google Ads (before Google AdWords). Inside Google Ads, most of the time, a budget is set such that no more is spent on bidding each time window. In addition, a bidding system is used to show the advertisement on one of the spots on the SERP (Tricahyadinata and Za, 2017). With the bidding system, there is no payment until a customer clicks on the advertisement, also known as pay-per-click (PPC). A bid, in this case, is thus equal to the amount the organization is willing to spend on one click of a customer searching for a particular keyword. Hence for a specific advertisement, the organization has to determine the keywords it wants to advertise on and the corresponding height of the bid on them. The height of the bid is generally determined by the expected ROI (Return On Investment) or ROAS (Return On Advertising Spend), where ROI focuses on whether the campaign is profitable, and ROAS looks at the effectiveness of

the campaign by investigating if it generates clicks (Gilfoil et al., 2015). The equations of ROI and ROAS are stated below (Gilfoil et al., 2015):

$$\text{ROI} = \frac{\text{Net Profit}}{\text{Net Spend}} * 100$$

$$\text{ROAS} = \frac{\text{Total Campaign Revenue}}{\text{Total Campaign Cost}} \tag{1}$$

On the SERP, it is not the case that the displayed advertisement is the organization that placed the highest bid; it depends on more factors. Otherwise, it might be the case that a high bid is placed, but the site is not relevant to the searcher. Hence, a ranking is made which considers the relevance of the advertisement, the user experience, and the expected click-through rate. Moreover, together with the bid, this decides which ads are shown on the SERP (Abou Nabout and Skiera, 2012).

Which keywords or segments to focus the campaign on has yet to be clarified based on the information above. This decision could be made based on values related to specific keywords. The keyword-related values are the search volume, CPC (Cost Per Click), and Mediavalue.

Search Volume

Search volume is the total number of searches on a keyword over a specific time; this paper uses a year as the period. It indicates if a keyword is well visited; thus, the customer's interest is reflected in search volume. From the search volume, a few things can be concluded. First, if the search volume is high, the keyword is used much by search engine users. These high search volumes can result in two different conclusions. On one side, the given keyword is interesting to focus on since many users have that keyword in their search query. However, on the other side, the high search volume means the competition on ranking for that given keyword is probably high, and thus more effort would be needed to be ranked in the highest places. Second, if the search volume is small, the search engine user is less interested in the keyword, meaning the price and competition are probably low.

Most of the time, the keywords with a small search volume are long. Because these queries are more specific, it is less likely that a user will use the exact query and thus has less

competition. With a short-tail query, the chance of being used is more significant as these queries are usually more general and thus have more competition.

Cost Per Click

Another indication is the cost per click (CPC). The CPC equals the approximate amount needed to pay for a click on the advertisement. For example, if the amount spent on an advertisement is €150 and 164 people click on it, the CPC would be $\frac{150}{164}$, so approximately €0.91. CPC indicates the price corresponding to specific keywords. Focusing on this keyword could be a good idea if the CPC is low. However, it could be the case that there is a minimal amount of clicks, so focusing on this keyword could bring a low additional value as the total amount of clicks on the advertisement could increase by only a few. The CPC is always an estimation beforehand, as the number of clicks on an advertisement is unknown.

Mediavalue

The mediavalue is an estimated cost of buying organic traffic through Google Ads. Often the value is close to the search volume multiplied by the CPC. It indicates the required marketing budget when advertising on a particular keyword. For example, in figure 3, when focussing on the segments ”laundry” and ”waste bins”, it can be seen that the search volume of ”waste bins” is lower than the search volume of ”laundry”. Moreover, the mediavalue of ”waste bins” is larger than the mediavalue of ”laundry”. Thus there is more competition in the ”waste bins” segment as the ”laundry” search volume is larger. From this, it can be concluded that it would be better to focus on the ”laundry” segment because the price would be lower, and the interest from the search engine user is higher in the ”laundry” segment. This choice is made when a choice should be made between these two segments. It should be taken into account that if all segments are compared, it could be that both segments are not the best choice to focus on. Moreover, the choices about which segments to focus on depends on the advertising budget and the expected ROAS, equation 1. If ROAS is high on a more expensive segment, it can still be of interest to focus on this segment instead of the cheaper segments, but for the remaining part of the paper, these options are not considered.

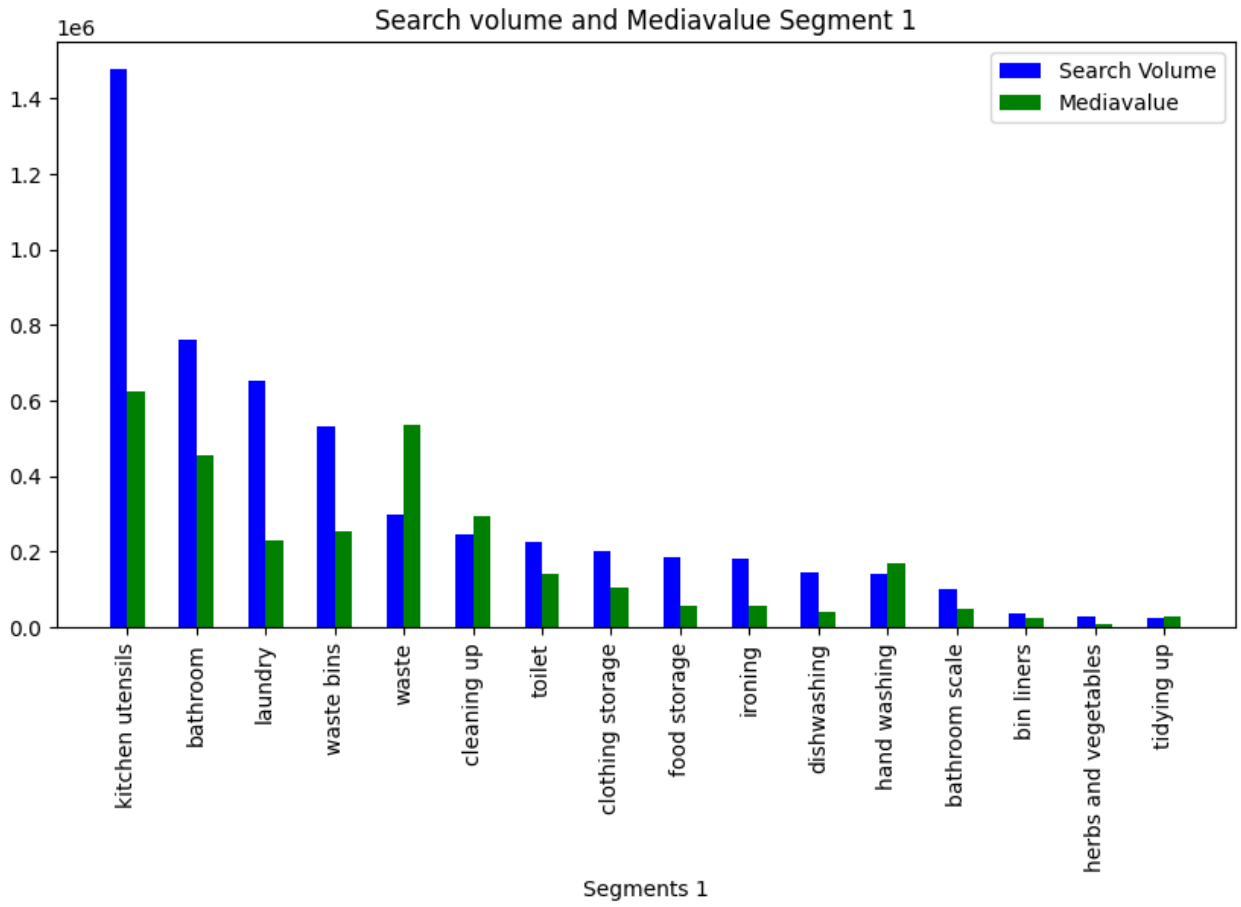

Figure 3: Summed search volume and mediavalue for all main segments (dataset 1)

2.1.4 Keyword Research

Keyword research is essential to choose which keywords improve visibility and find customers interested in the organization's advertisements. An organization can have various reasons to select a particular keyword, which could be related to a product they sell or, for example, to the organization's vision. The goal of keyword research is to map the organization's market and give insights into which areas search engine users are searching and where the possible potential is present but where the organization has yet to focus.

Using keyword research, a file is constructed consisting of a list of useful search queries. An online (paid) tool is often used where one or more keywords are inserted, and the output is a list of many related search queries. Afterward, data corresponding to the keywords in the list are added, for example, CPC, search volume, and mediavalue. Furthermore, it is preferred to

have segments added to each keyword because the segments of the keywords in the file would be used to indicate which segments of an organization are the largest and smallest based on the sum of search volumes per segment. They can lead to new exciting segments that were initially not noted but are relevant due to the large search volume or low mediavalue. Segments also help to reduce a large number of keywords to several relevant categories for the advertiser.

As the created list of keywords may consist of keywords the organization does not want to be related to, for example, war, these keywords are removed or added to the list of negative keywords, which are not added to the list in the first place. The removal is done before segmenting the keywords.

2.2 Recent Developments

The amount of publications about text classification is increasing fast. In 2000, there were only 31 publications; in 2019, there were 292, which increased to 463 publications in a year (Zhu and Lei, 2022). On the other hand, the number of text classification publications about hierarchical classification has decreased in frequency. In the last two decades, more attention has been drawn to topic models, which are models to find the themes and topics of documents, word embedding that find the meaning and relation between words using vector representation, and convolutional neural networks, which are deep neural networks using filters to get features from the input (Zhu and Lei, 2022). Less attention was given to k-nearest neighbors, an algorithm classifying data based on similarity to the other data points, and naive Bayes (Zhu and Lei, 2022). Naive Bayes uses the probability of labels based on the input features to predict labels.

In 2013, Google started by designing series trained on a large amount of data to use in Natural Language Processing (NLP) tasks. The first popular word embedding tool is word2vec. This led to other parties developing better embedding models, such as the Long Term Short Term Model (LSTM), which was trained on much more data. In 2018, embedding models were developed by OpenAI using transformers (Vaswani et al., 2017), a new neural network

structure developed by Google. Due to these developments, specific new models are designed, such as GPT and $BERT$, which again use more training data than the previous models. The trend of using more training data for improvements started in 2018 and is continuing. These models were just the beginning of the developments on this topic. Many variations were made to these models, which should lead to even better performances on NLP tasks. These pre-trained models addressed new state-of-the-art methods in many NLP problems, including text classification.

As the models mentioned before still need some data to fine-tune the model, more recent research has been done on zero and few-shot models. However, these models are made for less data, so they are bigger. To indicate, $BERT$ has $340M$ parameters, $GPT-3$ (Brown et al., 2020), which is a few-shot model, has $175B$ parameters, and $PaLM$ (Chowdhery et al., 2022) has even more parameters, $540B$. $GPT-3$ showed such a large model could be effective for a few-shot model. Hence, other models continued this, as $PaLM$, which uses pathways (Barham et al., 2022). However, using few-shot models results in an accuracy of approximately 60 percent, but fine-tuning these models gives more than 10 percent extra accuracy (Chowdhery et al., 2022).

Concluding, the few-shot models are preferred as they need less data, but few-shot models are not performing better than fine-tuned $BERT$ models. In addition, fine-tuning the $BERT$ models is lightweight. Hence, $BERT$-like models will achieve better results easier. As in this paper, the performance is most important; the other models are not considered. This paper uses hierarchical text classification instead of regular text classification. However, since text classification is in many ways identical, the developments in text classification can also be an option for hierarchical text classification when specific changes are made. From the recent developments stated before, the transformer-based $BERT$-like models are used during this paper; more on these methods is explained in section 4.2.

3 Data

The data used in this paper is collected from the performed keyword research, as explained in section 2.1.4. In this section, the data format will be shown with some insights and changes that should be performed before the models explained in the remaining part of this paper can be used.

3.1 Example

This section explains an example used to increase the understanding and application of the problem. A small amount of keywords is selected from the dataset. Hence the conclusions drawn are only based on the reduced dataset and have no real meaning. The reduced dataset used during this example is displayed in table 1.

Keyword	Search Volume	Mediavalue	Segment 1	Segment 2
Bathroom mirror with shelf	12,100	€ 6,776.00	Bathroom	Mirror
Vanity mirror	22,200	€ 9,768.00	Bathroom	Mirror
Plastic shower caddy	2,900	€ 754.00	Bathroom	Bathroom accessories
Bathroom sets	9,900	€ 7623.00	Bathroom	Bathroom accessories
Basket for toiletries	480	€ 146.40	Bathroom	Storage container
Toothbrush holder	18,100	€ 7,421.00	Bathroom	Toothbrush holder
Wooden spoon for baking	50	€ 27.50	Kitchen utensils	Spatulas
Spatula	27,100	€ 11,111,00	Kitchen utensils	Spatulas
Chopping board	33,100	€ 13,240.00	Kitchen utensils	Cutting board
Ladle	12,100	€ 3,025.00	Kitchen utensils	Spoons

Table 1: Example dataset

The paper aims to design a model that predicts the segments corresponding to the keywords. In this example, using the data displayed in table 1, the first three columns, keyword, search volume, and mediavalue, would be given. The last two columns, segment 1 and segment 2, are the columns to predict using one of the models explained in this paper. The hierarchy

needed for the model is extracted from the data; thus, all possible combinations of segments are assumed to be in the data. Figure 4 shows the hierarchy extracted from the data in table 1. As this paper focuses on supervised learning, some labels should be known to train the model properly.

From the data follows that this example consists of two levels: the first layer consists of two different segments, and the second layer has seven different segments. The hierarchy is extracted from the data and is displayed as a hierarchical tree in figure 4. This hierarchy is assumed to be given before knowing the segments corresponding to the keywords. Knowing this hierarchy and the keywords, a model can be used to predict the segments. More is explained on the possible models in section 4. The results are displayed in figure 13 of the appendix. The only incorrectly predicted keyword is "Wooden spoon for baking"; the actual segment is "Spatulas", but it was predicted as "Spoons". As spoon is present in the keyword, this is a logical prediction of the model. However, spoons are meant to eat; in this keyword, it is a baking spoon, and thus spatulas is the correct subsegment. In addition, in some cases, it can be the case that more than one segment could be a good match, but in the model, only one segment can be predicted, and the others would thus be seen as incorrect; real examples of this problem are discussed in section 5 with the results. The exact predicted and actual values per segment are displayed in tables 9 and 10 of the appendix. Since the mediavalue and search volume of the incorrectly predicted keyword is small, the change in total mediavalue and search volume is also small compared to total values per segment and would thus not significantly affect the conclusion. In addition, from the results can be seen that in this case, the main topics are almost equal, and from the sub-topics, the mirror segment is the largest for this organization. However, no conclusion can be drawn from this since only a small part of the data is used. Furthermore, to evaluate these results, the evaluation metrics discussed in section 4.3 are typically used for model performance and comparison.

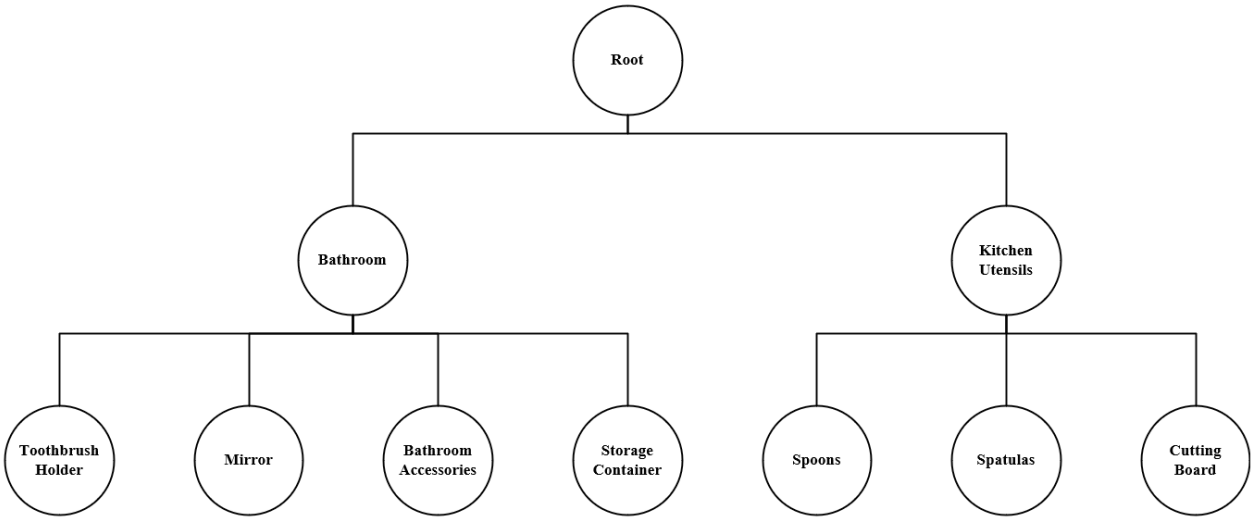

Figure 4: Hierarchical tree of example data table 1

3.2 Datasets

This paper uses several datasets to test the model's performance on datasets with different segments. All the datasets are in various fields and lengths, as displayed in section 3.3. Table 1 in the previous section shows part of dataset 1. The columns with the actual segments are labeled based on a classification structure delivered by the organization. A specialist labels the keywords, and several other specialists verify them such that the segments are not dependent on personal opinion. An assumption based on the labeling is that the labeling done by the specialists is true. Hence from this point onward, the manually classified model will be referred to as the "True model".

Before the dataset can be input into the classification model, a few steps should be executed on the whole dataset and some only on the items in the keyword and segment columns. These steps will be described in the following sections.

3.2.1 Data Preparation

First, all of the unnecessary columns are removed from the dataset. Only the columns about the keyword, segments, search volume, and mediavalue stay in the dataset. Some of the

datasets have hierarchical layers with a large number of unknowns. If this amount is larger than ten percent of the total length, the layer is not included in the classification, and else the blank values are regarded as a different segment. Second, the segments in the segment columns are converted to lowercase since they would be recognized as the same segments. Lastly, as seen in the data statistics, some layers have segments with only one keyword allocated to that segment. When splitting into different sets, these observations would lead to no keyword of the segment in the training, test, or validation set. Thus the choice was made to remove the segments with the corresponding observations, which have less than six observations per segment.

After these preparations, a hierarchy dictionary is created based on the combinations of the different segments. Every node, not an end node, would have the possible children followed in the dictionary. This establishes the hierarchy needed for the model explained in the next section. During the creation of the hierarchy dictionary, it is assumed that every possible combination of segments is present in the dataset. Furthermore, suppose a segment is present beneath several different parent nodes. In that case, the child node is included beneath all the parent nodes separately to ensure the structure is a tree instead of a DAG. This assumption is made because a DAG would lead to models that cannot be used anymore or are more challenging, for example, a classifier per parent node, as explained in section 4.1.2. The data used in this paper only has a few child nodes that have multiple parent nodes. For the models different from the classifier per parent node, it does not have an effect changing from a DAG to a tree.

3.2.2 Training, Validation, and Test set

The Neural Network and the other used models need training, testing, and validation data. The methods used are supervised, where the training, testing, and validation observations cannot overlap. Hence, the dataset is split into three sets, the training, validation, and test set. Let the training set consist of eighty percent of the observations and the validation and test set of ten percent. The training set is used to train the model, the test set tests the model, and the validation set is used to check if the trained model performs better than

before on data it was not trained on. Furthermore, as follows from figure 14 of the appendix, the dataset is unbalanced, which is the case for all datasets used in this paper. Hence, to reduce this problem during the splitting process, stratification is used on the last layer of used segments.

3.3 Data Statistics

This paper will test the models on three different datasets, as they could perform differently on datasets from another field. The datasets used, and some statistics from the datasets are displayed in table 2. The number of unique segments per layer is stated within the "Hierarchical layers" brackets. Dataset 1 initially had three hierarchical layers. However, figure 15 of the appendix shows a large percentage of unknowns in the third segment of dataset 1. As explained in section 3.2.1, this segment is removed because the percentage of unknowns is too high.

Dataset	Field	Language	Hierarchical Layers	Number of Observations	Training Set	Test/Validation Set
1	Domestic products	English	2 (16/108)	11,932	8,352	1,790
2	Funerals and Insurances	Dutch	2 (33/129)	8,791	6,153	1,319
3	Perfume, Cosmetics and Skin care	Dutch	2 (6/31)	5,860	4,102	879

Table 2: Details on used datasets

Furthermore, figure 5 displays how the lengths of the keywords are distributed for each dataset, and the average is displayed with a red dotted line. The keywords' lengths are similar in the different datasets from these plots.

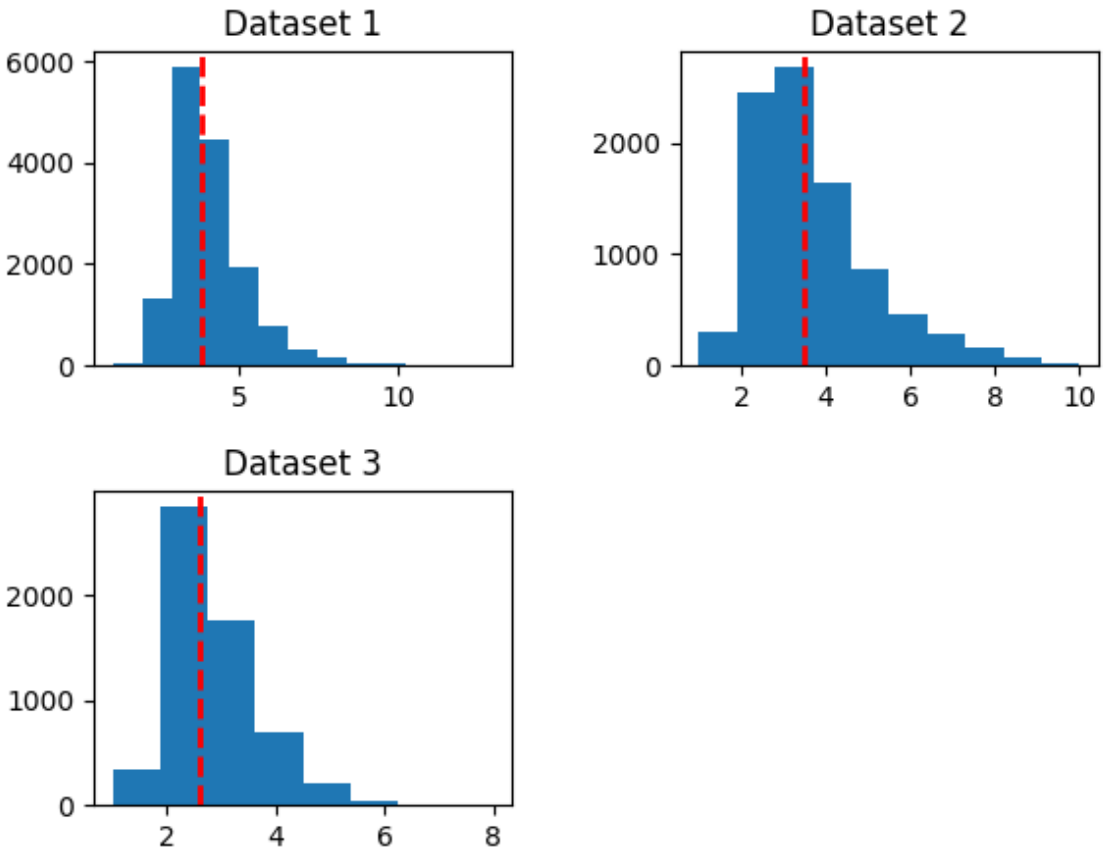

Figure 5: Distributions k dataset

In addition, it is essential to know how many keywords are in each segment. This is only displayed for dataset 1, the only English dataset, but it is the same idea for the other datasets. The result of the first segment of dataset 1 is displayed in figure 14 of the appendix. As discussed in section 3.2.1, the segments with less than six keywords are already removed from this dataset. From this figure, the dataset is unbalanced because the number of keywords in the segments is far from the red-dotted average line for specific segments.

3.4 Keywords and Segments

Given the prepared data, the words should be converted to numbers, so the model understands the terms and, later, the sentence's meaning.

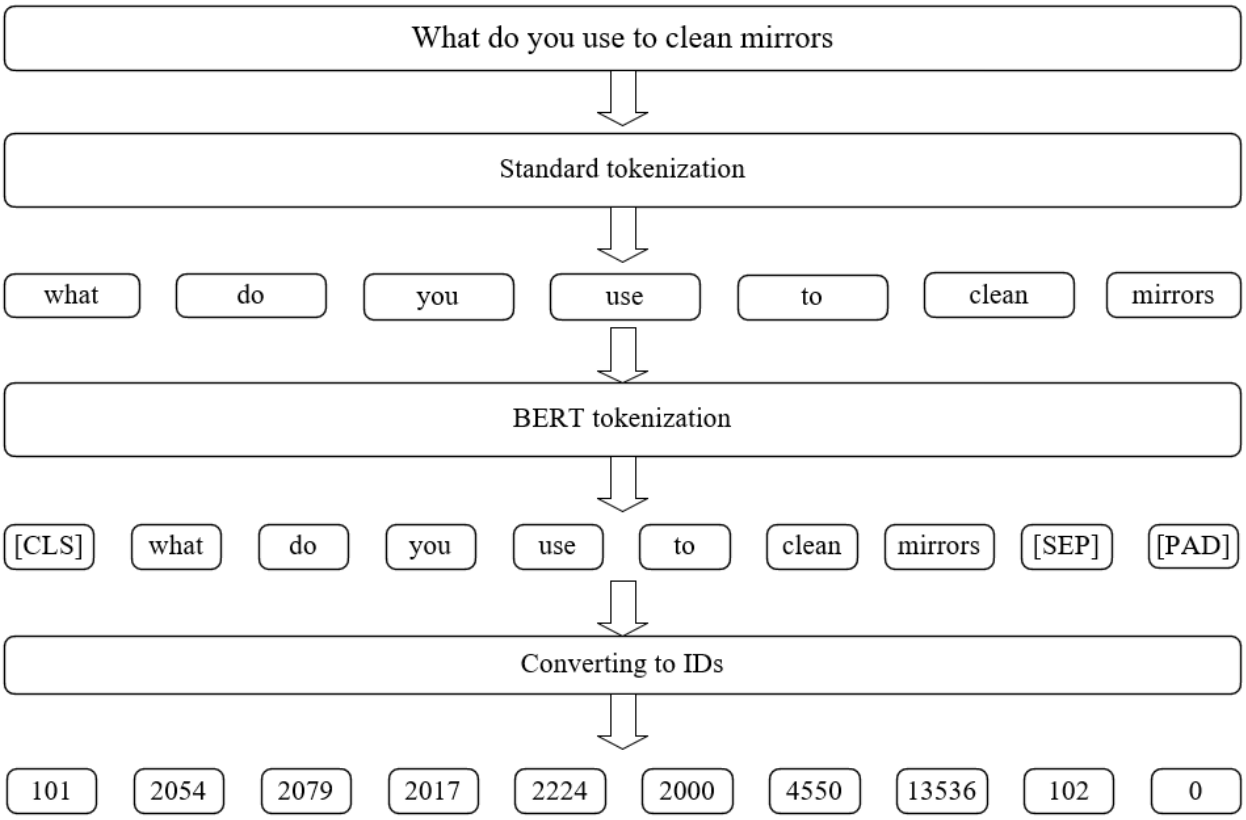

Figure 6: Tokenization and vectorization process

3.4.1 Tokenization

Every sentence in the keywords column should be split into words and punctuations. Tokenization is having all words and punctuation separate in a list. An example of a tokenized sentence is displayed in figure 6. For the multi-output neural network, the standard tokenization is enough. However, $BERT$ or one of the $BERT$-like models requires a different tokenization. The first step of $BERT$ tokenization is the same as the standard tokenization. The upcoming steps are different. At the beginning of every sentence, the $[CLS]$ token should be added, and at the end, the $[SEP]$ token should be added (Devlin et al., 2018). Furthermore, all tokenizations of $BERT$ should have an equal length. Hence, if the size of the current tokenization is shorter than the maximum size, the current tokenization should be supplemented with $[PAD]$ tokens (Devlin et al., 2018). The steps specific to a $BERT$-like model are included in figure 6 in the "$BERT$ tokenization" step.

3.4.2 Vectorization

The process of converting the tokenized sequence to numbers to be fed into a machine-learning model is called vectorization. There are multiple methods to vectorize the data. In this paper, the segments for the multi-output model are vectorized using one-hot encoding; since the segments are categorical variables, every possible segment gets a different number assigned. The assigned number shows which place in the vector a one is placed instead of a zero. The keywords need another method because every keyword is unique. For this, the LabelEncoder of the sklearn python package is used, which assigns a number to every unique token in the training data and sets the number at the place of the token in the vector. Suppose the vector is smaller than the maximum length encoding. In that case, the vector is complemented with zeros till all the encoded vectors have an equal length, just as padding in this $BERT$ tokenization. LabelEncoder is simple and has good results, but it has the disadvantage that the integer values do not have any intrinsic meaning. The $BERT$ model uses a different vectorization method; it uses WordPiece embedding with a vocabulary of $32,000$ tokens designed by Wu et al. (2016). This vectorization is included inside the pre-trained "BertTokenizer". The other $BERT$-like models use a similar method.

4 Empirical Approach

This section explains the approach used to find the most suitable model to hierarchically classify the keywords based on the metrics F_1-score and the predicted search volume and mediavalue per segment. The empirical approach can be split into two categories. First, the model used to obtain a hierarchical classification of keywords is explained. Second, the metrics used to evaluate and compare the model's performances. These topics are further discussed in the following subsections.

4.1 Hierarchical Classification

A standard text classification attaches the text to one of the provided labels. It is called hierarchical classification when the given text can have multiple labels, and these labels should follow a specific order. The structure of the labels can be described as a tree. Hence, hierarchical classification can be seen as an organized classification. Figure 7 shows a hierarchical tree with two levels. The positions in a tree are defined with specific terms. The top node is always called the root node, and the nodes at the end of a path are called the leaf nodes. Furthermore, the tree has some family relations. The parent nodes are defined as the nodes which have nodes beneath. In figure 7 the *Root*, 1 and 2 are parent nodes (Gopal et al., 2012). The nodes beneath a parent node are defined as child nodes. In figure 7, 1 and 2 are child nodes of *Root*, and 1.1 and 1.2 of 1, and lastly, 2.1 and 2.2 are child nodes of 2 (Gopal et al., 2012). In the following sections, the existing classification types and the types of hierarchical classification are described.

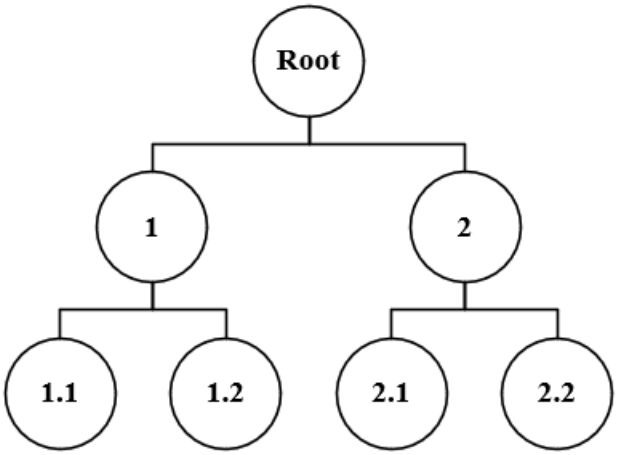

Figure 7: Basic hierarchical tree

4.1.1 Classification Types

As Borges et al.(2013) explains, classification has three different main characteristics, namely: Tree (T) or Directed Acyclic Graph (DAG), Single Path of Labels (SPL) or Multiple Path of Labels (MPL) and Full Depth (FD) or Partial Depth (PD).

Tree or Direct Acyclic Graph

In both cases, the root node has no parent node. The difference between labels of a hierarchical classification problem following a DAG and a Tree is that with a Tree, every node has only one parent node. Hence, with labels following a DAG, a node can have more than one parent node. (Borges et al., 2013)

Single Path of Labels or Multi Path of Labels

Single Path of Labels is used when the result for every text should be only one path. Otherwise, Multi Path of Labels is used when more than one path could result from a text. (Borges et al., 2013)

Full Depth or Partial Depth

If the classification is Full Depth, every path should lead toward a leaf node. Otherwise, the length of the classification output is not equal to the number of layers in the hierarchy. In this case, the classification is Partial Depth. (Borges et al., 2013)

Furthermore, three different kinds of problems could be present. First, a multi-class problem. In a multi-class problem, the label of a text classification model is chosen out of more than two different classes. Second, multi-label classification matches one or more labels to every input. This differs from the multi-class problem because it can have multiple labels, whereas a multi-class can only have one. Lastly, multi-output differs from multi-class and multi-label since it can occur simultaneously with one of the two cases mentioned before. Multi-output occurs when the model has multiple branches and, thus, multiple outputs per observation.

During this paper, it is assumed that nodes can have only one parent node (tree), the result should be a single path (Single Path of Labels), and the path should end with a leaf node (Full Depth). In addition, every dataset has more than two different labels per layer, multi-class, and every layer in the hierarchy has a different output. Hence, this paper combines a multi-output model with a multi-class model. In addition, hierarchy is added to this combination. Which types of hierarchical classification can be used are explained in the next section.

4.1.2 Types Hierarchical Classification

Hierarchical classification is known to have different approaches to solving the problem, but all have some pros and cons. The three main types of approaches are explained in the following paragraphs: flat, global, and local.

Flat Hierarchical Classification

Until the mid-1990s, the hierarchy was ignored, and hierarchical classifications were performed similarly to standard text classifications (Kiritchenko et al., 2005). All layers in the hierarchical tree, except the layer including the leaf nodes, are removed. In figure 7, only the

leaf nodes 1.1, 1.2, 2.1, 2.2, and the *Root* node are left in the classification problem. This approach is known as flat classification. The flat classification does not consider meaningful information about the hierarchy.

Local Hierarchical Classification

A local hierarchical classification method builds several different classifiers. This method is also known as top-down level-based because it takes the most relevant categories first, which are displayed in the first layers (Borges et al., 2013). Hence, the model works from top to bottom. This local approach can be split into three different methods based on the choice of classifiers.

First, the local classifier per node. In this case, a binary classifier is trained for every node in the hierarchical network (Pereira et al., 2021). For example, in figure 7, this would result in six classifiers, two for the first level and four for the second level. However, a disadvantage of the local classifier per node is that if the hierarchy gets larger, the number of classifiers needed to train will increase rapidly. Furthermore, it is still being determined if the hierarchy is respected using this approach. The local classification method per node is the most used local classification method. It was, for example, used by Xue et al.(2008), Valentini (2009), Barutcuoglu et al. (2006), and Guan et al. (2008). This method is preferred to the other local methods when the sub-categories of the child nodes of a parent node are very different.

Second, the local classifier per parent node. As the name indicates, one multi-class classifier is trained per parent node (Pereira et al., 2021). One disadvantage of this method is that it cannot be used on problems with a DAG structure (Borges et al., 2013). Referring to figure 7, this would mean two classifiers are trained, one for node 1 and one for node 2. Hence, the amount of classifiers to train is reduced compared to the local classifier per node method. This approach was used by Silla and Freitas (2011). This method is preferred to the other local methods when the child nodes of a parent are more similar.

Lastly, the local classifier per level approach. This approach trains one multi-class classifier

per level in the hierarchy (Freitas and Carvalho, 2007). This method can be used for trees and DAGs', but applying it to a DAG may be more difficult as there can be more than one possible path (Borges et al., 2013). Applying the local classifier per level approach to figure 7 would lead to training two multi-class classifiers, as these two layers are present in this example. This method is preferred to the other local methods when the different categories at every level in the hierarchy are similar. Still, the categories at a different level can be very different from another level.

Global Hierarchical Classification

The local classifiers mentioned in the previous paragraph need more than one trained classifier. Hence, if the number of levels or nodes in the hierarchical structure increases, the number of trained classifiers also increases. Hence, considering the hierarchy, having one classifier for the whole model would be preferred. This is defined as a global hierarchical classification. There has been limited research on the global methods compared to the local methods. However, in the last few years, research on global methods increased.

Global hierarchical text classification methods organize text data into a hierarchy of categories or classes. Here are some papers that discuss existing global hierarchical text classification methods: Zhou et al. (2020), Deng et al. (2021), Meng et al. (2019), Wang et al. (2021). Zhou et al. (2020) used a "novel end-to-end hierarchy-aware global model" where the hierarchy is a directed graph, and the model is implemented as a hierarchy encoder. However, this model has two limitations which are resolved in HTCInfoMax (Deng et al., 2021) by using information maximization, including the following two modules: "text-label mutual information maximization" (Deng et al., 2021) and "label prior matching" (Deng et al., 2021). Furthermore, WeSHClass uses a weakly-supervised neural method for hierarchical text classification with a hierarchical neural structure (Meng et al., 2019). Lastly, Wang et al. (2021) imitated the cognitive structure learning for the hierarchical multi-label text classification; the model is called Hierarchical Cognitive Structure Learning Model (HCSM). HCSM exists of an Attentional Ordered Recurrent Neural Network (AORNN) part and a Hierarchical Bi-Directional Capsule (HBiCaps) part (Wang et al., 2021).

This paper uses a flat classification method, a local classification with a classifier per parent node, and global methods. The flat classifier is selected to have a baseline, and the local classifier is selected to compare to the global hierarchical classifiers. Specifically, the local classifier per parent node is selected, instead of the other local classifiers, because the child nodes are similar. However, this is an interpretation that may differ per person. Furthermore, the global methods are selected because they are preferred as it is trained ones. However, the models mentioned in the previous paragraph are used on long keywords in English. This paper uses a hierarchical classification method for shorter keywords and also in Dutch. Hence, the models mentioned before are not used in this paper. Global methods will be combined with neural networks for a new model, as neural networks were shown to be efficient to use with hierarchical classification by Wang et al. (2021) and Meng et al. (2019), which are explained in the next section.

4.2 Neural Network

A neural network (NN) can be compared to the working of a human brain, the biological neuron. Biological neurons have a cell body consisting of a nucleus, synapses, and axons (Livshin, 2019). Synapses get input/impulses and process them towards the cell body. The cell body then processes it towards the axon and the axon to the output synapses (Livshin, 2019). Converting it towards an artificial neuron is done by simplifying and changing it slightly. Each input to a neuron has a specific weight indicating the influence an input has on predicting the output. The neuron itself most often consists of two parts. First, the calculations performed on the input layers are explained in the next section. Second, the activation function is explained below.

There are several different activation functions. The choice of function depends on the range, how fast the function changes, and personal preferences. The most popular activation function is the Softmax (2) which is defined below (Schmidt-Hieber, 2020). The Softmax function rescales the values, so all are in the $[0, 1]$ range. However, there are many more options to use as an activation function.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2}$$

4.2.1 Layers

The layers used during this paper for the structure of a NN are explained in this section.

1. **Embedding Layer**: a layer that converts categorical variables into dense vectors representing complex relationships between the categorical and target variables. It is often used as the first layer of a neural network. It takes a list of integers as input, each integer representing a different vocabulary word, and the output is the embedding. The embedding layer searches for the embedding vector for each integer in a weight matrix and returns a dense vector for each integer.

2. **Linear Layer**: a fully connected or dense layer, is often used in neural network architectures. It consists of a set of neurons, which receive input from the neurons of the previous layer and produce an output passed to the next layer. The output of a neuron is then a linear combination of the inputs, $y = xA^T + b$, weighted by a set of parameters called weights and biases. These weights and biases are learned during the training process of the neural network by using an optimization algorithm that adjusts them to minimize the error between the network's predictions and the actual labels of the training data. Linear layers are used to learn complex nonlinear relationships between the input and output of a neural network, and they are an essential part of many neural networks. The input of the layer is equal to the size of every input sample, and the output is equal to the size of every output sample.

3. **BERT Layer**: as this layer is more complex, it will be explained separately in section 4.2.5.

To find the best-performing model for our hierarchical data. First, a multi-output neural network is tested to determine how a model which does not contain hierarchy performs. Afterward, a custom loss function is added to learn the model hierarchical understanding.

Lastly, *BERT* or a *BERT*-like model is implemented because it has more knowledge of the context of sentences. These possible models are further explained in the upcoming sections.

4.2.2 Multi-Output Neural Network

A multi-output neural network is a model which gives two or more outputs. In a hierarchical classification model, the number of outputs would equal the number of hierarchical levels. Hence, the model's input would be a list of keywords, and each output would correspond to the segments of a different layer in the hierarchy. For the model to classify the keywords correctly in multiple classes, it should know how many unique hierarchical classes are present where the keywords could be classified into for each hierarchical layer. The layout of the model, including layers, would then be as follows:

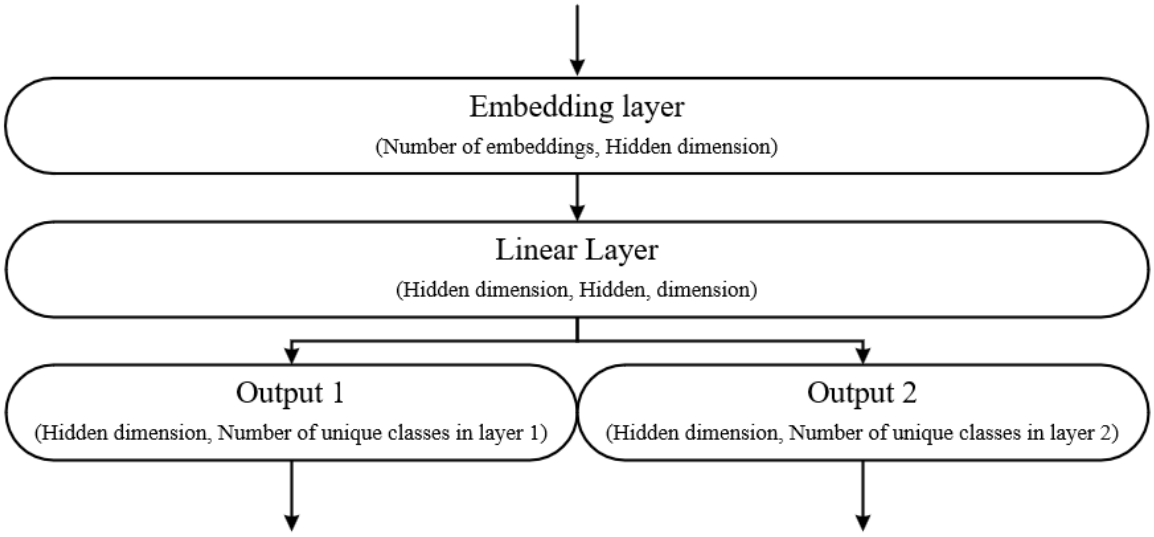

Figure 8: Model architecture of multi-output neural network

Within the brackets of the linear and output layers, the input size is stated, and the output size is displayed. In this figure, two outputs are displayed. When more than two layers are present in the hierarchy, more outputs should be added to outputs. In section 4.2.1, the function of each different layer is explained, including the input and output.

In this model, Categorical Cross-Entropy loss, displayed in equation 3, is used, which is frequently used for multi-class classification problems and explained further in the next sec-

tion. It is applied separately to each output with a Softmax function and summed afterward. Other settings of the model are displayed in table 11 of the appendix.

4.2.3 Loss Functions

A Neural Network is trained using back-propagation by adjusting the network weights to minimize the error between the desired and actual outputs (Goh, 1995). It uses gradient descent to take small steps in the direction that reduces the error and a user-defined learning rate to control the size of these steps. The process involves feeding the input data through the network to get the output, calculating the error, and propagating it back through the network to adjust the weights (Goh, 1995).

One of the most common optimizers used to train using back-propagation is Adam (Huang et al., 2019). Adam uses gradient-based stochastic optimization; it uses a small amount of memory and needs first-order gradients. The algorithm used with Adam was designed by Kingma and Ba (2014). In addition, a new, improved version of Adam is designed by Loshchilov and Hutter (2017) called AdamW. AdamW is Adam with decoupled weight decay, with a better performance on most tasks than the regular Adam. The pseudo-code of AdamW designed by Loshchilov and Hutter (2017) is stated in algorithm 1. AdamW only has one small change compared to the Adam algorithm, and this change is displayed with a green box in the algorithm. In the algorithm, λ is the weight decay value, and η_t is a scaling factor defined using $SetScheduleMultiplier(t)$. To clarify, g_t^2 is an element-wise square and β_1^t and β_2^t are the betas defined in the algorithm to the power t. In addition, all operations applied to vectors are done element-wise.

In algorithm 1, the pseudo-code of the AdamW algorithm is displayed. In this paragraph, the steps are explained more extensively. Starting in lines 4 and 5, $f(\boldsymbol{\theta})$ is a stochastic scalar function with parameter $\boldsymbol{\theta}$ (Kingma and Ba, 2014). In this algorithm, $\nabla f_t(\boldsymbol{\theta}_t)$ represents the gradients with respect to $\boldsymbol{\theta}$ at t because the goal is to minimize the expected value of $f(\boldsymbol{\theta})$. Continuing at line 6 up to and including 9, $\mathbf{m}_t$ and $\mathbf{v}_t$ are updated, which are the exponential moving averages and the squared gradient. The betas in these lines take care

Algorithm 1 Adam and Adam with decoupled weight decay (AdamW) taken from: Loshchilov and Hutter (2017) and Kingma and Ba (2014). The default settings used are $t = 0$, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\lambda \in \mathbb{R}$.

1: **initialize** parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\mathbf{m}_{t=0} \leftarrow \mathbf{0}$. second moment vector $\mathbf{v}_{t=0} \leftarrow \mathbf{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
2: **while** θ_t is not converged **do**
3: $t \leftarrow t + 1$
4: $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow SelectBatch(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the gradient
5: $\mathbf{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ ▷ get gradients with respect to stochastic objective at t
6: $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$ ▷ update biased first moment estimate
7: $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2$ ▷ update biased second raw moment estimate
8: $\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\beta_1^t}$ ▷ compute bias-corrected first moment estimate
9: $\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1-\beta_2^t}$ ▷ compute bias-corrected second raw moment estimate
10: $\eta_t \leftarrow SetScheduleMultiplier(t)$ ▷ can be fixed, decay, or be used for warm restarts
11: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t(\frac{\alpha \hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon} + \lambda \boldsymbol{\theta}_{t-1})$ ▷ update parameters
12: **end while**
13: **return** optimized parameters θ_t

of the exponential decay rate of the moving averages. Since $\mathbf{m}_t$ and $\mathbf{v}_t$ are initialized as zero vectors, they can be biased towards zero in the first steps or if the decay rates are low (Kingma and Ba, 2014). To solve this problem, bias-corrected first and second moments estimates are computed in lines 8 and 9. Afterward, the parameters can be updated, and start the loop again till the stopping criterion is met.

The optimizer is trying to reduce the error of the next round by changing the weights. This error is also known as the loss function. Every problem prefers a different loss function. Most loss functions require a Softmax layer before applying the loss function. The application of Softmax is explained in equation 2. For multi-class classification, the most popular loss

function is cross-entropy loss. Below the cross entropy loss is stated (Teahan, 2000):

$$Lloss_l = -\sum_{j=0}^{n} y_{lj} log(\tilde{y}_{lj}) \tag{3}$$

where n is the number of different classes and l means the l^{th} hierarchical layer. Hence, y_{lj} corresponds to the expected output of layer l and class j. Furthermore, $\tilde{y}_{li}$ corresponds to the probability of the Softmax layer of class i and layer l of the hierarchical layers.

$$y_{lj} = \begin{cases} 1 & \text{if class j of the } l^{th} \text{ layer is expected} \\ 0 & \text{if class j of the } l^{th} \text{ layer is not expected} \end{cases} \tag{4}$$

However, the cross entropy loss does not consider the hierarchy between the layers. To include this, a hierarchical loss is designed by Gao et al. (2020):

$$Hloss_l = -c_l^{\mathbb{I}_l \mathbb{Q}_l} c_{l-1}^{\mathbb{I}_{l-1} \mathbb{Q}_l} \tag{5}$$

where $\mathbb{Q}_l$ and $\mathbb{I}_{l-1}$ are defined below and c can be a constant or an error in prediction.

$$\mathbb{Q}_l = \begin{cases} 1 & \text{if predicted label in the } l^{th} \text{ layer is not a child class of the predicted class} \\ & \text{in the } (l-1)^{th} \text{ layer} \\ 0 & \text{else} \end{cases} \tag{6}$$

$$\mathbb{I}_{l-1} = \begin{cases} 1 & \text{if } \hat{y}_l \neq y_l \\ 0 & \text{else} \end{cases} \tag{7}$$

In equation 7, $\hat{y}_l = \max_l \tilde{y}_{li}$ is the predicted class for layer l. Furthermore, y_l represents the label of the l^{th} layer. The indicator function mentioned in equation 6 is 1 when the hierarchy is wrong; the predicted parent and child nodes do not correspond to the hierarchy given in this case. Moreover, the indicator in equation 7 shows when the prediction is incorrect. Overall, the hierarchical loss function penalizes when the hierarchy of the predictions is incorrect and the prediction of one of the labels in the layer is incorrect.

Combining the loss per layer and the custom hierarchical loss leads to the total loss function:

$$Loss(\theta) = \sum_{i=1}^{L} \alpha_i Lloss_i + \sum_{i=2}^{L} \beta_i Hloss_i \tag{8}$$

where α_i and β_i can be chosen to determine the weights between layer loss and hierarchical loss. In this paper, the choice is made to use $\alpha_i = 0.8 \; \forall \; i = 1, \ldots, L$ and $\beta_i = 1 \; \forall \; i = 2, \ldots, L$. Furthermore, the θ displayed in the loss function is the parameter. According to Gao et al. (2020), this loss function is working effectively on text data.

Custom Loss Function

The loss function of a NN determines where the model should optimize on. When using the categorical cross-entropy loss, mentioned in equation 3, the models' only importance is to have the correct segment to the keyword, and the different hierarchies in the segments are not taken into account. To account for the hierarchy in the classification model, the loss function of Gao et al.(2020), mentioned in the previous paragraph, is implemented. Using this loss function, the model is optimized on the correctness of the hierarchy.

As explained before, the loss function of Gao et al.(2020) consists of a standard, cross-entropy part, and a hierarchical part. Penalization of the hierarchical part in the loss function only occurs if the hierarchy is wrong. Hence, only the standard cross-entropy loss function is present when the hierarchy is correct, but both predictions are wrong. An extra penalty is included in the loss function as this case is worse because the model should not get less loss because of "accidentally" having a correct hierarchy while all predictions are wrong. The following part would be added to the hierarchical loss defined in equation 5:

$$(1 - \mathbb{Q}_l)(c^{\mathbb{I}_l \cdot \mathbb{I}_{l-1}} - 1) \tag{9}$$

where c is a constant and $\mathbb{Q}_l$ and $\mathbb{I}_l$ are defined in equations 6 and 7. The first part of equation 9, $(1 - \mathbb{Q}_l)$, filters out the observation that already has a hierarchical loss, where $\mathbb{Q}_l = 1$. Furthermore, $c^{\mathbb{I}_l \cdot \mathbb{I}_{l-1}}$, is included such that the extra penalty is not added if one of the previous or current observations is predicted correctly. Lastly, the -1 is added to let the penalty start from 0 instead of 1. The total hierarchical loss will then change toward the following equation:

$$Hloss_l = -c_l^{\mathbb{I}_l \mathbb{Q}_l} c_{l-1}^{\mathbb{I}_{l-1} \mathbb{Q}_l} + (1 - \mathbb{Q}_l)(c^{\mathbb{I}_l \cdot \mathbb{I}_{l-1}} - 1) \tag{10}$$

Custom Loss Function based on Search Volume

The previous paragraph showed a custom loss function designed to minimize the amount of incorrectly predicted segments. However, as explained in section 2.1.3, some keywords are searched for often, and some are not used frequently. In the custom loss function mentioned before, these cases are weighted equally. However, if a keyword with a small search volume is incorrectly predicted, this will not have an enormous influence on the sum of search volumes which conclusions are drawn from. Although, if a keyword with an extensive search volume is incorrectly classified, it has a much more considerable influence on the conclusion. Hence, a different loss function can be designed to consider the difference in importance based on the height of the search volume. In addition, the mediavalue could be added, but as the relationship is usually close together, only the search volume is used in this paper.

The custom loss function, considering the importance of a keyword, is added to an existing loss function in this paper on the newly designed loss function mentioned in the previous paragraph. The search volume part is displayed in the equation below. A natural logarithm is used to scale the search volumes, and a plus one is added to exclude zero values. Furthermore, it is multiplied by the indicator function, indicating correct classification.

$$SVloss_l = ln(sv_l + 1) \cdot \mathbb{I}_l \tag{11}$$

Including the function in the new custom loss function mentioned in the previous paragraph leads to the loss function below. Where the search volume loss is an addition to the previous custom loss. This loss function is meant to ensure the model creates correct predictions and reduces the deviation from the sum of search volumes per segment, as explained in section 4.4.2, the $MAPE$ metric.

$$Loss(\theta) = \sum_{i=1}^{L} (\alpha_i \cdot Lloss_i + \gamma_i \cdot SVloss_i) + \sum_{i=2}^{L} \beta_i \cdot Hloss_i \tag{12}$$

4.2.4 Transformers

The previous sections provide the options in a NN when a model is designed. Hence, the focus is on the design of a model from this point. A part of the used model will consist of $BERT$, which was stated in the layers section, but to explain the $BERT$ model better, the transformers should be introduced. Transformers are a network architecture introduced in the paper "Attention is All You Need" (Vaswani et al., 2017) based on self-attention mechanisms, which make the model better at considering the relationship between different parts of the input sentence, and it processes the sentences sequentially. This allows the model to weigh some input parts more than others (Vaswani et al., 2017).

In figure 9, the structure of a transformer model is displayed. In this paper, a transformer takes a sequence of tokens extracted from a sentence and produces an output. To get to the output, the transformer first embeds the input tokens and then processes the embedded input tokens through a series of self-attention layers (Vaswani et al., 2017). In each self-attention layer, the model computes attention weights for each input vector relative to all the other input vectors (Lin et al., 2022). The attention weights represent the importance of each input vector in relation to the others and are used to compute a weighted sum of the values, which is then used as input to the next layer in the model. The self-attention layers are interleaved with feedforward layers, which process the input using standard neural network techniques such as convolution and pooling layers (Lin et al., 2022). The transformer's output is computed by applying a final feedforward layer to the output of the self-attention layers. Lastly, the outputs of the decoders are inserted into a linear layer and afterward in a softmax layer. To conclude, transformers can process the input sequence in parallel instead of sequentially, like many other models (Vaswani et al., 2017). This makes transformers faster to train and evaluate and handle long input sequences more easily.

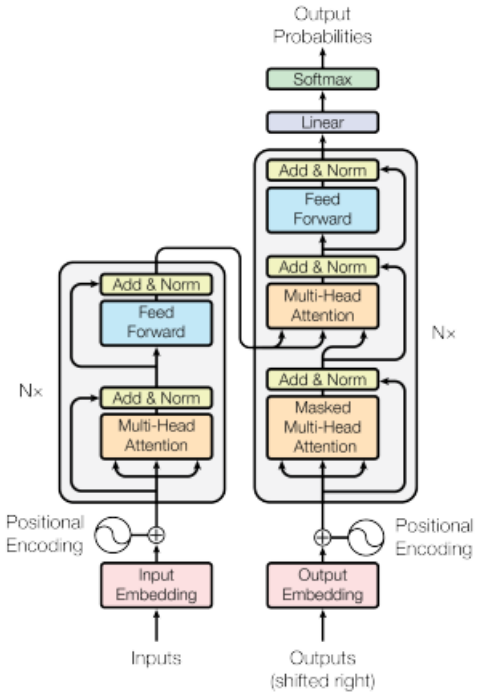

Figure 9: Transformer model architecture (Vaswani et al., 2017)

4.2.5 BERT

BERT, which stands for "Bidirectional Encoder Representations from Transformers", is a type of neural network architecture based on the transformers of Vaswani et al. (2017) explained in the previous section. It has been used for several natural language processing tasks, including text classification (Devlin et al., 2018). *BERT* is designed to process the input self-supervised by predicting missing or randomly permuted words in a sentence (Devlin et al., 2018). This allows *BERT* to learn relations between words in a sentence and to understand the meaning of words in the context of a sentence instead of only the particular meaning of a word.

In text classification, *BERT* can be used to encode the input text into a fixed-length vector representation that represents the meaning of the text and the relationships between the words. This vector representation can then be fed into a classifier, for example, a linear layer, to make predictions about the input class. *BERT* is helpful for text classification tasks

because it can represent relationships between words and their meanings and generalize well to unseen data (Devlin et al., 2018).

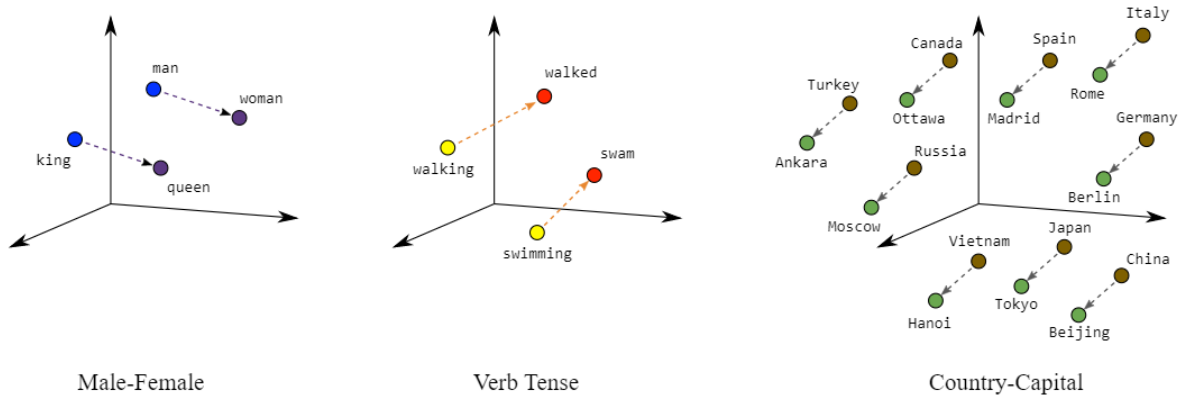

Figure 10: Word embeddings (GoogleDevelopers, 2022)

One key advantage of $BERT$ is that it is a pre-trained model, meaning it has already been trained on a large dataset. A pre-trained model is preferred since the model already knows the syntax and semantics. During pre-training, a sentence embedding is created of the keyword, which captures the meaning of a whole sentence. Figure 10 shows an example of word embeddings, similar to sentence embeddings. In the figure, similar words are closer to each other than more different words. The similarity of words, or sentences, can be seen as the distance between the words or sentences. Thus, if words or sentences are similar, it knows that they can be represented in the same way, and thus the model should only be fine-tuned to the type of data used and the corresponding wanted output, thus saving much training time. The model can be fine-tuned for specific tasks by adding specific layers on top of the $BERT$ layers, and fine-tuning learns the model which kind of data is used and what output is expected. The pre-trained base model can be used for a wide range of NLP tasks and achieve substantial results with minimal labeled training data, as it already knows the syntax and semantics of sentences. During the remaining part of the paper, the pre-trained $BERT_{base}$ model, trained by Devlin et al. (2018), is used, which consists of 12 layers, a hidden size of 768, 12 self-attention heads, and $110M$ parameters. The architecture of the $BERT_{base}$ model is displayed in figure 11, where the second box is a transformer layer. In figure 9, the used transformer structure, the encoder, is displayed on the left while the right side, the

decoder, is not used in a $BERT$ model. Because of the transformer layers, $BERT$ is called a transformer model. It can process input sequences in parallel and takes care of dependencies between words in long sentences more effectively than other models, for example, recurrent neural networks (RNNs) (Devlin et al., 2018). This makes $BERT$ suitable for tasks that need an understanding of the context and structure of sentences or documents, such as text classification.

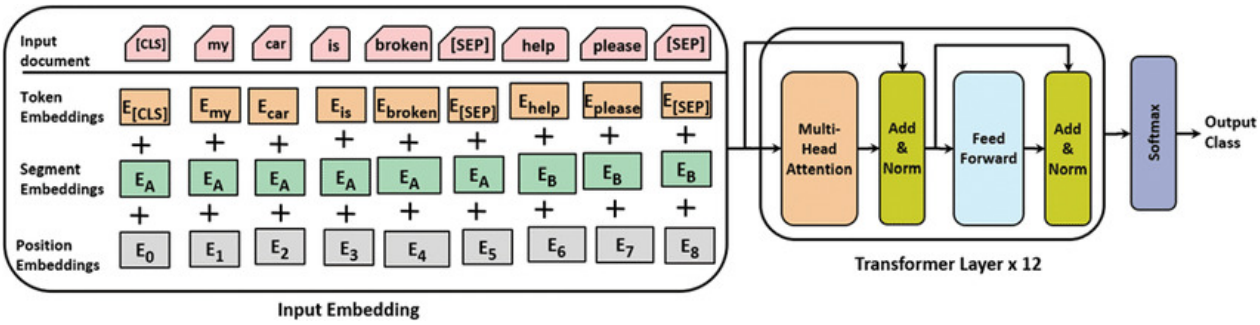

Figure 11: $BERT_{base}$ model architecture (Khatoon et al., 2021)

To adjust the $BERT$ model to the kind of data used, the model is fine-tuned. When willing to fine-tune for a specific task the available data, including the correct labeling, can be inserted into the model as the in and output of the task. This labeled data is used to adjust the model to the corresponding NLP task.

However, since $BERT$ was designed, many models have been produced to improve the performance of $BERT$. For example, $RoBERTa$ (Liu et al., 2019) was designed. $RoBERTa$, "Robustly Optimized $BERT$ Pretraining Approach", is a variant of the $BERT$ model researchers at Facebook AI developed. It is based on the $BERT$ architecture and shares many of its characteristics, but it was specifically designed to improve upon the pre-training procedure of $BERT$. $RoBERTa$ was trained on a larger dataset and for a more extended period of time than $BERT$. It was also designed to be more robust to various forms of data disruption, such as word masking and sentence shuffling (Liu et al., 2019). The model is trained using dynamic masking where the masking pattern is generated every time a sequence is inputted to the model, full-sentences without NSP (Next Sentence Prediction) loss, larger

mini-batches, step sizes, and learning rates and a different kind of masking (Liu et al., 2019).

Sequentially, the fine-tuning of $RoBERTa$ can be done by adding task-specific layers on top of the pre-trained $RoBERTa$ layers as with BERT. It has achieved state-of-the-art results on various natural language processing tasks, including text classification (Liu et al., 2019). $RoBERTa$ is outperforming $BERT$, $XLNet$, and $DistilBERT$, while $XLNET$ is performing better than $BERT$, $DistilBERT$ is performing worse on accuracy but is faster (Liu et al., 2019). In addition to the strong performance, $RoBERTa$ has the advantage of being relatively fast to train compared to $BERT$, making it a popular choice for many NLP applications.

All the models mentioned before are pre-trained in the English language and thus would not perform as well in different languages as it does in English. Due to the lack of performance in different languages, these models are also designed for other languages, for example, Dutch. The Dutch version of $BERT$ is called $BERTje$ (de Vries et al., 2019) and is pre-trained on books, TwNC (news corpus), SoNaR-500, Web news, and Wikipedia. Furthermore, the Dutch version of $RoBERTa$ is called $RobBERT$ (Delobelle et al., 2020). The model has a different tokenizer and was trained on the OSCAR corpus, which is much larger than the data used for pre-training $BERTje$, 12 GB versus 39 GB.

Nevertheless, as language usage keeps evolving, a new model of $RobBERT$ was designed to take care of the changes in the Dutch language over the last three years (Delobelle et al., 2022). The model is pre-trained on a new version of OSCAR with language up until January 2022. This new model is called $RobBERT-2022$, designed by Delobelle et al. (2022). The results of this model are better on more recent datasets and have a similar performance on the older datasets (Delobelle et al., 2022).

Concluding, a selection is made to use in this paper. For English datasets as a comparison, the original $BERT$ (Devlin et al., 2018) model is used, and the $RoBERT$ (Liu et al., 2019) model because it performs better than $BERT$. For the Dutch datasets, models similar to

the English models are used, which are $BERTje$ (de Vries et al., 2019), and the renewed $RobBERT$ (Delobelle et al., 2020) named $RobBERT-2022$ (Delobelle et al., 2022).

4.2.6 BERT-like Models

Multi-output Neural Networks using one of the custom loss functions, with the hierarchical part given in equation 10 and 12, could be improved by learning more about the context of the sentences. Since $BERT$ is a bidirectional encoder, it has a better sense of the meaning of the whole sentence and not just word by word (Devlin et al., 2018), as stated in section 4.2.5. The classic $BERT$ model is trained in the English language. However, not all data sets are in English. As most of the data sets available for classification are in English or Dutch, this paper will focus on those two languages.

The Dutch version of $BERT$ (Devlin et al., 2018) is called $BERTje$ (de Vries et al., 2019). As explained in section 4.2.5 $RoBERTa$ (Liu et al., 2019) is performing better than $BERT$ in most cases. Hence, the performance of $RoBERT$, and the Dutch renewed version $RobBERT-2022$ (Delobelle et al., 2022), are compared to the $BERT$ models.

The implementation of the $BERT$-like models is approximately equal to the model architecture displayed in figure 8. Different from the multi-output classification architecture, the embedding layer is replaced by one of the $BERT$-like models. The new architecture is displayed in figure 12. Furthermore, the settings of the model are displayed in table 12 of the appendix. The steps performed for a $BERT$-like model are only fine-tuning steps as a pre-trained model is used, the $BERT_{base}$ model. Further information on $BERT$-like models and the other layers was explained in sections 4.2.5 and 4.2.1.

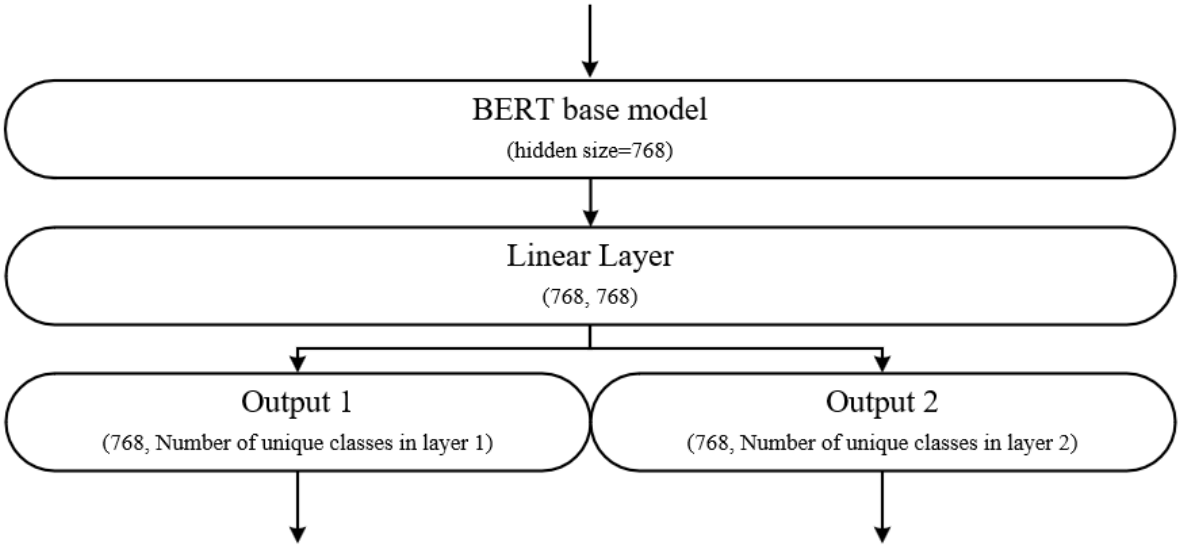

Figure 12: Model Architecture of *BERT* multi-output neural network

4.3 Evaluation Metrics

There are several options to measure how a model performs; some options are explained in this section. The metrics can be divided into two groups. On the one hand, the metrics designed for a flat model. On the other hand, metrics designed for hierarchical classification.

4.3.1 Precision, Recall and F_1-score

Predictions can be classified into four groups for binary classification, as displayed in table 3. Based on these results, the goodness of the model is determined. A perfect model would have no false positive (FP) predictions and false negative (FN) predictions.

	Actual Positive	**Actual Negative**
Predicted Positive	True positive (TP)	False negative (FN)
Predicted Negative	False positive (FP)	True negative (TN)

Table 3: Confusion Matrix

Text classification most often uses the following metrics: F_1-score, accuracy(acc), precision(P), and recall(R). In binary classification, this could be calculated in the following way (Opitz

and Burst, 2019):

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

$$F_1 = 2\frac{P \cdot R}{P + R}$$

Accuracy measures the ratio of correct predictions over the total amount of predictions. Precision shows how many of the actual positives are positively predicted (TP). Recall counts how many of the positively predicted are indeed positive. Lastly, the F_1-score uses precision and recall to calculate a harmonic mean(Lipton et al., 2014). Accuracy measures all classes equally important, while F_1-score better measures the incorrectly classified classes. Moreover, accuracy can be used when the dataset is balanced, and F_1-score is a better choice when the dataset is imbalanced.

4.3.2 Hierarchical Evaluation Metrics

Different metrics are designed to analyze the performance of a hierarchical multi-label classification model than the metrics specified in the previous section. Accuracy, Precision, and Recall can be applied to a hierarchical classification using a flat model. In hierarchical classification, the functions would be applied separately to the different layers of the hierarchical tree. However, they do not consider how far the prediction is from the actual label in the hierarchical tree. Kiritchenko et al. (2005) designed a precision and recall function applicable to hierarchical classification. Displayed below are hP (Hierarchical Precision) and hR (Hierarchical Recall), which lead to hF, giving precision and recall the same weight (Silla and Freitas, 2011)

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|}$$

$$hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|}$$

$$hF = \frac{2 \cdot hP \cdot hR}{hP + hR}$$

where C_i is the actual class and C'_i the predicted class. Taking into account the hierarchy, the ancestors are added to the classes. This leads to $\hat{C}_i$ being the actual path and $\hat{C}'_i$ the predicted path.

These hierarchical evaluation metrics are based on three requirements (Kiritchenko et al., 2005). First, if a prediction is partially correct, it should be penalized less (Kiritchenko et al., 2005). This means that if part of the path with segments in the hierarchical tree corresponds to the correct path of segments, it should be penalized less. Second, suppose the prediction is further away from the actual leaf node. In that case, it should be penalized more (Kiritchenko et al., 2005) because a segment almost equal to the predicted segment is not as bad as a completely different one. Lastly, an incorrect prediction at a higher level in the hierarchy should be penalized more than an incorrect prediction at a lower level (Kiritchenko et al., 2005).

4.4 Model Evaluation

The model performance can be evaluated using some of the metrics stated in the previous section in combination with improved methods specific to this paper's problem. First, the baseline models to compare the new models with are stated. Lastly, the evaluation metrics used in this paper are chosen and explained.

4.4.1 Baseline and Benchmark models

Are the newly designed models performing better than the existing methods? The models stated in this paper should be compared to the results of the current methods. Hence, the existing techniques are tested using the same data used in this paper. The existing methods can be split into two groups. First, the methods that do not account for the model's hierarchy

are flat models, just classifying all levels in the hierarchy together. Second, existing methods that consider the hierarchy, using global or local methods, as explained in section 4.1.2.

First is the model group that does not consider the hierarchy. For this group, three models are selected, the multi-branch neural network, a $BERT$ or $BERTje$ model, and the $RoBERTa$ or $RobBERT-2022$. The choice in the $BERT$ and $RoBERT$ models depends on the dataset's language. The second group consists of models that take the hierarchical structure into account. Two existing models are used as comparisons, one using local classifiers per parent node, HiClass (Miranda et al., 2021), and the other using the existing loss functions of equations 3 and 8 in combination with $RoBERTa$ or $RobBERT-2022$. These models are compared to each other and to the models designed during this paper using $RoBERTa$ or $RobBERT-2022$ in combination with the two new loss functions.

4.4.2 Evaluation Metrics

Determining which model performs better should be based on a metric. The most popular metric in text classification is F_1-score, explained in section 4.3.1 because most text classification datasets are imbalanced. The datasets used in this paper are also imbalanced, as explained in section 3.2.2, F_1-score is thus the best metric for this data. The requirements for a hierarchical evaluation metric, stated in section 4.3.2, do not add value to the problem in this paper. Because, for the data used in this paper, the evaluation metrics per layer are equally important, and the depth of the hierarchical trees is low, mainly two and a maximum of three hierarchical layers are present. Thus, the standard F_1-score is used as the evaluation metric.

Another metric used in this paper is based on the mediavalue and search volume, and these values help decide about segments to focus on during advertising. As explained in section 2.1.3, the decision depends on the size of the search volume and mediavalue. Suppose the search volumes are approximately equal and the mediavalue of one segment is smaller than another. In that case, focusing on the one with the lower mediavalue for paid advertising is better. Hence, the classification can be evaluated by investigating the difference in search volume and mediavalue per segment compared to the values in the "true" model. From the

difference in search volume, it can be concluded that the number of searchers on a segment where predicted to be lower or higher than the actual number of searchers. If one segment was predicted too low, more searchers saw the link than predicted. On the other hand, if the predicted value was too high, fewer searchers saw the link than predicted. Thus, if the classification is incorrect, a different amount of searchers is targeted at each segment than expected. If the classification was incorrect, the advertiser could also focus on the wrong segment based on the supposed search volume. The difference between the true model and the predicted model in mediavalue may lead to a wrong indication of the required budget for advertising on a specific segment. Hence, it could be focussing on a segment that, in reality, is more expensive than predicted while another segment is predicted to be expensive but is cheaper. Thus, the model performs better if the difference between the predicted and actual search volume and mediavalue is smaller.

In addition, the metric based on search volume and mediavalue compare the different models based on how close the predicted mediavalue and search volume are to the actual values per segment. The Mean Squared Error (MSE) is one option to measure how close the predicted values are. However, MSE does not consider the total value of the search volume or mediavalue. Hence, if a segment with a search volume of ten is predicted as zero, the MSE is equal to the MSE of a segment with a search volume of $10,000$ and was predicted as $10,010$. Therefore, as a difference is preferred between the two cases mentioned before, Mean Absolute Percentage Error ($MAPE$) is selected, which is displayed in equation 13 below.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{13}$$

5 Results

In this section, the models explained in section 4 are applied to the datasets mentioned in section 3. All results in this section are based on running the models on an "Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz" using the settings explained in section 4. First, the results per model are stated and explained. Second, it is shown what improvements the model has compared to manually classifying. Lastly, the possibility of misclassifications is shown.

5.1 Results per Model

In table 4, the used models are stated together with the abbreviations used in the successive tables. More information on these models is explained in the previous section. Running these models on the different datasets gave the results displayed in table 5. In this table, $S1$ and $S2$ refer to segment 1 and segment 2. Furthermore, "Hierarchy" shows the number of keywords where the predicted first segment is not a parent of the predicted second segment. The "Incorrect" column indicates the number of incorrectly classified segments. Lastly, the time column shows the number of seconds needed to classify the test set.

Model	Type	Abbreviation
Multi-output neural network	Flat classification	MONN
HiClass	Local Classifier per Parent Node	HiClass
BERT/BERTje model		BERT(je)
RoBERTa/RobBERT-2022 model		Ro(b)BERT
RoBERTa/RobBERT-2022 + custom loss	Loss function equation (8)	Ro(b)BERTcl
RoBERTa/RobBERT-2022 + custom loss	Loss function equation (10)	Ro(b)BERTclNew
RoBERTa/RobBERT-2022 + custom loss	Loss function equation (11+12)	Ro(b)BERTclNewSV

Table 4: Comparison models with further used abbreviations

Table 5: Results per dataset and model

		F_1			MAPE SV		MAPE MV		MAPE SV and MV			
	Model	S1	S2	Mean	S1	S2	S1	S2	Mean	Hierarchy	Incorrect	Time (s)
Dataset 1	HiClass	0.9703	0.8549	0.9126	0.0978	0.2886	0.0611	0.6318	0.2699	**0**	186	0.049
	MONN	0.9727	0.8968	0.9348	0.0112	0.1024	0.0079	0.1460	0.0669	28	181	**0.001**
	BERT	0.9922	0.9274	0.9598	**0.0037**	0.5506	**0.0036**	0.2539	0.2030	8	115	196.363
	Ro(b)BERT	0.9906	0.9492	0.9699	0.0057	0.1141	0.0073	0.2567	0.0959	8	88	177.969
	Ro(b)BERTcl	0.9896	0.9512	0.9704	0.0051	0.0689	0.0060	**0.0687**	**0.0372**	6	84	174.783
	Ro(b)BERTclNew	**0.9923**	**0.9633**	**0.9778**	0.0121	0.1780	0.0304	0.5113	0.1829	8	**74**	174.219
	Ro(b)BERTclNewSV	0.9853	0.9553	0.9703	0.0130	**0.0317**	0.2347	0.4846	0.1910	8	90	177.448
Dataset 2	HiClass	0.7186	0.5465	0.6326	0.1756	0.3743	0.1166	0.5085	0.2938	**0**	410	0.037
	MONN	0.7682	0.5805	0.6743	0.2440	0.4995	0.1891	0.3288	0.3154	46	418	**0.001**
	BERT	0.8791	0.7666	0.8228	0.1112	**0.3199**	0.0734	0.4304	0.2337	14	186	102.230
	Ro(b)BERT	0.8675	0.7820	0.8247	0.2200	0.8933	0.0939	1.0642	0.5679	6	164	101.131
	Ro(b)BERTcl	0.8770	0.7838	0.8304	0.1335	0.3426	**0.0675**	0.3456	0.2223	15	169	99.978
	Ro(b)BERTclNew	**0.8827**	0.7732	0.8280	**0.1014**	1.8823	0.0873	2.1527	1.0559	11	161	101.283
	Ro(b)BERTclNewSV	0.8630	**0.8388**	**0.8509**	0.1151	0.0984	0.3459	**0.2894**	**0.2122**	20	**160**	100.285
Dataset 3	HiClass	0.9367	0.8142	0.8755	0.1128	0.1198	0.1154	0.1198	0.1169	**0**	124	0.027
	MONN	0.9168	0.8013	0.8590	0.0873	0.1451	0.0593	0.1666	0.1145	21	145	**0.001**
	BERT	0.9716	0.8576	0.9146	0.0816	0.5551	0.0683	0.6401	0.3363	5	78	68.427
	Ro(b)BERT	0.9781	0.9351	0.9566	0.0633	0.2156	0.0646	0.2248	0.1421	2	61	69.159
	Ro(b)BERTcl	0.9712	0.9058	0.9385	0.0830	0.1131	0.0810	0.1297	0.1017	9	71	69.103
	Ro(b)BERTclNew	0.9794	**0.9549**	**0.9672**	0.0629	0.1376	0.0640	0.1815	0.1115	**0**	**50**	69.010
	Ro(b)BERTsclNewSV	**0.9797**	0.8985	0.9526	**0.0425**	**0.0445**	**0.0820**	**0.0644**	**0.0584**	6	56	72.770

Several things can be observed in this table when comparing the different models and datasets. Beginning at the left-hand side of the table, the F_1-score. From these scores, all $BERT$-like models perform better than the multi-output neural network and the local classification method, $HiClass$. Afterward, $BERT$ is compared to the $Ro(b)BERT$ model. In all three datasets, the $Ro(b)BERT$ model outperforms the $BERT$ model. Lastly, it can be examined if the models considering the hierarchy perform better than the basic model. All three hierarchical models outperform the standard $Ro(b)BERT$ model based on the F_1-score. Hence, it does improve when taking the hierarchy into account.

Choosing between the existing and new loss functions is less straightforward because, in datasets 1 and 3, the model with the new custom loss function performs better. However, datasets 2's model with the existing custom loss function performs better. Table 6 displays the averages of the F_1-scores and $MAPEs$'. From this can be seen over the three different datasets, based on the F_1-scores, the $Ro(b)BERT$'s with the new custom loss functions are performing better than the other tested models. Moreover, the $Ro(b)BERTclNew$ model has the least incorrectly classified keywords in two out of three datasets, and in the other dataset, $Ro(b)BERTclNewSV$ has the least incorrectly classified keywords.

Model	Average F_1	Average MAPE
HiClass	0.8069	0.2269
MONN	0.8227	0.1656
BERT	0.8991	0.2577
Ro(b)BERT	0.9171	0.2686
Ro(b)BERTcl	0.9131	**0.1204**
Ro(b)BERTclNew	0.9243	0.4501
Ro(b)BERTclNewSV	**0.9246**	0.1538

Table 6: Averaged results

Next, the focus is on the $MAPE$ displayed in the table. The best values, shown in bold, are present in almost all models. Thus, they are more distributed over the models than with the F_1-scores. Nevertheless, the mean $MAPE$ over dataset 2 and 3 is the least for $Ro(b)BERTclNewSV$, but not in the first dataset where the $Rob(b)BERTcl$ model is performing best based on average $MAPE$. Focusing on the averages over the dataset in table 6, the $Ro(b)BERTcl$ model outperforms the new model made to minimize the $MAPE$. This happens due to the worse performance of the $Ro(b)BERTclNewSV$ model on the first dataset. The newly designed $Ro(b)BERTclNewSV$ is performing better than the model it is an extension to, $Ro(b)BERTclNew$.

Lastly, the inference time of the different models. The less-performing models, $HiClass$ and $MONN$, need the least time and are almost constant while the number of keywords increases, and the remaining models need approximately equal time. Figure 16 of the appendix shows the time evolution, dependent on the number of keywords. This figure points to the lengths of the test sets of the different datasets. Meaning there are points at 879, 1,790, and 1,319. From the figure, $HiClass$ and $MONN$ have an approximately horizontal line, and the other models are linearly increasing until the 1,319 keywords checkpoint. After this point, the time increases more rapidly for the $BERT$-like models.

Furthermore, one of the challenges of the model is to perform well on short keywords containing little information. In figure 17, the distribution of the incorrectly classified keywords of $Ro(b)BERTclNew$ is displayed, and the incorrectly classified keywords of $Ro(b)BERTclNew$ as part of the distribution of all keywords. From the top row of this figure, the distribution of incorrectly classified keywords of $Ro(b)BERTclNew$ does not differ much from the distribution of all keywords, but comparing it is performing slightly worse on the keywords with only one or two words and better on the longer keywords. From the lower row of the figure can be seen it is still performing well in classifying shorter keywords as well as longer keywords. The figures of the other designed model, $Ro(b)BERTclNewSV$, are similar.

5.2 Costs and Benefits

Performing the classification with a model instead of manually has specific costs and benefits explained in this section. Using a model to classify the keywords leads to decreased time spent on keyword research and, thus, a cost reduction. A specialist's classification time can be spent on different tasks. Keyword research takes approximately 15 hours for 5,000 keywords. At the same time, the time needed for keyword research is roughly 70 percent of classification. Hence, the classification time corresponding to these numbers is approximately 7.56 seconds per keyword; the calculation is below. However, the time needed for keyword research typically also depends on the number of segments and the list and length of possible segments. The other dependencies are beyond the scope of this paper.

$$\frac{15 * 3600 * 0.7}{5000} = 7.56$$

As the results in section 5.1 never make a perfect classification, it is still essential to check whether the model did a good job classifying and correct possible mistakes. For this, assume approximately one hour is needed to check part of the classification. The cost of performing the classification by a model is the running time, assuming the model is already trained on similar data. However, as displayed in table 5, this is negligibly small compared to the time needed for manual classification. Furthermore, the specialist can perform other tasks while the model is running; hence this is not a real cost to consider. What still should be done by the specialist when the classification is done is the classification check, as explained before. Compared to manual classification, it is advantageous that the classification is faster than the manual classification.

Focusing on manual classification, it costs the number of hours needed for the classification times the price. However, an advantage of manually classifying is that the results are more accurate than those reproduced by the model. Hence, the checking time is, in this case, not needed.

Comparing the costs and benefits using dataset 1, containing 11,932 keywords, as a comparison model. The time needed to perform classification on this dataset manually would take

approximately 25 hours. Hence, the time benefit of performing it with a model is 24 hours. Let us assume the rate of a specialist is 125 euro per hour; this would equal an extra earning of 3,000 euro. So, an earning of 3,000 euro would, in this case, equal a reduction in the F_1-score of approximately two percent.

5.3 Misclassifications

From the incorrectly classified keywords, there can be doubt about the correctness of the true classification or that different classifications could be correct as well. This section will focus on the type of misclassifications using the output first dataset because this is the only English dataset used. The following paragraphs show some of the different pitfalls the classifications can have.

Some keywords need to be longer to find clear corresponding segments. For example, the true segments of ”kitchen brush” are ”kitchen utensils” and ”baking utensils”. However, these segments refer to what is also called a pastry/basting brush, but could also be meant as a dish brush. If it was meant as a dish brush, the segments should be ”dishwashing” and ”dish brush”; thus, the classification would be incorrect due to an unclear formulation.

Table 7 displays several incorrectly classified keywords. Multiple segments could be correct for those keywords, true and predicted. Most of these multiple options arise from multiple segments called inside the keywords. For example, the last keywords in the table state ”white” and ”metal” inside the keyword, but it is unknown which of the two segments is most important, and the keyword should thus be classified into.

Keyword	True segment 1	Predicted segment 1	True segment 2	Predicted segment 2
refrigerator wipes	cleaning up	cleaning up	kitchen	cleaning wipes
white metal trash can with lid	waste bins	waste bins	material	bins with lid
wall mounted waste bins with lid	waste bins	waste bins	bins with lid	wall mounted bins
bathroom scale memory	bathroom scale	bathroom scale	general	additional features
white metal trash can	waste bins	waste bins	material	colour

Table 7: Incorrectly classified classifications which have multiple correct classes in a layer

Some classifications are incorrect. Two incorrect classifications are displayed in table 8. The first keyword is predicted as displayed in the predicted columns in the table by five models, and one time ”correct”. However, the true segment is incorrect, as a clothes rack is not used for laundry or drying. Furthermore, the second keyword is translated from dataset 2. The second segment was, in this example, predicted four times, as displayed in ”Predicted segment 2”. The true segment, in this case, is not incorrect, but the predicted segment is more accurate.

Keyword	True segment 1	Predicted segment 1	True segment 2	Predicted segment 2
clothes rack stainless steel	laundry	clothing storage	drying rack	clothing rack
necklace for ash	remembrance products	remembrance products	jewellery	necklace

Table 8: Incorrect true classifications

Nevertheless, most incorrectly classified keywords are ”really” incorrect. For example, an electric brush holder should be classified as ”bathroom” and ”toothbrush holder”, but it was classified as ”laundry” and ”toiletbrush”. A large amount of incorrect classified keywords are keywords containing the word of a segment but which is not the correct segment. For example, the ”perforated spoon” subsegment is ”spatulas” but is predicted as ”spoon”. This is a more common problem as search engine users often exchange these terms with different meanings.

6 Discussion

Following the results stated in the previous section in tables 5 and 6, the newly designed models are performing well on the datasets. However, there are several remarkable observations. These observations can be split into two parts: the observations related to the evaluation metrics and some observations related to the hierarchy and incorrect classifications in the results.

First, the observations of the F_1-scores in tables 5 and 6 are evaluated. From these F_1-scores, the $Ro(b)BERTclNew$ and $Ro(b)BERTclNewSV$ perform better than the other tested models based on the F_1-score. However, there are two remarkable observations in these scores. First, while comparing $Ro(b)BERTcl$ with $Ro(b)BERTclNew$, the new custom loss function results in a better F_1-score, but not in dataset 2. Second, concerning the previous observation, the $Ro(b)BERTclNewSV$ results in a better F_1-score in dataset 2. At the same time, this is not the case for the other datasets, but it does result in a better average F_1-score in table 6. This could have several explanations, for example, the composition of the dataset as this dataset consists of many rarely used and complex words. However, the actual reason behind this cannot be investigated due to a lack of datasets where this occurs. Furthermore, the best F_1-score of the second dataset is much lower than the F_1-scores of the other datasets. This can come due to the structure of the data, but as mentioned before, this cannot be investigated with certainty because it is only seen in the results of one dataset.

Continuing on dataset 2, comparing the $Ro(b)BERTcl$ with the $Ro(b)BERTclNew$. It can be observed that the F_1-score of the new loss is worse, but the amount of keywords with an incorrect hierarchy and the amount of incorrectly classified keywords is better. The difference in the number of keywords with incorrect hierarchy can be because the F_1-score is not dependent on hierarchy. The lower amount of incorrectly classified keywords can be explained by how the number of incorrectly classified keywords and the F_1-score are defined. Referring to section 4.3.1, the number of incorrectly classified keywords is the sum of the false positive and false negative predictions. However, the F_1-score also depends on the true positive predictions, which causes the difference in best, based on F_1-score, and best, in

the lowest number of incorrectly classified keywords. Hence, this value cannot conclude if the model performs better than another but does indicate the number of incorrect predictions.

Looking at the $MAPE$ results, it can be observed that the $MAPE$ is minimal in different models every time. For this reason, the new loss function was used on the $Ro(b)BERTclNew$ model. This change in loss function positively influenced the average $MAPE$. Although it was not performing better than all tested models, it performed much better than the $Ro(b)BERTclNew$ model it applied to. However, as the other models are not learning on $MAPE$, it is "fortunate" to incorrectly classify the keywords with a lower search volume and mediavalue instead of the high search volumes and mediavalues. This also applies to the other $MAPE$ columns, where the lowest values are divided over different models.

Generally, it can be observed from the data that if the F_1-score is high, the number of wrong hierarchies and incorrectly predicted observations is low. However, this does not correspond to the lowest $MAPE$. The definition of $MAPE$ can explain this. $MAPE$ considers how far the predicted search volume and mediavalue are away from the original, while this is not used in the other metrics. Hence, it can be the case that one keyword with an extensive search volume or mediavalue is incorrectly predicted. However, the model did perform well for the other keywords. The F_1-score and the number of keywords with a correct hierarchy and amount of incorrectly predicted can still be good. Similarly, in all three datasets, the $Ro(b)BERTclNew$ has fewer incorrectly classified keywords than $Ro(b)BERTcl$ and the $MAPE$ is only smaller in the third dataset.

Furthermore, some points may cause varying results. For example, the small amount of data to train on. Another point to consider is the data quality, which includes misclassifications, rare words, and words out of different languages than trained on, which influence the evaluation metrics in the results.

7 Conclusion

7.1 Summary and Conclusion

This paper aimed to design a good model for hierarchical keyword classification such that this process would no longer have to be performed manually. The models should classify each keyword in the corresponding main segment, subsegment, and if preferred, subsubsegment. This should be done as well as possible, depending on the evaluation metrics. This paper uses the often-used F_1-score and the $MAPE$, which provides an evaluation metric considering the importance of search volume and mediavalue corresponding to keywords.

First, an elaborate explanation was given of the background of the problem. Starting with a general explanation of search engine marketing and the options within search engine marketing, leading toward search engine advertising and optimization. With these two parts, Google Ads can perform the best task when using Google as a search engine. To achieve this, a business should focus on the right keywords. This decision can be made based on the search volume and mediavalue corresponding to specific keywords; keyword research helps with this task. Keyword research looks at which keywords could improve visibility and find possibly interested customers for the business based on the values mentioned before.

Afterward, when the background of the problem was known, the focus was on recent developments in the field of text classification. As most researchers focus on standard text classification, the biggest and more developments are in this field, and less is hierarchical text classification as this is more specific. Furthermore, text classification has much in common with hierarchical text classification. Hence, the biggest developments in text classification can be changed to apply to hierarchical text classification. From these developments in text classification, it could be concluded that there is an increase in the usage of transformer-based language models such as $BERT$. In addition, these models were also performing better on NLP tasks than the methods used before.

Hence, models were designed using $BERT$-like models. However, as these models would work for classic classification and not for hierarchical text classification, a newly designed loss function considering the hierarchy was used to fine-tune the model hierarchy in the data. Besides, an addition to this loss function is designed in this paper to consider the difference in search volume between the different keywords. These models with new loss functions were tested against several different models.

From the results given in this paper, two different conclusions can be drawn, one based on the F_1-score as an evaluation metric and the other where the $MAPE$ is seen as a vital evaluation metric. First, using F_1-score as the evaluation metric. From this metric, both newly designed models perform better than the other model, where the average F_1-score over these models is almost equal. Hence, both models would have an additional value for a hierarchical text classification problem using F_1-score as the evaluation metric and, second, using $MAPE$ as the evaluation metric. The model with the loss function made to minimize the $MAPE$ is performing better on two out of three datasets. Hence, when willing to reduce the $MAPE$, the new loss function is a good option as it performs significantly better on two out of three datasets. Further recommendations to improve the average $MAPE$ are given in the next section. Moreover, one of the challenges in this paper was for the model to perform well on short keywords, which thus contain less context to classify with certainty. The results show that both models perform well on all used datasets' shorter and longer keywords.

7.2 Recommendations and Further Research

The research conducted in this paper had several limitations which could be changed in further research. There is also possible future research related to the assumptions made during this paper. These are stated and further explained in the upcoming paragraphs.

Focussing on the changes connected to limitations in this research. For the results, the models were tested on three different datasets. However, to achieve more reliable decisions, the models should be tested on more datasets because from this small number of datasets cannot be concluded why it is performing better on one dataset than on another with certainty. The

models in this paper are fine-tuned on the existing data, but if the classification is performed for a business in a different market, it should first be fine-tuned on part of the data. This is not ideal, as some provided data should still be classified. Hence, it would be a good idea to perform further research on models which require less training data or unsupervised models. Two possible unsupervised options would be hierarchical clustering and Latent Dirichlet Allocation (LDA). This would reduce or remove the problem of fine-tuning specific markets and needing data in cases the model was not trained on. Not having enough data could also be resolved using methods to generate more data, such as making minor changes to the data and adding it as new data or using a generative adversarial network (GAN) to generate similar data. Furthermore, the clustering methods help track down where possible misclassification can occur with a higher probability because they are more likely located at the boundary of clusters when visualized. It would also make the model less dependent on manually classified data. Another case is when a new business would like a classification, but the business is in the same market as a business where classification was known. In this case, it is recommended to use a similar fine-tuned model for the new business. Moreover, a different choice on the model could be made since, in this paper, a $BERT$-like model is used, which should be trained and fine-tuned on much data. Nevertheless, as explained in section 2.2, new models were designed using few-shot or zero-shot learning, requiring much less data. In further research, there could be new models with better performance, or these models might be improved.

Furthermore, there were several assumptions made to be able to have enough training samples and to avoid overfitting. If too many empty values were in a hierarchical layer, they were removed to prevent overfitting. Segments with too few keywords were also removed because they would not be suitable for creating the training and test set. In further research, a solution for this should be found, different than removing the cases, to keep the dataset as large as possible and keep performance high on the small segments. However, these small segments often have a small search volume, so the effect would be negligible small when looking at the $MAPE$ metric.

In addition, when doing further research, it is essential to consider that keywords can have more than one possible segment. Hence, it could be an option to look at multi-label models, where keywords can be allocated to multiple segments instead of only one. However, most of the keywords only have one appropriate segment, and thus it is important to contemplate that it only has to change for a few keywords.

Moreover, further research could be done on the $BERT$-like models used in this paper. Because in this paper, pre-trained versions of the $BERT$-like models are used. However, training these models self can also bring additional value instead of using the pre-trained model. There are two main options to change this. First, the data used to train the model can be altered to data more applicable to the problems the model is used for. Second, the architecture of the model can be changed. In the pre-trained model, a specific order of layers is used after the encoder layers, and these layers could be changed to improve the model further.

Lastly, from the results, the loss function made to minimize the $MAPE$ is not resulting in the lowest $MAPE$. Hence, further research could add it in another way to the loss function or include it in the model's training. Alternatively, the search volume and mediavalue could be taken into account, training the model differently from adding the hierarchy through the loss function to make the model more aware of the importance of different keywords compared to other keywords.

Concluding, there are several ways to continue and improve this research. Shortly, consider models which require less data, improve model assumptions such that some of these do not have to be made anymore, think about models taking care of multiple possible segments, change model structure for further improvements, and lastly, design new methods to improve models based on the $MAPE$ metric.

References

Abou Nabout, N., & Skiera, B. (2012). Return on quality improvements in search engine marketing. *Journal of Interactive Marketing*, *26*(3), 141–154.

Barham, P., Chowdhery, A., Dean, J., Ghemawat, S., Hand, S., Hurt, D., Isard, M., Lim, H., Pang, R., Roy, S., et al. (2022). Pathways: Asynchronous distributed dataflow for ml. *Proceedings of Machine Learning and Systems*, *4*, 430–449.

Barutcuoglu, Z., Schapire, R. E., & Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, *22*(7), 830–836.

Bianchi, T. (2022a). *Google: Annual advertising revenue 2001-2021*. https://www.statista.com/statistics/266249/advertising-revenue-of-google/

Bianchi, T. (2022b). *Google: Global annual revenue 2002-2021*. https://www.statista.com/statistics/266206/googles-annual-global-revenue/

Borges, H. B., Silla Jr, C. N., & Nievola, J. C. (2013). An evaluation of global-model hierarchical classification algorithms for hierarchical classification problems with single path of labels. *Computers & Mathematics with Applications*, *66*(10), 1991–2002.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Decarolis, F., Goldmanis, M., & Penta, A. (2020). Marketing agencies and collusive bidding in online ad auctions. *Management Science*, *66*(10), 4433–4454.

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

Delobelle, P., Winters, T., & Berendt, B. (2022). Robbert-2022: Updating a dutch language model to account for evolving language use. *arXiv preprint arXiv:2211.08192*.

Deng, Z., Peng, H., He, D., Li, J., & Yu, P. S. (2021). Htcinfomax: A global model for hierarchical text classification via information maximization. *arXiv preprint arXiv:2104.05220*.

Deniz, E., Erbay, H., & Coşar, M. (2022). Multi-label classification of e-commerce customer reviews via machine learning. *Axioms*, *11*(9), 436.

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

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

Freitas, A., & Carvalho, A. (2007). A tutorial on hierarchical classification with applications in bioinformatics. *Research and trends in data mining technologies and applications*, 175–208.

Gao, D., Yang, W., Zhou, H., Wei, Y., Hu, Y., & Wang, H. (2020). Deep hierarchical classification for category prediction in e-commerce system. *arXiv preprint arXiv:2005.06692*.

Gilfoil, D. M., Aukers, S. M., Jobs, C. G., et al. (2015). Developing and implementing a social media program while optimizing return on investment-an mba program case study. *American Journal of Business Education (Ajbe)*, *8*(1), 31–48.

Goh, A. T. (1995). Back-propagation neural networks for modeling complex systems. *Artificial intelligence in engineering*, *9*(3), 143–151.

GoogleDevelopers. (2022). *Embeddings: Translating to a lower-dimensional space*. https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space

Gopal, S., Yang, Y., Bai, B., & Niculescu-Mizil, A. (2012). Bayesian models for large-scale hierarchical classification. *Advances in Neural Information Processing Systems*, *25*.

Guan, Y., Myers, C. L., Hess, D. C., Barutcuoglu, Z., Caudy, A. A., & Troyanskaya, O. G. (2008). Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome biology*, *9*(1), 1–18.

Huang, C.-L., Shih, Y.-C., Lai, C.-M., Chung, V. Y. Y., Zhu, W.-B., Yeh, W.-C., & He, X. (2019). Optimization of a convolutional neural network using a hybrid algorithm. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Kemp, S. (2022). *Digital 2022: October global statshot report*. https://datareportal.com/reports/digital-2022-october-global-statshot

Khatoon, S., Alshamari, M. A., Asif, A., Hasan, M. M., Abdou, S., Elsayed, K. M., & Rashwan, M. (2021). Development of social media analytics system for emergency event detection and crisismanagement. *Computers, Materials and Continua*, 3079–3100.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiritchenko, S., Matwin, S., Famili, A. F., et al. (2005). Functional annotation of genes using hierarchical text categorization. *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*.

Lieb, R. (2009). *The truth about search engine optimization*. Que Publishing.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*.

Lipton, Z. C., Elkan, C., & Narayanaswamy, B. (2014). Thresholding classifiers to maximize f1 score. *arXiv preprint arXiv:1402.1892*.

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

Livshin, I. (2019). *Artificial neural networks with java*. Springer.

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Meng, Y., Shen, J., Zhang, C., & Han, J. (2019). Weakly-supervised hierarchical text classification. *Proceedings of the AAAI conference on artificial intelligence*, *33*(01), 6826–6833.

Miranda, F. M., Köehnecke, N., & Renard, B. Y. (2021). Hiclass: A python library for local hierarchical classification compatible with scikit-learn. *arXiv preprint arXiv:2112.06560*.

Opitz, J., & Burst, S. (2019). Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*.

Pereira, R. M., Costa, Y. M., & Silla, C. N. (2021). Handling imbalance in hierarchical classification problems using local classifiers approaches. *Data Mining and Knowledge Discovery*, *35*(4), 1564–1621.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, *48*(4), 1875–1897.

Sen, R. (2005). Optimal search engine marketing strategy. *International Journal of Electronic Commerce*, *10*(1), 9–25.

Silla, C. N., & Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*(1), 31–72.

Statcounter. (2022). *Search engine market share*. https://gs.statcounter.com/search-engine-market-share

Statista. (2022). *China's e-commerce growth trails the rest of the world*. https://www.statista.com/chart/22729/e-commerce-sales-growth-by-region/

Stats, I. L. (2022). *Internet live stats google*. https://www.internetlivestats.com/one-second/#google-band

Teahan, W. J. (2000). Text classification and segmentation using minimum cross-entropy. In *Content-based multimedia information access-volume 2* (pp. 943–961).

Tricahyadinata, I., & Za, S. Z. (2017). An analysis on the use of google adwords to increase e-commerce sales. *SZ Za and I. Tricahyadinata (2017) Int. J. Soc. Sc. Manage*, *4*, 60–67.

Valentini, G. (2009). True path rule hierarchical ensembles. *International Workshop on Multiple Classifier Systems*, 232–241.

Van Looy, A. (2022). Search engine optimization. In *Social media management* (pp. 125–146). Springer.

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

Wang, B., Hu, X., Li, P., & Philip, S. Y. (2021). Cognitive structure learning model for hierarchical multi-label text classification. *Knowledge-Based Systems*, *218*, 106876.

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*.

Xue, G.-R., Xing, D., Yang, Q., & Yu, Y. (2008). Deep classification in large-scale text hierarchies. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 619–626.

Yang, Z., Shi, Y., & Wang, B. (2015). Search engine marketing, financing ability and firm performance in e-commerce. *Procedia Computer Science*, *55*, 1106–1112.

Zhou, J., Ma, C., Long, D., Xu, G., Ding, N., Zhang, H., Xie, P., & Liu, G. (2020). Hierarchy-aware global model for hierarchical text classification. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1106–1117.

Zhu, H., & Lei, L. (2022). The research trends of text classification studies (2000–2020): A bibliometric analysis. *SAGE Open*, *12*(2), 21582440221089963.

A Appendix

Keyword	Segment 1	Predicted Segment 1	Segment 2	Predicted Segment 2
Bathroom mirror with shelf	Bathroom	Bathroom	Mirror	Mirror
Vanity mirror	Bathroom	Bathroom	Mirror	Mirror
Plastic shower caddy	Bathroom	Bathroom	Bathroom accessories	Bathroom accessories
Bathroom sets	Bathroom	Bathroom	Bathroom accessories	Bathroom accessories
Basket for toiletries	Bathroom	Bathroom	Storage container	Storage container
Toothbrush holder	Bathroom	Bathroom	Toothbrush holder	Toothbrush holder
Wooden spoon for baking	Kitchen utensils	Kitchen utensils	Spatulas	Spoons
Spatula	Kitchen utensils	Kitchen utensils	Spatulas	Spatulas
Chopping board	Kitchen utensils	Kitchen utensils	Cutting board	Cutting board
Ladle	Kitchen utensils	Kitchen utensils	Spoons	Spoons

Table 9: Results example dataset table 1

Segment 2	Search volume	Mediavalue	Predicted Search volume	Predicted Mediavalue
bathroom accessories	12800	€ 8,377.00	12800	€ 8,377.00
cutting board	33100	€ 13240.00	33100	€ 13240.00
mirror	34300	€ 16,544.00	34300	€ 16,544.00
spatulas	27150	€ 11,138.50	27100 ↓	€ 11,111.00 ↓
spoons	12100	€ 3,025.00	12150 ↑	€ 3,052.50 ↑
storage container	480	€ 146.40	480	€ 146.40
toothbrush holder	18100	€ 7,421.00	18100	€ 7,421.00

Table 10: Example total search volume and mediavalue for segment 2

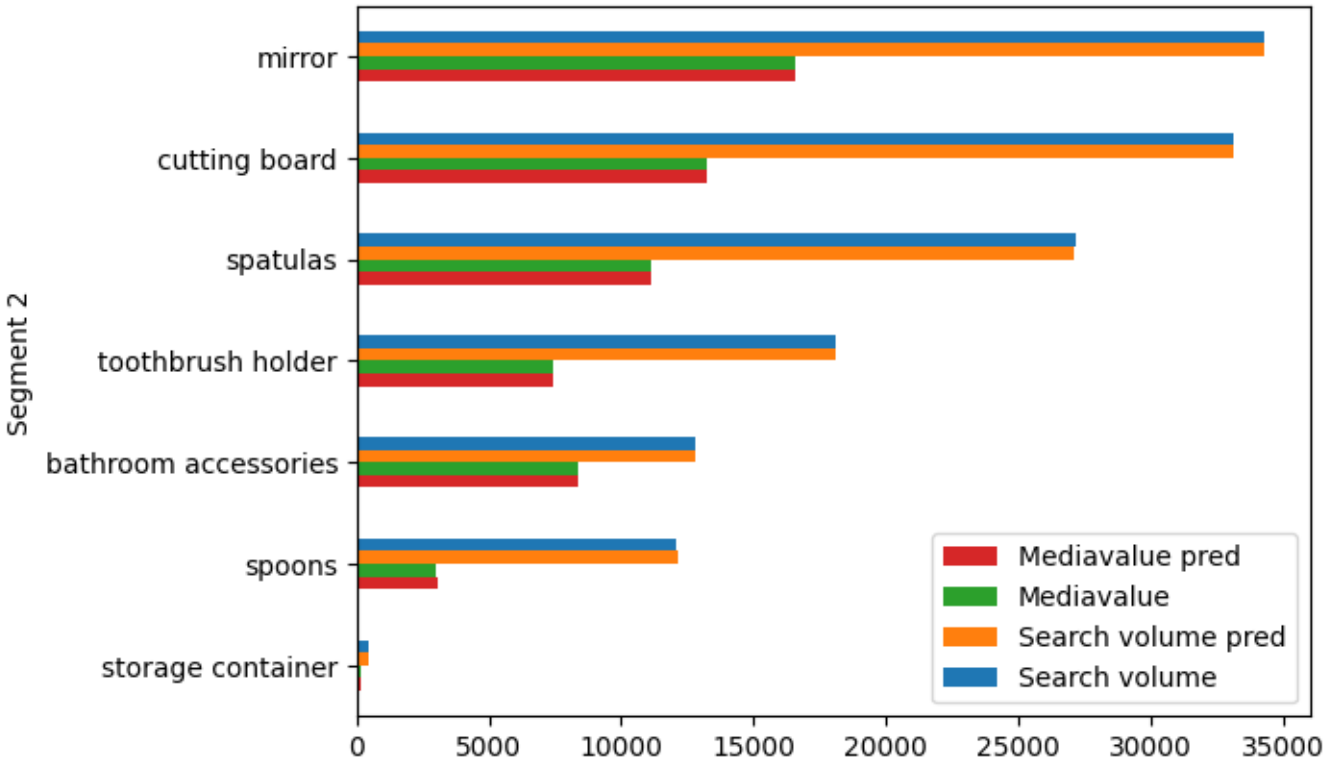

Figure 13: Results example dataset table 1

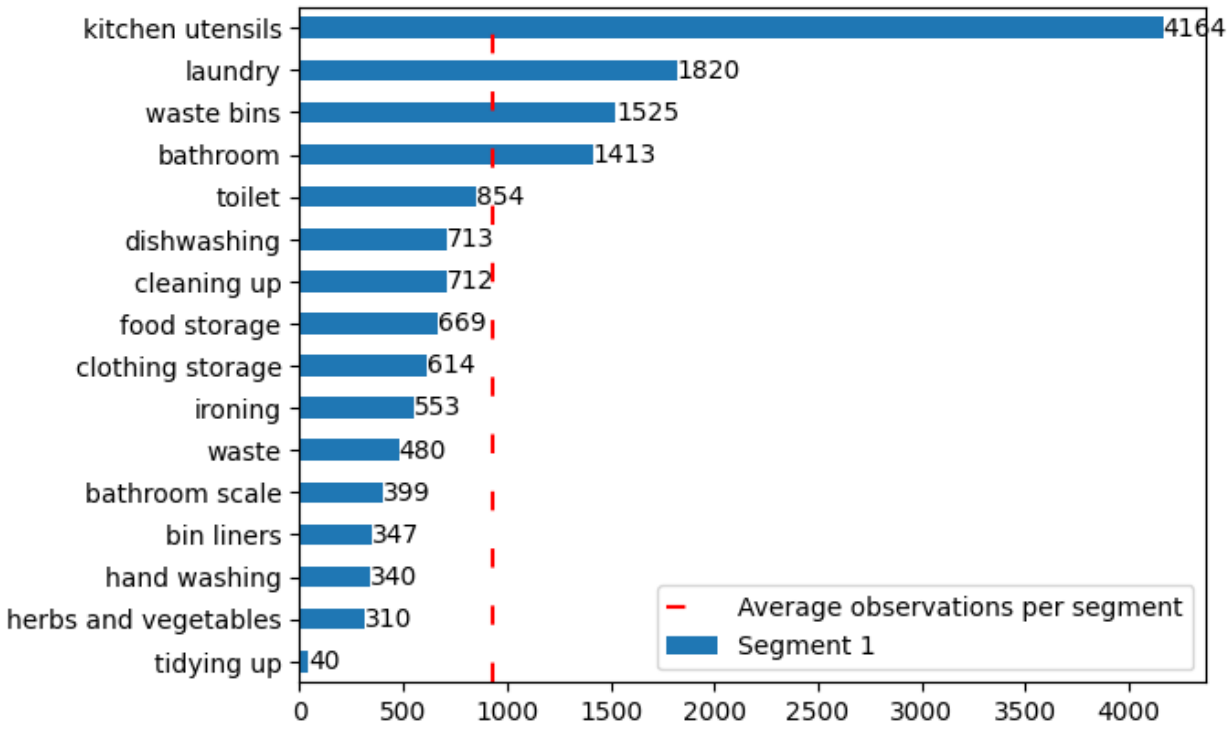

Figure 14: Amount of keywords classified per segment for segment 1 with the red line as average over segment 1

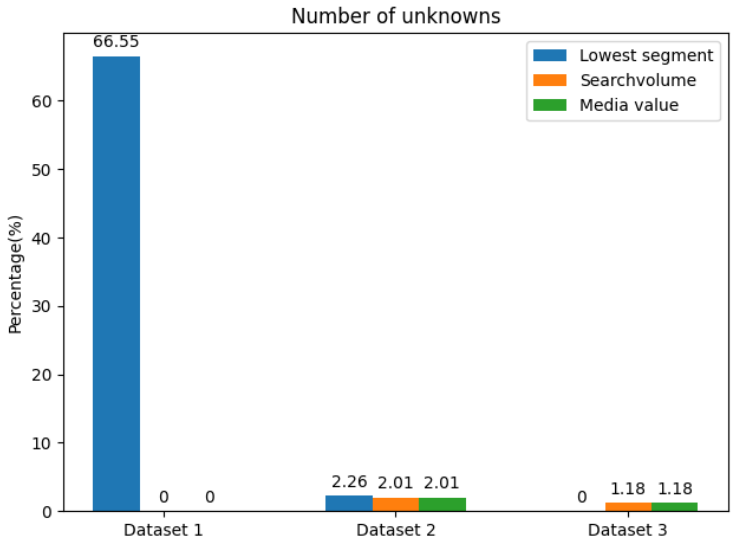

Figure 15: Percentage of unknowns

	value/type
Learning rate	$1e-3$
Learning rate type	Linear
Batch size	32
Epochs	75

Table 11: Settings multi-output neural network

	value/type
Learning rate	$1e-4$
Learning rate type	Linear
Batch size	32
Epochs	10

Table 12: Settings multi-output BERT-like neural network

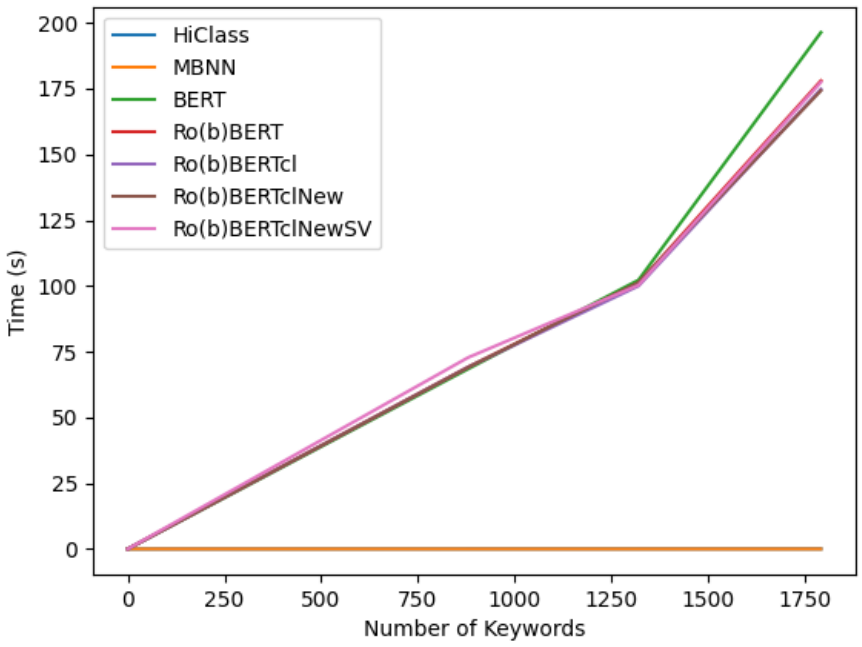

Figure 16: Test time needed for a certain number of keywords

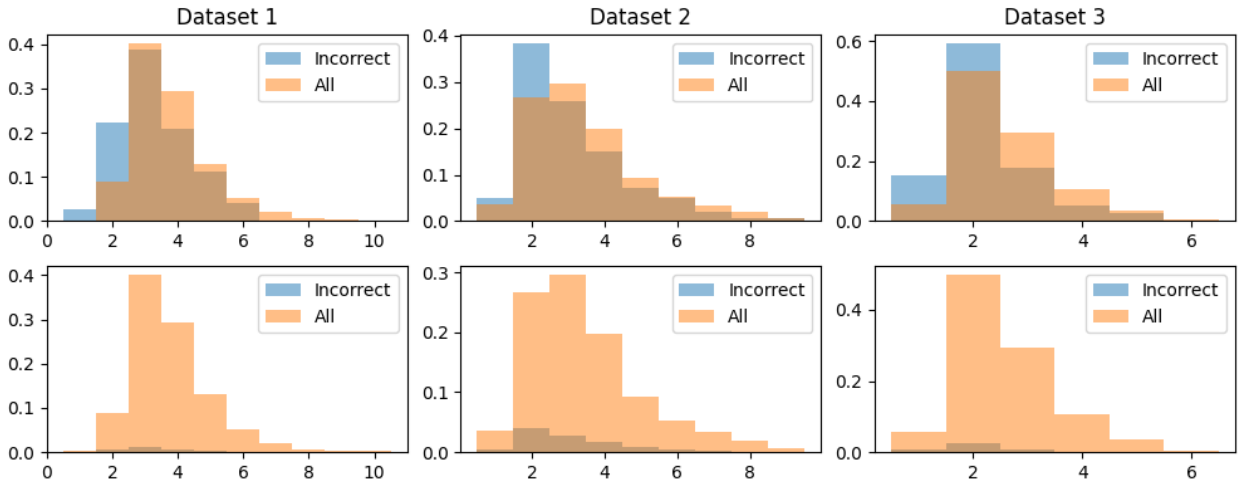

Figure 17: Top row: Distribution incorrectly classified keywords in combination with distribution of all keywords. Bottom row: incorrectly classified keywords displayed as part of the distribution of all keywords.