HIERARCHICAL ECONOMIC ACTIVITY CLASSIFICATION

ROBIN DE HEER

A Case Study with Text found on Web Pages

Bachelor of Science (B.Sc.)
Artificial Intelligence
Faculty of Social Sciences
Radboud University
SIDN Labs

2019-2020 – Master Thesis

Robin de Heer: *Hierarchical Economic Activity Classification,* A Case Study with Text found on Web Pages, © 2019-2020

SUPERVISORS:
Dr. F.A. Grootjen
Prof. Dr. Ir. A.P. de Vries
T.R. Wabeke, M.Sc.

LOCATION:
Radboud University, Nijmegen, the Netherlands
SIDN Labs, Arnhem, the Netherlands

TIME FRAME:
2019-2020

ABSTRACT

Gaining insight in what economic activities the businesses with a domain in the `.nl` zone participate in can help to increase online security. In this research project, the focus lies on classification of economic activity for business and e-commerce related domains in the `.nl` zone based on the text found on those domains.

Not all economic activities are represented as well as others in number of domains participating in that economic activity. A selection of which economic activities were considered was made based on the statistics of the Dutch National Statistics Office (CBS). Certain economic activities are better represented in number of businesses than others. The number of businesses operating in several economic activities is low enough to manually label the corresponding domains. Not considering those activities in the final classification model results in higher classification performance.

The main experiments in this research project consisted of three parts: Analyzing the influence of textual features extraction methods, the classification methods and the classification approach. The results indicate that, in order to achieve optimal classification performance, a term frequency inverse document frequency (tfidf) feature extraction method should be combined with a linear classifier trained using Stochastic Gradient Descent (SGD) of the Modified Huber (MH) loss function. These findings show that more complicated feature extraction methods or more complicated classifiers do not guarantee higher classification performance.

Hierarchical classification can be employed to perform classification on the second level the economic activity taxonomy. The final and most important experiment in this research project is designed to analyze if an advantage of hierarchical second level economic activity classification exists when compared to regular "flat" classification. The results show that when the hierarchical classification approach is used, a higher classification performance can be achieved.

From these results can be concluded that the use of more complicated methods for both feature extraction and classification does not guarantee increased classification performance. Classification performance can however be increased by exploiting an exisiting hierarchical structure in the data. Using the example of economic activity classification, we show that this performance increase generalizes from benchmark datasets to a non-benchmark problem.

The research project itself was deemed a success: Stichting Internet Domeinregistratie Nederland (SIDN), the administrator of the `.nl` zone, decided to take the second level economic activity classifier into

production. Every month, the hierarchical classifier is used to generate economic activity classifications for all business and e-commerce related domains in the `.nl` zone.

ACKNOWLEDGMENTS

I would like to thank both my internal and external supervisor for their valuable feedback and guidance throughout the project and express special thanks to the reviewers for their remarks. I would also like to thank my colleagues at SIDN for motivating comments, input and entertainment at foosball challenges.

This thesis would not look as good as it does without the Classic-Thesis template provided by André Miede for nothing but a postcard as thanks. I will send this postcard after graduation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

SBI Standaard Bedrijfs Indeling

NACE Statistical Classification of Economic Activities in the European Community

DDoS Distributed Denial of Service

KvK Dutch Chamber of Commerce

SIDN Stichting Internet Domeinregistratie Nederland

DMAP Domain name Ecosystem Mapper

tfidf term frequency inverse document frequency

SGD Stochastic Gradient Descent

MH Modified Huber

SVM Support Vector Machine

DT Decision Tree

RF Random Forest

LR Logistic Regression

NN Neural Network

CNN Convolutional Neural Network

RCNN Recurrent Convolutional Neural Network

RNN Recurrent Neural Network

PoS Part of Speech

NLTK Natural Language Toolkit

CBS	Dutch National Statistics Office
LSTM	Long Short Term Memory
tf	term frequency
idf	inverse document frequency
SSTb	Stanford Sentiment Treebank
STS	Stanford Twitter Sentiment
LCN	Local Classifier per Node
LCPN	Local Classifier per Parent Node
LCL	Local Classifier per Level
DAG	Directed Acyclic Graph
IPC	International Patent Classification

Part I

THE INTRODUCTION

1 INTRODUCTION

The Internet has grown rapidly since its introduction. The `.nl` zone alone consists of over six million different domains. SIDN, the administrator of the `.nl` zone, has classified almost one million of those domain as business or e-commerce related. The economic activity of the businesses corresponding to the domains can be determined in order to gain additional insight in the zone. These insights can be used for marketing purposes, but also and more importantly, to increase the security of the `.nl` zone.

Identification of domains linked to businesses with relatively high revenue compared to how vulnerable they are is valuable, because a domain can be attacked. Such an attack can be a Distributed Denial of Service (DDoS) attack. The goal of a DDoS attack is to disrupt or prevent any possible Internet traffic to a specified domain by sending an extremely high number of requests to this domain, effectively rendering it unable to respond to any other request [60]. These attacks can be used to extort a business or person, but also to cripple domains. For example, this effectively freezes the revenue generated by web shops. Estimating the economic damage of such an attack to a business or e-commerce domain is a difficult task, because estimating the missed revenue can depend on different variables. Among these variables is the economic activity a business participates in, since businesses can be more or less dependent on their online accessibility and the revenue of a business can vary based on the economic activity. The estimation is useful, because it not only yields information about which economic activity attracts DDoS attacks, but it can also be combined with other records. For example, in which economic activity businesses are using additional security measures to prevent or complicate DDoS attacks. This information can be used to advise or warn about the risks of not using these security measures. Some possible effects of DDoS attacks are:

- DDoS attacks can cost up to 20000 US dollars per hour when used against certain companies according to Kaspersky. Kaspersky is a multinational cybersecurity and anti-virus company [1].

1 https://usa.kaspersky.com/resource-center/preemptive-safety/how-does-ddos-attack-work

- The number of DDoS attacks will be 15.4 million US dollars by 2023 according to Cisco. Cisco is a multinational network technology company[2].

- Up to about 25% of all traffic in a country can be DDoS attack related[3].

Machine learning will be employed to classify the economic activity of a domain based on the text found on that domain. Machine learning is the task of learning a function which maps the input to an output. In this case, the input is the text found on a domain and the output is the corresponding economic activity. After learning such a function, it can be used to automatically obtain economic activity predictions for previously unseen domains. This is much faster than manual labelling, which is time-consuming and labor intensive, especially for larger collections [71]. More information about machine learning in general can be found in [70] and more information about text classification can be found in [1, 93]. Appendix A contains a few examples of text classification as well.

Various approaches to text classification exist: different feature extraction methods, classification schemes and approaches to the classification problem itself. All three of those aspects will be examined using the following research questions:

1. How important is the usage of different features with regard to classification performance in the economic activity classification problem?

2. How does the use of different classifiers affect classification performance?

3. Can classification performance be improved by using hierarchical classification?

Most more complicated concepts result in increased classification performance on benchmark datasets in the literature. See Section 3.1 for literature related to feature extraction and classifier selection and Section 4.1 for literature related to hierarchical classification. This practical example of economic activity classification based on text for domains and these research questions are intended to validate practical use of sophisticated implementations of these three concepts.

Machine learning is heavily dependent on features (e.g. [22]). Features are extracted from a data sample. Different methods exist to represent text in features. One approach is the usage of tfidf. In the case of economic activity classification, a term is a word found on

2 https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

3 https://cybersecurityventures.com/the-15-top-ddos-statistics-you-should-know-in-2020/

a domain and a document is all the text found on a domain. These vectors approximate word importance and contain a number for every word in the document collection. Another approach is to use word embeddings. Then, every word is represented as a vector in a semantic space. Words that have similar meaning, like "chair" and "table", have more similar vectors than words that do not have a similar meaning, like "chair" and "volleyball". The final approach examined in this research project is to use the importance measure of tfidf as a weight to nullify the embedding of less important words. Feature extraction for textual data on domains for economic activity classification is elaborated on in Chapter 3. To answer research question 1, tfidf vectors and (inverse document frequency (idf) weighted) word embeddings will be constructed. These features will be used as input to train a classifier per input. The performance of the resulting classifiers will be compared to answer the research question. The use of tfidf vectors resulted in the highest performance.

Different approaches to machine learning exist, resulting in numerous classification methods. Some linear classifiers are based on minimizing a loss function using SGD. Others are based on the creation of a hyperplane, which maximally separates one class from all other classes. This is the Support Vector Machine (SVM) classifier. Another type of classifier is the Decision Tree (DT) classifier. A DT classifier traverses a tree of decisions based on features to a leaf node. The leaf node will be the prediction for a set of features. In a Random Forest (RF) classifier, the use of multiple randomly initialized DT classifiers is an example of an ensemble method. The final prediction is a function of the predictions of all trees in the RF classifier. A Logistic Regression (LR) classifier estimates a probability per class, achieved by learning weights per feature. These weights are multiplied with the corresponding features and summed, resulting in the probability of the presence of a class.

The potentially most complicated classification method used in this research project is a Neural Network (NN). A NN consist of a number of layers of neurons, each connected to the previous layer. An input is propagated through the network and a prediction can be extracted from the final layer. Many different approaches to the creation of a NN exist, some more complicated than others. For example, a LR classifier can be implemented as a NN with three layers and a sigmoid activation function [34]. When more layers are added, like in [48], the NN becomes the most complicated classification method for this research project. Classifier selection with regard to the economic activity classification task based on textual data on domains can be found in Chapter 3. In order to answer the research question 2, the selection of the optimal performing classifier, several different classifiers will be used to perform economic activity prediction for text found on domains based on the features which resulted in the best performance in research question 1. The performance of these different classifiers will be compared to find the answer to this research question. The use of a linear classifier trained using SGD of the MH loss function resulted in the highest performance.

For the final and most important research question, research question 3, the best performing features and classifiers found in research question 1 and 2 will be used to create a hierarchical classification scheme. This classification scheme will be compared to one classifier, which is an identical classifier based on the same features. The difference or lack of difference between the performances of the classification scheme and the classifier will be inspected in order to determine whether or not hierarchical classification can increase economic activity classification performance. The use of a hierarchical classification system resulted in slightly higher performance when compared to a single classifier. This experiment can be found in Chapter 4.

The outline of this research project is illustrated in Figure 1.1. The introduction was this chapter. The next chapter, Chapter 2, contains all pilot experiments. These experiments include data exploration, a comparison of the distribution of the labeled data to the actual distribution as provided by the CBS and data augmentation methods. Preprocessing methods are described in this chapter as well. Chapter 3 and Chapter 4 contain the three main experiments of this research project and correspond to research questions 1, 2 and research question 3, respectively. These chapters both have the same structure: a literature background followed by methods, results and a short discussion of these results. Next, the deployment of the hierarchical classifier is discussed in Chapter 5. This research project concludes with a general conclusion and discussion in Chapter 6 and Chapter 7, respectively.

Figure 1.1: Structure of this research project.

2 PILOT EXPERIMENTS

Several pilot experiments were conducted in order to gain insight in the labeled data. These insights were required to be able to conduct experiments and draw valid conclusions to answer the research questions in Chapter 1. These experiments and their requirements are discussed in this chapter. First, a description of the dataset, a possible method to extend the dataset and the preprocessing methods are described. Then, the label distributions in the dataset are explored and compared to the actual distribution as provided by the CBS. Based on the insights acquired in those steps, a final data selection is made. The chapter concludes with a short reflection on the methods described.

2.1 DATASET AND PREPROCESSING

Due to the unique position of SIDN, the registry of the `.nl` zone, the database with domain registration information was available to obtain a list of all domains registered in the `.nl` zone. Domain name Ecosystem Mapper (DMAP) [87], a web scraper, was used to obtain various properties of the domains on this list, such as the text found on the domain, whether or not the domain is related to a business or e-commerce company, whether or not the domain was used to redirect to another domain and Dutch Chamber of Commerce (KvK) registration data. Domains which only redirected to another domain were not considered. As data, the text found on the domain is used.

Economic activity can be quantified as in the Standaard Bedrijfs Indeling (SBI) [13], which is derived from the Statistical Classification of Economic Activities in the European Community (NACE) [26]. Rather than simply listing all possible economic activities, the SBI is a taxonomy containing different level of specificity. The highest level of the SBI taxonomy are the sections, for example section C, industry. 21 sections exists and are identified by a capital character ranging from A to U. The second highest level of the SBI taxonomy are the divisions, for example division C.11, the production of drinks. Every division is in exactly one section. 86 different divisions exists and their identifiers range from 1 to 99.

As label corresponding to data, the top two levels, the section and the divisions of the SBI code are used, because a deliverable of this research project is a well functioning division classifier. Three methods were used to link SBI codes to business and e-commerce domains:

1. Scrape the domain of a business, identify a KvK identification number and extract the SBI code from the KvK database. Listing

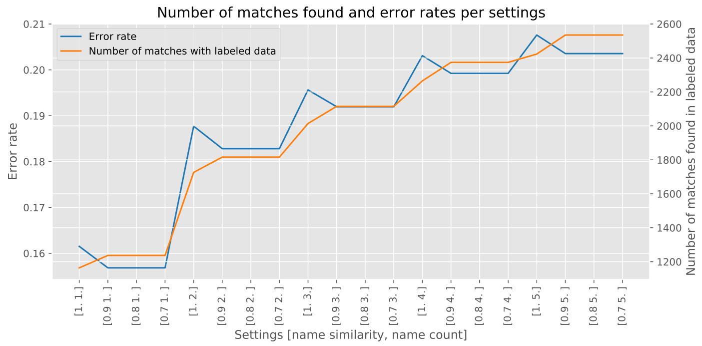

Figure 2.1: Error rates and number of matches with the already labeled data. On the x-axis, the extraction settings, name similarity and name count, are listed. On the left y-axis, the error rate (in blue) can be found. On the right y-axis, the number of matches (in orange) can be found.

an SBI code is mandatory to register a business, which ensures that SBI code will be present.

2. Scrape the KvK database for businesses which listed a domain.

3. Find matches based on name and address in the database with registration information and KvK database to obtain a combination of domain and corresponding SBI code.

Methods 1 and 2 were already in use by SIDN and the combination of domains and corresponding SBI code resulting from these methods is regarded as ground truth. Method 3 depends on two variables: the similarity between the name and address in the domain registration and KvK databases and the maximum number of domains per registrant. See Appendix B.3 for a visualization of how many domains are typically registered per registrant. The name similarity was calculated as the Ratcliff Obershelp distance [63], because this measure does not take substring location into account. Expectedly, the number of matches is higher when the name similarity threshold is lower and the number of domains per registrant is higher. Matches were extracted using various settings and an error rate was computed based on the intersection of the set of new matches and the ground truth. Figure 2.1 shows that the error rate as well as the total number of new matches goes up when the name similarity was lower and the maximum number of domains per registrant is higher.

This figure shows that the minimum error rate is just below 16% at a name similarity of 0.9 and a maximum number of domains per registrant of 1. Using these settings to minimize the noise introduced in by the new data, 18930 additional domains were labeled. However, after several methods of testing, the newly extracted data proved to contain too much noise. This was tested in six different ways, shown in Table 2.1. All classifiers were tfidf vector based linear classifiers

	Accuracy	F_1 score
Match data only	0.4171	0.4173
Pre-train using match data	0.7039	0.7053
Not using match data at all	0.7055	0.7066
Merge match data with ground truth	0.6678	0.6703
Train with match data, test with ground truth	0.4810	0.4853
Train with ground truth, test with match data	0.3678	0.3779

Table 2.1: Performance of several classifiers to evaluate the quality of the newly extracted match data.

trained using SGD of the MH loss function (further elaborated on in Chapter 3). The text was first preprocessed as described later in this section. Confusion matrices and classification reports of the listed classifiers can be found in Appendix B.3.

In this research project, only the 104115 domains related to business or e-commerce which could be labeled using methods 1 and 2 were used, because the quality of the data resulting from method 3 proved to contain too much noise.

Data preprocessing is an important if not necessary step for any automated classification system [77, 86]. The first step is to convert all words to lowercase. The reason this step was taken, is because the meaning of most words does not change when capitalized. Possible exceptions include a name, for example John Fisher. The last name implies a connection to the fishing profession when decapitalized. The assumption is that such examples are rare and can therefore be ignored.

The second step eliminates irrelevant words such as stopwords, because those words do not carry any substantive meaning and can only imply false relationships between a domain and its SBI. Digits are removed as well, as they are assumed to not add to the meaning of a text. A full stopword list can be found in Appendix B.1 and was obtained from the Natural Language Toolkit (NLTK) [9]. Both Dutch and English stopwords were included. Most of the `.nl` zone is assumed to be in Dutch and the English stopwords were included to account for web pages in (partly) English. Figure 2.2 further illustrates the relevance of removing stopwords. The most occurring word without stopword removal is "de", which is Dutch for "the". This is a meaningless word. The most occurring word after stopword removal is "contact", which translates to "contact" is not meaningless. 83251 different words were eliminated by stemming and stopword removal. A peculiar detail is that both distributions shown in this plot are not Zipfian, contrary to the expectation that any large corpus conforms to Zipf's law [94]. A possible explanation for this phenomenon is that the dataset is very

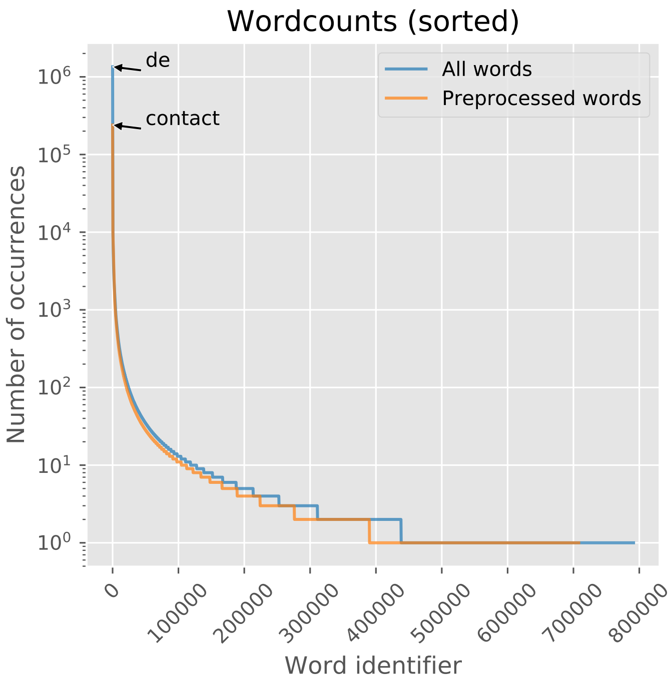

Figure 2.2: Number of occurrences per word. The y-axis indicates the number of occurrences. The x-axis is the word identifier. In blue, the initial word distribution is used. In orange, the preprocessed word distribution is used. This graph shows that the most occurring words are all stopwords.

diverse. It may not be large enough to conform to Zipf's law. Zipf's law states that the word frequencies in natural language corpora are inversely proportional to their rank, resulting in a straight line in the plot.

The third step of the preprocessing process is to stem all words. This step ensures not only that one single word representation is used for singular and plural forms of a word, but also for multiple verb tenses. Any word, regardless of the language the word is in, was stemmed using the Dutch Snowball stemmer implementation in the NLTK [9]. Stemming was chosen over lemmatization, because stemming is more efficient. It does not use a corpus and Part of Speech (PoS) tagging, requiring less computational power and is therefore faster.

A disadvantage could be that the roots generated by stemming are not necessarily actual words. The assumption is that this does not influence the ability of a classifier to differentiate between classes.

2.2 DATA EXPLORATION

Data exploration is necessary in order to get familiarized with the data. Several potential problems can be addressed early on if one is

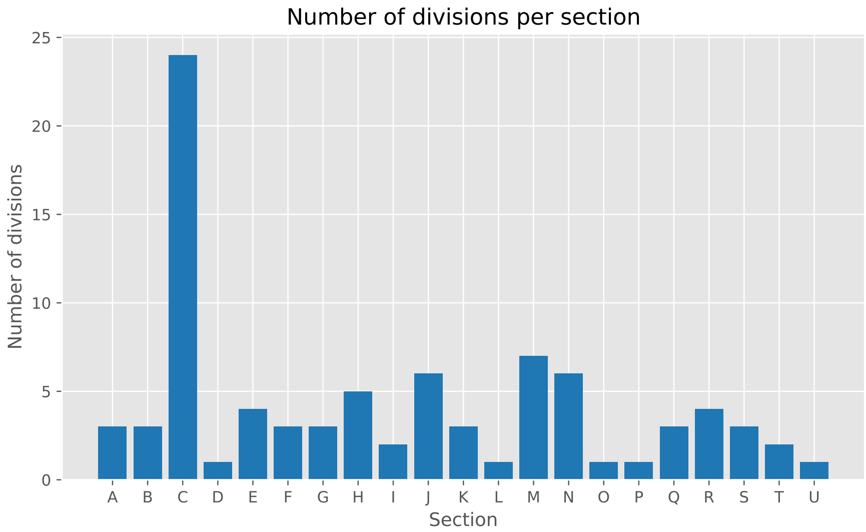

Figure 2.3: The number of divisions per section as specified by the CBS. On the x-axis, the sections are listed. On the y-axis, the number of divisions of the corresponding section is shown.

familiar with the data. In this section, the following potential problems are addressed:

- How many divisions exist per section of the SBI? And how many labeled data samples exist per section and per division? Are the section and division distributions uniform? Or are certain sections or divisions over- or underrepresented? This is known as class imbalance. Early identification class imbalance is useful, because measures such as class weight introduction or balanced batch training during training phase can prevent problems in the testing phase.

- Are labeled data distributions for sections and divisions similar to the actual distribution as provided by the CBS? This is required to ensure a well functioning classifier can be deployed to generate a classification for all business and e-commerce related domains in the `.nl` zone.

- How many words are typically found on a domain? Is that number high enough to be able to accurately classify the section and division of that domain?

Not every section contains the same number of divisions. For example, section "C" contains 24 divisions, whereas for example section "D" contains only one division. A full visualization is found in Figure 2.3.

The labeled data distribution for the sections is shown in Figure 2.4. Over fifty percent of the labeled data is contained in 4 sections, namely sections "M", "G", "Q" and "S", while 21 sections exist. Another

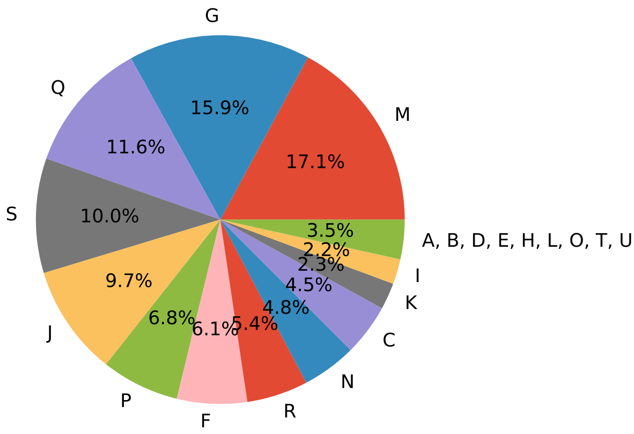

Figure 2.4: Illustration of the labeled data distribution over the sections.

peculiar observation is that, while section "C" contains 24 divisions, it contains only 4.5% of the labeled data. Therefore, section "C" is an example of an underrepresented class in the labeled data.

Potential problems caused by class imbalance are also present on division level. The labeled data is not evenly distributed over the divisions of a section and therefore not evenly distributed over all divisions at all. Several examples are shown in Figure 2.5. For section "A", over 90 percent of the labeled data is in division "1". A more uniform labeled data distribution is found for section "N". Still, division "77" contains almost 6 times as many labeled data samples as divisions "80".

In order to quantify the statistic similarity between the labeled data distribution and the data distribution as provided by the CBS, the Kolmogorov-Smirnov test [46] is applied. This test measures the distance between two distributions. The null hypothesis is that both distributions are the same. If the K-S statistic is small or the p-value is high, the null hypothesis cannot be rejected. For the sections, this test resulted in a statistic of 0.19 and a p-value of 0.80 and for divisions, the test resulted in an statistic of 0.07 and a p-value of 0.98. Therefore, the null hypothesis on both section and division level cannot be rejected and can be concluded that the distribution of the labeled data is drawn from the same distribution as the data distribution as provided by the CBS. A visualization of the comparison between the labeled data distribution and the actual data distribution on both section and division level can be found in Appendix B.2.

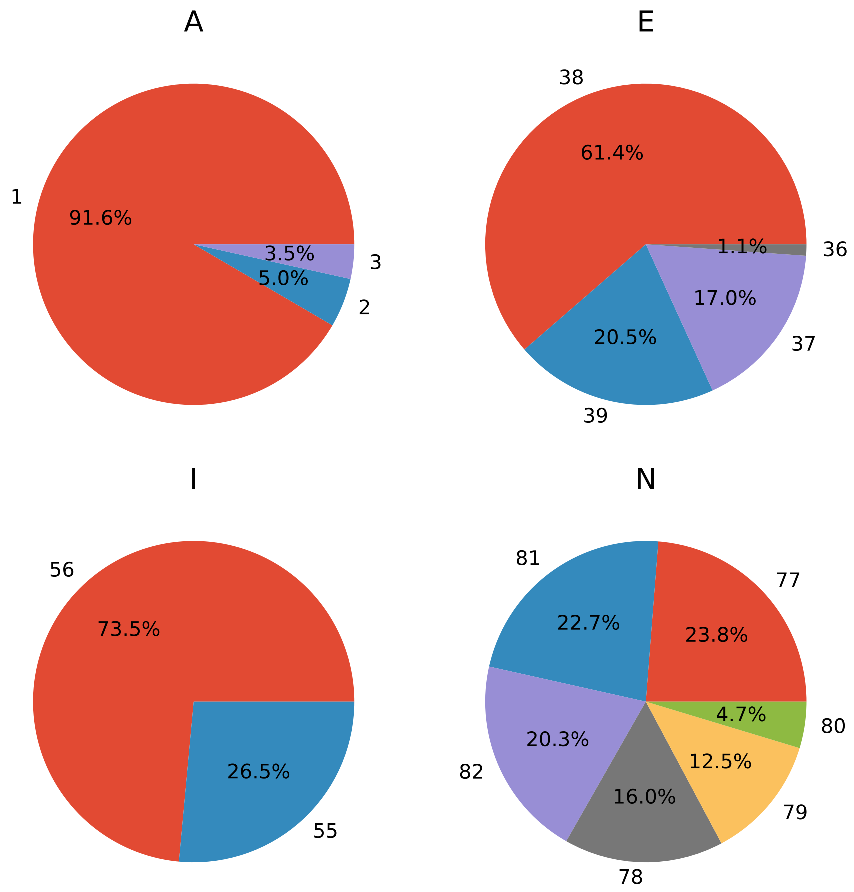

Figure 2.5: Illustration of the labeled data distribution for 4 sections. These graphs show that the data distribution in a single division can be far from uniform.

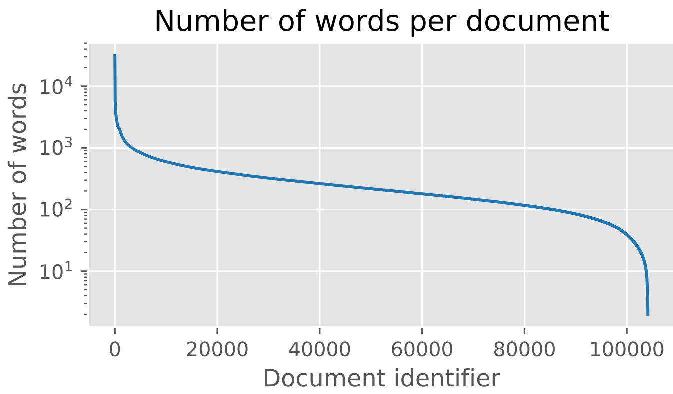

Figure 2.6: The number of words per document. On the x-axis, the documents are listed with an integer identifier. On the y-axis, the number of words is indicated.

The text on these domains naturally varies in length. Some domains have more text than others. More text can introduce more noise to a classifier, but it can also help the classifier to make a decision on the prediction. In order to find out how many words are typically on a domain, a visualization was made in Figure 2.6. Inspection of this plot reveals that most domains contain enough text to perform economic activity classification.

2.3 DATA SELECTION

Data selection is an important aspect of any classification problem. Determining which classes will be used and therefore can be predicted and which classes to ignore can be seen as an optimization problem: disregarding classes can improve classification performance, but trims the number of classes a classifier can predict, therefore reducing the amount of information a classifier can yield. In this section, the data selection is made and a motivation for this selection is elaborated on.

Two methods to find thresholds for removing divisions from the data can be identified:

1. The labeled data contains only a few samples for a division (class).
2. Only a few companies operating in a division according to the CBS. Therefore, in the best case, only a few domains which can be labeled with that division exist.

From these two methods, two perspectives and motivations emerge:

1. A real-world perspective: if only a handful of these web pages for companies in a certain division exist, the use of economic activity classification is relatively low for these web pages. One

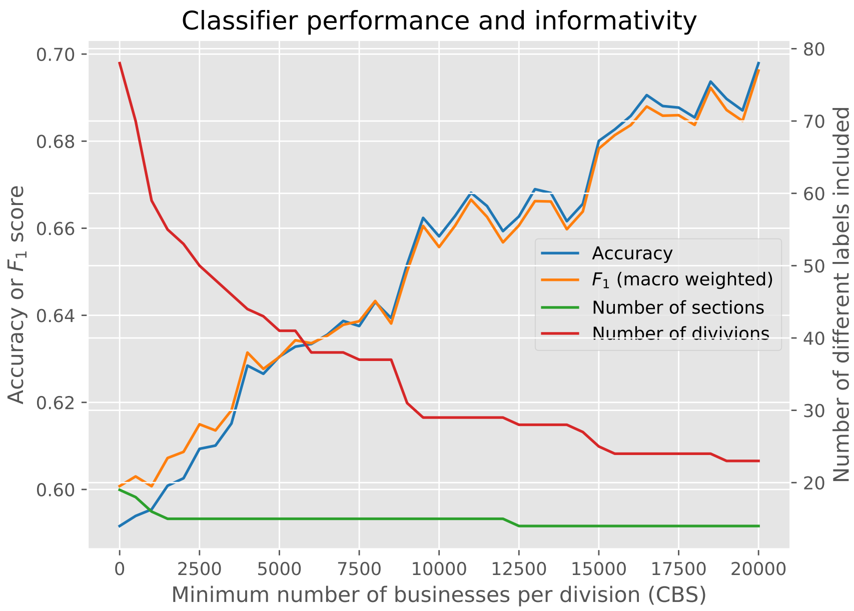

Figure 2.7: Illustration of how removing less well-represented classes influences the classification performance. On the x-axis, the number of actual businesses in a division is found. On the left y-axis, the performance in accuracy and macro weighted F_1 score corresponding to the blue and orange lines is found. On the right y-axis, the number of different sections or divisions, the red and green lines, taken into account by the classifier is found.

might even consider manually labeling those web pages and using the classification system for the remainder of the web pages. Also, hand-labeling up to 4000 domains per division of 45 divisions can be done within reasonable time according to SIDN.

2. A data-driven perspective: removing underrepresented classes will result in a more robust classification system.

The performance boost by removing classes is further illustrated in Figure 2.7. Removing divisions (in red) leads to a higher classification performance in terms of both accuracy and F_1 score (in blue and orange, respectively) of a tfidf based linear classifier trained using SGD of the MH loss function (further elaborated on in Chapter 3). The final threshold was set to at least 4000 businesses per division, resulting in at least 65 samples per division. Refer to Appendix B.4 for a complete overview of how well every division was represented, which divisions were removed and which sections were removed because they contained no divisions anymore.

The remaining data was split into two sets: training (including validation) and testing sets. 90% of the data was training and validation data and 10% of the data was testing data. These splits were stratified on the division labels to ensure a proportionate number of samples per class per set.

2.4 DISCUSSION

In this chapter four main subjects were discussed: the dataset itself and a possible extension are described, the preprocessing methods are described and motivated, the data is explored and the insights acquired are used to create a final data selection. These subjects were discussed in order to obtain valuable insight in the dataset, which was a requirement for the experiments concerning the research question.

The dataset consists of the business or e-commerce related domains without redirects and a corresponding SBI code. This SBI code was used to obtain the section and the division of that domain. This dataset may contain noise, because not every entrepreneur knows the actual SBI code when registering their business and may just check some box to get it over with or consider participating in another economic activity. No measures were taken to find out whether or not this happened. The assumption is that such examples are rare.

Not all of the text found on domains was written in Dutch. These texts were not filtered out, but English stopwords were removed as well to account for them in a limited way. Future work could include a translation step to account for non-Dutch domains in a more optimal way.

The labeled data distributions were proven to be statistically similar to the actual distributions, which was required to be able to effectively use any resulting classifier for real-life applications. Several sections or divisions were underrepresented in both the labeled data and actual businesses. To overcome this problem, a selection of the data was made. This selection was based on the number of real businesses operating in a division. Domains belonging to businesses operating in sections or divisions which did not make the selection are relatively scarce and could be manually labeled if required. The remaining data was split into training and testing sets. These training and testing sets are used in the next chapters to evaluate the use of different features and classifiers in Chapter 3 and classification approaches in Chapter 4.

Part II

THE EXPERIMENTS

3 TRADITIONAL ECONOMIC ACTIVITY CLASSIFICATION

Traditional text classification methods usually consist of two main components: the feature extraction method and the classifier. This chapter also consists of those two components: firstly, different feature extraction methods are examined in order to select the feature extraction method resulting in the highest performance for economic activity classification. Secondly, various classifiers are employed in order to select the highest performing classifier for this problem. The combinations of feature extraction methods and classifiers which are examined are marked with an "X" in Table 3.1. This table also shows that not all combinations are tested. The feature extraction method is selected using a linear classifier trained using SGD of the MH loss function. All other classifiers with exception of the NN classifiers are tested with the resulting feature extraction method. The NN classifiers were tested using word embeddings, because this is what NN classifiers with Long Short Term Memory (LSTM) [37] neurons are optimized for.

The reason that the use of different features is explored is that features of higher quality can result in higher classification performance [22]. Three feature extraction methods are considered: the importance measure based term tfidf vectors and the vectorspace based word embeddings. The final feature set is engineered by using the importance measure of tfidf, idf, as a weight for the word embeddings to see if both advantages can be combined in a single feature set. Next, in order to obtain a full overview of classifier performance per classification method, different tfidf vector based classifiers are trained and evaluated. Furthermore, a RCNN [48] and a RNN are trained using word embeddings.

The hypotheses are that:

a) The use of idf weighted word embeddings will result in the highest classification performance, because this feature extraction method combines the importance measure of tfidf vectors and the semantic information of word embeddings.

b) The RCNN classifier will outperform all other classifiers. This classifier achieved state of the art performance in various text classification tasks and is the most complex and innovative of all tested classifiers.

By testing these hypotheses, the use of more complicated features and classifiers can be validated for a non-benchmark problem. The use of both more complicated features and classifiers shows promising

	tfidf vectors	Word embeddings	idf weighted word embeddings
SGD (MH)	X	X	X
SGD (hinge)	X		
LR	X		
SGD (log)	X		
DT	X		
RF	X		
RNN		X	
RCNN		X	

Table 3.1: Overview of which combinations will be tested. Per classifier is indicated with an "X" if the feature extraction type is tested. Linear classifiers trained using SGD of a loss function are indicated as "SGD (loss function)".

results in the literature. For example, the RCNN [48] broke several records of benchmark tasks.

This chapter is built as follows: a literature background on both feature extraction and classification schemes followed by experiment methods, results and wrapped up by examining the implications of the results and answering research questions 1 and 2 in the discussion.

3.1 BACKGROUND

Various approaches for feature extraction from textual data exist. In this research project, the importance measure per word based tfidf vectors and the vectorspace based word embeddings are examined more closely. Selecting a feature extraction method is important, because it may influence final classification performance greatly, just like the type of classifier used. A literature background for both feature extraction methods and classifiers will be given in this section.

3.1.1 *Feature Extraction Methods*

Intuitively, word statistics can yield information about text subjects [54]. Multiple occurrences of the words 'processor', 'memory' and 'hard disk' can for example indicate that a document is about computers. Luhn [54] was mainly thinking about term frequency (tf) per document, but did not consider the possibility of words occurring in many documents. A word's discriminative power diminishes when it occurs in more documents. When combined with term specificity [76] or idf, which is a measure of term importance across the entire document collection, a tfidf vector per document, which is a robust metric of term importance per term can be constructed.

A problem with tfidf vectors is that they do not account for semantic similarity between words. For example, synonyms may have completely different idf values. To address this problem, Mikolov et al. [59] proposed to use word embeddings. Those embeddings are constructed by training a neural network with one hidden layer to predict nearby words based on the word used as input. Every word is represented as a one-hot vector. After training, the embedding of a word can be extracted by executing one forward pass through the trained network with a the one-hot vector of the desired word as input and extracting the weights of the hidden layer. These embeddings proved quite powerful and they are easily trained and fine-tuned for specific collections. As such, Word2Vec [59] was used as input for Convolutional Neural Network (CNN)s to perform various sentence classification tasks [42]. The CNN performed remarkably well compared to other approaches which did not use word embeddings. This work makes it clear that word embeddings can make a CNN perform exceptionally well. For example, Lai et al. [48] employs word embeddings in a RCNN to perform various text classification tasks and achieves (better than) state-of-the-art results. While the word embeddings themselves are static, the use of a convolution layer ensures a local contextual representation.

A combination of tfidf vectors and word embeddings is possible as well. Then, the idf value of a word is used as the weight for the word embedding. The idea here is to nullify the non-important embeddings in order to minimize the influence of the non-important words and to put an emphasis on the influence of the important words. A detailed description of how to do this can be found in i.e. [4, 19].

3.1.2 *Classifiers*

A commonly employed classifier type for text classification is the SVM, because of its simplicity and high performance in general on text classification tasks [39, 40] compared to Naive Bayes, Rochhio, C4.5 and k-Nearest-Neighbours classifiers. An SVM is normally a binary classifier, but by creating a one-vs-all classification scheme [44], a problem of n classes can be approached by using $n_{class} * (n_{class} - 1)/2$ SVM classifiers in which each classifier distinguishes between two classes. An SVM classifier finds the optimal hyperplane in the input space to separate one class from other classes. This plane does not have to be linear in nature, but can also be implemented by some other function. SVMs perform well on text classification problems, because they can handle high dimensional data well and reducing the dimensionality disregards information. It is shown that a classifier performs best when using all features [40]. Even removing the features with the least (binary) information gain reduces the classification performance. The authors also show that even when only using the worst features, the classifier performs better than chance level. This indicates that even those features hold somewhat important information. This shows that all features are important and that a classifier should be able to deal with a large number of features. Another reason SVMs handle textual data well, is because they can handle sparse data quite well and textual data is sparse [43]. The final reason SVMs perform well at text classification problems is because those problems are usually linearly separable and the task of an SVM is to determine a decision boundary between a specific class and the remainder of the classes. All in all, the SVM classifier is robust and well performing at text classification problems.

Text classification can also be viewed as a rule-based problem. Therefore, a DT classifier is a valid option as well. The occurrence of specific words can indicate a topic. Previous research successfully applied DT classifiers to classify news articles [3]. However, DTs are prone to overfitting. A solution to this phenomenon is to generate multiple DTs and determine the final prediction as a function of the individual predictions [2]. This set of DT classifiers is known as a RF classifier [12]. This approach is known as an ensemble approach, in which multiple so-called weak learners are capable of achieving final predictions which are more accurate than the predictions of each individual weak learner.

Another classification method is LR. The name falsely implies usage for regression tasks. The classifier predicts a probability per class. The origins and developments of LR can be found in [18]. LR is used for various text classification tasks [30, 51]. Genkin, Lewis, and Madigan [30] used a Laplacian prior as an addition to the LR algorithm to avoid overfitting. While their algorithm does not greatly outperform an SVM, it has the advantage of handling sparse data very well. Therefore, it can to great effect be used when the data is sparse (i.e. text). In the field of information retrieval, problems of unlabeled data arise frequently. To circumvent this problem, Lee and Liu [51] proposed to label the unlabeled data as the negative class. While this approach introduces noise in the data, a LR model with sufficient regularization is capable of achieving high performance when employed for text classification. Their approach is proven successful after testing.

A linear classifier can be implemented using the SGD optimization method. This classifier uses a loss function, which can be any function of the predicted and true labels, and attempts to find the minimum of that loss function using SGD. This minimum is usually achieved when the predicted and true labels are identical. This loss function can be customized at will to optimize the classifier for any specific problem. The algorithm itself traces back to the Robbins–Monro algorithm [65]. SGD is now a well established and important tool in machine learning [10]. The classifier uses an approximation of the actual gradient in the data. Therefore, the computational power required to compute the gradient at each iteration is reduced and every iteration is completed faster at the cost of a lower convergence rate [11].

A trend in machine learning is to use (deep) NNs for classification tasks. Such a NN consists of a varying number of layers of neurons which are connected to neurons in the previous and next layer or even to themselves in a RNN. A computational model for NNs was first proposed by McCulloch and Pitts [57]. Later, 'calculators' as the authors called it [28] were created which operated according to the Hebbian learning rule [35]. All of those networks however, were static in nature and could not yet be adapted to learn a problem. Later, the first perceptron was created [67]. A perceptron is a linear classifier for binary problems which estimates its output depending on input variables. In the learning algorithm, the decision boundary, a straight line, will be learned to optimally separate the classes. While this perceptron did not have any hidden layers, extensions for those hidden layers were possible. Later, the backpropagation algorithm was proposed by Rumelhart, Hinton, and Williams, which allowed training NN with hidden layers to be trained for specific purposes. The backpropagation algorithm adjusts weights between neuron by a small value in order to perform better. Upon convergence, the training is complete and the network is ready for testing.

NNs have also been employed to tackle text classification problems. Previous work describes the use of Word2Vec [59] as input for CNN [42] to perform various sentence classification tasks. Word2Vec is a method to convert words into vectors and is widely used as an alternative to for example tfidf vectors to provide features. Convolutional layers are layers in a NN which are connected to specific neurons from the previous layer, in order to preserve spatial information. In the case of text classification, the spatial information consists of the context. The CNN performed remarkably well compared to other approaches which did not use word embeddings. This work makes it clear that word embeddings can make a CNN perform exceptionally well. Another approach using NNs is based on character level embeddings [23]. This approach resulted in the state-of-the-art performance in sentiment analysis on both the Stanford Sentiment Treebank (SSTb) and the Stanford Twitter Sentiment (STS) databases. The SSTb dataset contains sentences labeled by sentiment and the STS dataset contains Twitter messages labeled by sentiment.

Word embeddings were also used in a RCNN to perform various text classification tasks and achieved (better than) state-of-the-art results [48]. The word embeddings are static, but the use of a convolution layer ensures a local contextual representation. The recurrent aspect of the network allows the use of multidimensional word embeddings. Every document consists of multiple words and every word consist of multiple features. Further importance of local contextual representation is shown more recently, in [80]. The authors show that a CNN (context is considered) outperforms the RNN (context is not considered) in document classification.

3.2 methods

3.2.1 *Feature Evaluation Experiment*

The tfidf vectors were constructed as in Equation 3.1. In this equation, t is the term, the word, d is the document, the text, $tf(t, d)$ is the number of occurrences of a term t in a document d and $idf(t)$ is the inverse document frequency of term t. $idf(t)$ is calculated as in Equation 3.2. In this equation, $df(t)$ is the number of documents a term t occurs in. 1 was added in the denominator for smoothing purposes and to avoid zero-divisions. The resulting tfidf vectors were subsequently normalized by the Euclidean norm as in Equation 3.3. Only the 50000 most occurring words were considered. The same words are used in both feature extraction methods.

$$tfidf(t, d) = tf(t, d) \cdot idf(t) \tag{3.1}$$

$$idf(t) = \log(\frac{1+n}{1+df(t)}) \tag{3.2}$$

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}} \tag{3.3}$$

The second and third feature extraction methods depend on word embeddings [59]. These embeddings are obtained by training a neural network with only one hidden layer to predict the next word. After training, word embeddings can be obtained by using the target word as input and extracting the weights of the hidden layer. An advantage of this method is that semantically related words will activate the network similarly and are therefore closer together in the vectorspace. The cosine distance can be used to measure similarity between the word embeddings.

The word embeddings were obtained using Word2Vec, included in the Gensim [64] library for the Python programming language. Most parameters for training this were left on default. 4 parameters were changed: the number of iterations over the whole dataset was set to 10. Using a high value may result in overfitting. For word embeddings, overfitting occurs when non-existent relationships between words are found and expressed in the embeddings. The number of hidden units of the network and therefore also the length of the constructed embeddings was set to 300. Several advantages and disadvantages exist for a bigger size: the embeddings are better able to express the meaning of the word, but it takes longer to obtain the embedding itself. When a bigger value is used, the risk of overfitting also becomes bigger. However, taking a small value may result meaningless embeddings.

For fitting the model, the text on the entire business/e-commerce fraction of the `.nl` zone was used, including the text from domains without an economic activity label, because more data ensures a more robust final model. Training on all domains was possible, because Word2Vec is an unsupervised method, meaning that it does not require labels. Limiting the number of words included in the model was required due to memory issues. Therefore, and to avoid experimental design flaws, the words included in the Word2Vec model were the same 50000 words as in the tfidf feature extractor.

The features resulting from the Word2vec model were embeddings for every word in a document. Not all documents were of the same length and moreover, the classifiers provided by SciKit Learn [61] cannot handle multiple dimensions per data sample. These limitations will be addressed by using the more complicated RCNN [48] classifier. Therefore, two methods were utilized to transform the extracted features to one one-dimensional feature vector:

1. Average all embeddings of a document per dimension to obtain one feature vector containing 300 elements per document. This method effectively reduces all word embeddings of a document to a single document embedding of the same size.

2. Multiply the word embedding with the idf value of the word. Then average all word embeddings of a document per dimension to obtain one feature vector per document as in the first method.

In order to be able to answer research question 1, what features to use to achieve the highest performance, a SGD classifier with the MH loss function was trained for all features to predict SBI sections. The research question can then be answered by comparing the performances of these classifiers.

3.2.2 *Classifier Evaluation Experiment*

Various classification methods were explored in order to find the most suitable one for economic activity classification. The tested classification methods include a linear classifier trained with SGD of the hinge, log and MH loss functions, a LR classifier, a DT classifier and a RF classifier with 200 estimators and a maximum depth of 50.

SGD classifiers based on the hinge and log loss functions are essentially SVMs and LRs, because SVMs use the hinge loss function and LRs use a log loss function. SciKit Learn [61] implementations were used for all classifiers.

Several different methods of overcoming class imbalance are possible. Perhaps the easiest method is to introduce class weights in the training phase. This method was chosen for its simplicity. Alternatives include balanced batch training or generating addition data for the minority classes with for example SMOTE [17]. Class weights counter data imbalance effects from underrepresented classes. These weights were calculated as in Equation 3.4 [61]. In this equation, W_c is the weight of a class c, y consists of all labels and therefore $|y|$ is the length of the dataset, $|\text{set}(y)|$ is the number of different labels existing in y and $|c|$ is the size of class c.

$$W_c = \frac{|y|}{(|\text{set}(y)| \cdot |c|)} \tag{3.4}$$

MH loss is calculated as in Equation 3.5. In this equation, $f(x)$ is a classifier score and y is a binary class label $y \in \{+1, -1\}$. $max(0, 1 - yf(x))$ is also known as the hinge loss, which is used by SVMs. In the MH loss, the smoothed quadratic hinge loss is used. A one-versus-all scheme was employed, because of the binary nature of both the linear classifier trained with SGD of the MH loss. A binary classifier is trained to distinguish one class from all other classes for every class in the training phase. Confidence scores are calculated and the class belonging to the classifier with the highest score is adopted as final prediction.

$$L(y, f(x)) = \begin{cases} max(0, 1 - yf(x))^2, & \text{if } yf(x) \geqslant= -1 \\ -4yf(x), & \text{otherwise.} \end{cases} \tag{3.5}$$

A RCNN classifier was employed in addition to the baseline set by the linear classifiers trained with SGD of the hinge, log and MH loss functions, LR, DT and RF classifiers. This classifier trained for 10 epochs with the following parameters: a batch size of 128, the RMSprop optimizer and the categorical cross-entropy loss function. The architecture of the network was adapted from a record-breaking network [48] and uses LSTM units [37] to account for the dimensionality of the input. The full configuration of the RCNN can be found in Appendix D.3.

In order to generate the left and right contexts of a document, all words in that document were shifted one index to the left or right. The first and last words were set to the last and first index. Training the RCNN was a computationally heavy task, because the input size is three times as much as its non-convolutional counterpart due to the fact that left and right contexts were also used as input. Therefore, another network similar to the RCNN was trained to determine the value of the convolutional and pooling layers. This network had the exact same layout as the RCNN without those layers. Both networks were trained with various hidden layer sizes. Hidden layer 1, the LSTM [37] layer, always contains twice as many units as hidden layer 2, the normal layer, and was tested with sizes of 100, 150, 200, 250 and 300.

All classifiers described in this section were trained to predict SBI sections and the results will yield insight in how different classifiers perform at the prediction of economic activity when using nothing but the text found of the homepages of all domains of e-commerce or businesses in the `.nl` zone.

3.3 RESULTS

The results of both the feature and classifier experiments are listed in this section. Furthermore, the word embeddings themselves are examined further, because the usage of those embeddings did not result in a higher classification performance than the use of tfidf vectors.

3.3.1 *Feature Extraction Methods*

The goal of this experiment was to determine which type of feature extraction method resulted in the highest classification performance. In Table 3.2, the accuracy and macro weighted F_1 score [84] of the linear classifier trained using SGD of the MH loss function are listed on the first row. The accuracy indicates the percentage of correct classifications. The macro weighted F_1 score shows the performance in terms of precision and recall for every class. The F_1 score will be lower if the classifier is biased toward a class, while the accuracy is not necessarily lower. Using tfidf vectors as features resulted in the highest classification performance with an accuracy of 0.70 and the use of idf weighted average word embeddings resulted in the lowest classification performance with an accuracy of only 0.46.

To further investigate why the use of embeddings resulted in lower performance, the Word2Vec model itself was manually evaluated by extracting the words most similar to a manually picked word and checking if those words were semantically related to the word. Table 3.3 shows the 10 most similar words to the word 'accountant'. All words were already stemmed, which explains the occurence of the word "accountantskantor". All words found imply a certain presence of accountants, tax or have something to do with bookkeeping. The Word2Vec model itself was therefore found to function properly.

Full classification reports and confusion matrices for all linear classifiers trained using SGD of the MH loss function per feature extraction method can be found in Appendix C. The confusion matrices indicated

	tfidf vectors	Word embeddings	idf weighted word embeddings
SGD (MH)	0.70 0.70	0.60 0.62	0.46 0.49
SGD (hinge)	0.66 0.67		
LR	0.69 0.69		
SGD (log)	0.67 0.67		
DT	0.48 0.48		
RF	0.64 0.64		
RNN		0.64 0.60	
RCNN		0.67 0.62	

Table 3.2: Accuracy and weighted macro F_1 scores for different feature extraction methods and classifiers. Linear classifiers trained using SGD of a loss function are indicated as "SGD (loss function)"

Rank	Word	Cosine distance
1	accountantskantor	0.788605
2	belastingadviseur	0.768227
3	accountancy	0.737502
4	registeraccountant	0.689978
5	fiscalist	0.662791
6	administratiekantor	0.655958
7	boekhouder	0.627767
8	administratieconsulent	0.608103
9	bedrijfsadviseur	0.605927
10	belastingadvieskantor	0.601040

Table 3.3: Cosine distances to the word 'accountant'.

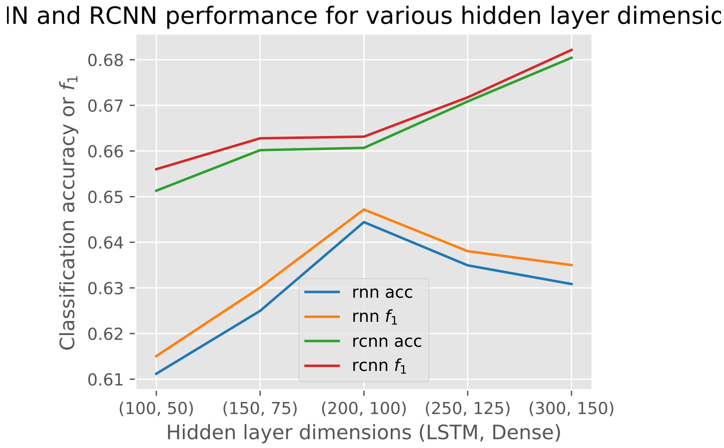

Figure 3.1: RNN and RCNN accuracy and F_1 score for various hidden layer sizes.

that every classifier made similar mistakes, but to a higher extent for the less well performing classifiers.

3.3.2 *Classifiers*

Various classifiers were tested to measure their performance in tackling the economic activity determination problem. Table 3.2 also contains the accuracy and macro weighted F_1 scores are listed per classifier. The linear classifier trained using SGD of the MH loss function achieved the highest performance with an accuracy of 0.70. The LR classifier performed second best with an accuracy of 0.69. The DT classifier performed worst with an accuracy of 0.48. A full classification report and a confusion matrix per classifier can be found in Appendix D.

Various hidden layer sizes were experimented with to determine the optimal configuration of the final neural network. The value of the semantic layers (convolution and pooling) was also determined by comparing an RCNN with an RNN. Figure 3.1 shows the performance of both the RNN and RCNN for different hidden layer sizes. The RCNN proved to be superior with an accuracy of 0.67, while the RNN reached an accuracy of 0.64. Notable is that the performance of the RNN kept increasing as the number of hidden units became bigger, while the RCNN shows a small drop in performance at the highest dimensions. Confusion matrices for the best performing RCNN and RNN can be found in Appendix D.3.

3.4 DISCUSSION

In this chapter, an experiments to investigate research question 1 and 2 are described. These research questions entailed the influence of the feature extraction method on classification performance and the influence of the classifier type on classification performance.

Three different feature extraction methods were used: tfidf vectors, average word embeddings and average word embeddings with idf weighting. Subsequently a linear classifier trained using SGD of the MH loss function was trained for each of those features. Contrary to the hypothesis, the use of tfidf vectors resulted in the highest classification performance, while the use of average word embeddings with idf weighting resulted in the lowest classification performance. Both these outcomes are surprising, because previous work shows that the use of word embeddings is a powerful method of representing text (e.g. [42, 48] and idf weighted word embeddings generally improve performance compared non weighted word embeddings (e.g. [4, 19, 55]).

A possible explanation for inferior performance when using word embeddings is that for every document, all embeddings were combined in one single embedding, reducing the information carried by the final embedding. This is reflected by the fact that both the RNN and RCNN classifiers, which were trained using full word embeddings, performed similar to the other classifiers. The tfidf vectors were not altered, meaning that they still contained all information that was originally there. Another possibility is that word embeddings are less powerful when used as features for longer documents. One controversial result is the sudden drop in performance when using average word embeddings with idf weighting. One should expect the performance to be higher, because using tfidf vectors results in the highest performance, but instead it is lower. The Word2Vec model was examined more closely to determine if it was functioning correctly, which was the case. The tfidf vectors were assumed to function properly as well, because the use of tfidf vectors resulted in the highest performance. Therefore, further work is required to investigate why the use idf weighted word embeddings resulted in such a sudden drop in classification performance.

Several different classifiers were employed as well: linear classifier trained using SGD of the hinge, log and MH loss functions, a LR classifier, a DT classifier, a RF classifier, a RNN classifier and an RCNN [48] classifier. Every classifier with exception of the RNN and RCNN were trained using tfidf vectors, because those vectors resulted in the highest performance. The RNN and RCNN were trained using full word embeddings, because these classifiers could handle the multiple dimensions of those features. Multiple dimensions for the hidden layers were tested as well in order to determine the optimal configuration. The main function of the RNN classifier was to determine the value of the convolutional and pooling layers of the RCNN. The RNN was consistently outperformed by the RCNN, implying that the semantic context of a word yields information relevant for classification. However, the difference is relatively small with only 3 to 4 percent in terms of accuracy or F_1 score.

The linear classifier trained using SGD of the MH loss function achieved the highest performance, contrary to the hypothesis. The hypothesis was that the more complicated RCNN with the more complicated word embedding features would achieve the highest performance. While the word embedding based RCNN achieved almost similar performance, both the classifier and the feature extraction methods are computationally much heavier and much more complicated. In order to keep the classifier and the decision it makes understandable, the choice for the linear classifier trained using SGD of the MH loss function was made to investigate the influence of the flat or hierarchical approach on the classification performance in Chapter 4. Training classification systems costs computation power. Simpler systems require less power, which is why classification systems should be kept as simple as possible. Further reading about the energy cost of training can be found, i.e. [72, 78].

Both these results prove that more complicated feature extraction methods or more complicated classification schemes do not guarantee higher performance. While these more complicated methods may result in higher performance when testing with a benchmark dataset such as the 20NG [49] dataset, they do not perform better at the task of economic activity classification based on text found on web pages.

4 IMPROVING CLASSIFICATION PERFORMANCE BY EXPLOITING THE HIERARCHICAL STRUCTURE

The use of hierarchical information can lead to higher classification performance [74]. This chapter aims to thoroughly investigate whether or not hierarchical classification can achieve higher performance than traditional "flat" classification as described in the previous chapter (research question 3). The hypothesis is that the hierarchical classification system would outperform the "flat" classifier. The intention is to prove that the concept of hierarchical classification will generalize from benchmark datasets to a real-life problem such as economic activity classification for web pages. This chapter contains a background on hierarchical classification approaches, applications and challenges in Section 4.1, the methods and results of the experiment conducted in Section 4.2 and Section 4.3 and concludes with a short discussion to answer the research question and interpret the results in Section 4.4.

4.1 BACKGROUND

Hierarchical classification is the task of exploiting an existing structure in the data to improve classification performance. An elaborate background on how hierarchical classification can be done is given in this section. First, several possible approaches to hierarchical classification as identified by Silla and Freitas [74] are described. Second, examples of applications of hierarchical classification are given for textual, visual and auditory modalities and protein classification. Last, possible limitations of hierarchical classification are given.

4.1.1 *Approaches*

Contrary to default "flat" classification, hierarchical classification aims to exploit an existing structure in the data. Different approaches have been distinguished [74]. An illustration of how "flat" classification works can be helpful to understand hierarchical classification. Given an existing structure in the data, only one single classifier is trained and any hierarchical information is disregarded. This approach is shown in Figure 4.1. In this figure, each node represents a label. The classifier does not consider the labels in nodes A, B, C, but only the labels in leaf nodes. The labels considered by a classifier are grouped in the blue outline. Nodes A, B, C are the toplevel nodes. Child nodes of toplevel nodes are sublevel nodes. Any hierarchical context from

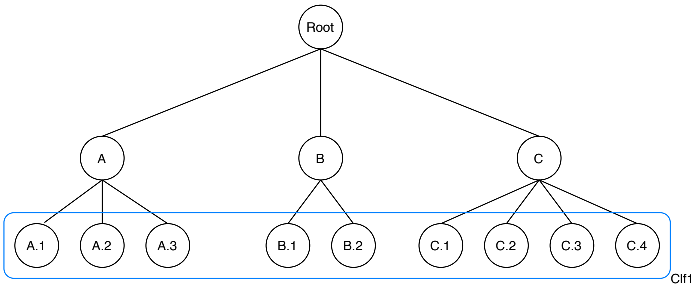

Figure 4.1: Illustration of a default "flat" classification model. The blue rectangle indicates between which classes the classifier Clf1 differentiates.

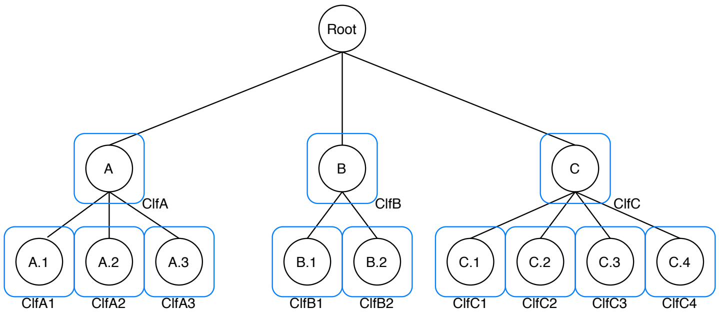

Figure 4.2: Illustration of a LCN classification model. The blue rectangles indicates between which classes a classifier Clf* differentiates.

the parent nodes is completely ignored. The classifier only considers the leaf nodes.

The first hierarchical approach examined is the LCN approach [74]. A binary classifier is constructed to recognize whether or not a sample belongs to a category for every node in the tree. Figure 4.2 illustrates this approach. This approach features natural multi-label incorporation and is simple to implement and understand.

A slightly more complicated approach is the LCPN approach. This approach differs from the LCN approach in the fact that for every parent node a different multi-class classifier is constructed. Figure 4.3 illustrates how LCPN functions. While this approach struggles with multi-label problems, less classifiers are trained compared to when using the LCN approach while it is still relatively simple to implement and understand.

The LCL approach is hardly ever used in the literature [74]. This approach trains a flat classifier for every level in a hierarchy. Hierarchical classification depends on the divide and conquer strategy and this approach makes minimal use of this strategy, which may be the reason it is sporadically used. Figure 4.4 illustrates this approach.

The final approach is the highly custom global or big bang approach. In this approach, a single model is trained for all classes. This model considers all class membership constraints in a straightforward manner in both training and testing phases, usually based on mutual

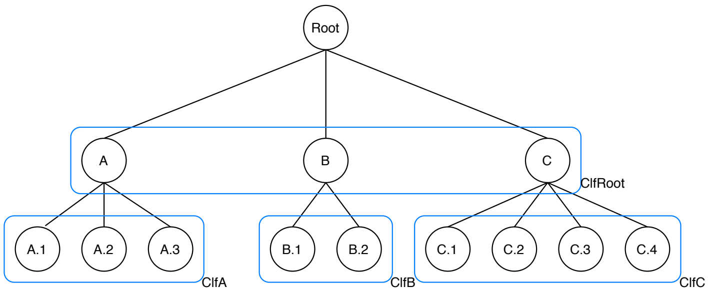

Figure 4.3: Illustration of a LCPN classification model. The blue rectangles indicates between which classes a classifier Clf* differentiates.

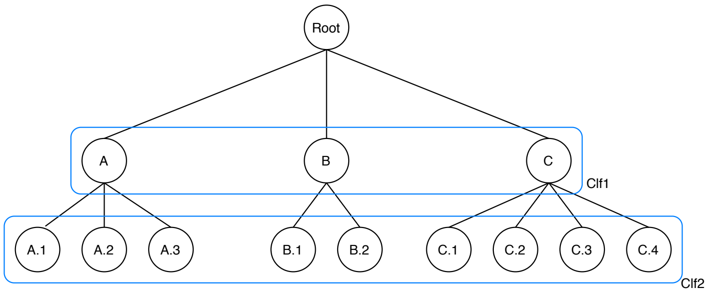

Figure 4.4: Illustration of an LCL classification model. The blue rectangles indicates between which classes a classifier Clf* differentiates.

exclusion. Because only a single model is trained, the classification performance is heavily dependent on the underlying learning algorithm and cannot be fine-tuned for subtrees, e.g. node C and its child nodes in Figure 4.5. This figure also illustrates the global approach.

4.1.2 *Applications*

Hierarchical classification has been used across different modalities. In text classification, datasets such as the 20 Newsgroups [49], OHSUMED [36] and the US Patent Database are hierarchical in nature. Other problems, across all modalities, can be transformed in their

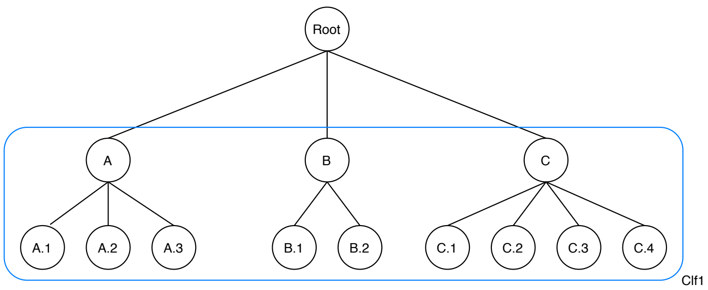

Figure 4.5: Illustration of a global or big bang classification model. The blue rectangle indicates between which classes classifier Clf1 differentiates.

hierarchical counterpart by creation of sub or super categories. An example of the creation of a document hierarchy is to generate one based on topic correlation [56].

4.1.2.1 *Text Classification*

The hierarchical approach can be used to improve text classification. Different datasets can be used to test methods. For example, the 20 Newsgroups dataset [49] is hierarchical in nature and can therefore be used to perform hierarchical document organization. The dataset will now have 7 toplevel categories and 16 sublevel categories, some of which also have child nodes. The OHSUMED [36] database is a subset of the MEDLINE database. It contains titles and abstracts from 270 medical journals in 23 categories, which are called Medical Subject Headings (MeSH). This dataset is not a tree, but a Directed Acyclic Graph (DAG), which may complicate learning. For example, Cesa-Bianchi, Gentile, and Zaniboni [15] removed the nodes which did not have a unique path to the root node, because their algorithm could not deal with DAGs. In for example [33, 68, 91], the authors show that a hierarchical approach outperforms the flat approach when tasked with text classification using the OHSUMED dataset. Chakrabarti et al., Yoon, Lee, and Lee also tested their hierarchical approach with the 20 Newsgroups dataset and it outperformed the standard flat approach. News article datasets (such as Reuters datasets) are inherently hierarchical. For example, in [45], the authors propose the LCPN approach for this problem and achieve higher performance when using this approach than when using the flat approach.

The US Patent Database also lends itself perfectly for hierarchical classification, because all patents are stored according to the hierarchy of the database. Therefore, labeled data is readily available. In [50], the use of an hierarchical approach did not yield significantly better performance than a flat method. Chakrabarti et al. [16] however shows that the hierarchical approach can outperform the flat approach in patent classification.

4.1.2.2 *Image Classification*

In image classification, hierarchies can be of use as well. For example, objects in images, (i.e. animals) can also be hierarchically ordered. The ImageNet dataset [20] is exceptionally suitable for hierarchical image classification. This dataset consists of 14 million labelled images in 27 top-level categories. For example, in [41], the authors use hierarchical classification to outperform flat classification.

In [38], a hierarchical system for live fish recognition with a reject option is created. The reject option is used if the confidence of the classifier decisions is below a threshold. While the bigger contribution of this paper is the fish recognition application, their approach outperforms both the flat approach and other hierarchical approaches.

The Princeton Shapes Database [73] can also be used to test hierarchical classification models. Barutcuoglu and DeCoro [8] show that using a Bayesian aggregation of a tree can correct class inconsistencies and improves classification performance in both tree and DAG hierarchies.

Another hierarchical image classification task can be found in the medical imaging field. In [21], the authors apply a global hierarchical image classification algorithm to the medical image annotation task from ImageCLEF2009 [81]. The data consists of X-ray images annotated by 4 axes: the Technical (modality) axis, the Directional (body orientation) axis, the Anatomical region axis and the Biological system axis. The annotations are provided by the IRMA group [52]. The authors find that the hierarchical approach outperforms flat approaches and even was the best performing method reported in the literature.

Another visual domain in which hierarchical classification can be employed is the prediction of crops based on satellite images [62]. In this work, satellite image is used to predict which crops are in which fields and where the fields are. The hierarchy consists of two toplevel classes, which contain three and six sublevel classes, respectively. The authors find that the best accuracy is achieved using hierarchical classification based on SVMs.

4.1.2.3 *Sound Classification*

Hierarchical classification can be employed in the auditory domain. For example, in [88], the authors propose to use different classifiers for male and female utterances and subsequently determine the sentiment carried in the utterance. This addition to the model improved the classification performance. Another auditory problem is to identify a bird by its song. Goëau et al. [32] conclude that not using the background species in classification actually improves the performance. This contradicts any other research in hierarchical context. No explanation is given for this phenomenon.

Hierarchical structures have also been used to improve performance in music genre classification. Li and Ogihara [53] propose to use two manually generated two-level hierarchies for the task. The idea is that music genres are not completely independent of one another, but that different genres can show similar characteristics. The authors find that the usage of the hierarchies improves classification performance. In this publication, the authors also propose a method to create hierarchies automatically in a data-driven fashion.

4.1.2.4 *Protein Classification*

The last research field in which hierarchical classification is widely employed is in protein function prediction. The biggest challenge here is that the function of very similar proteins can differ greatly. Different hierarchies exist, such as the Enzyme Commission [7] and the Gene Ontology [5]. In [47], the hierarchical approach once again proves better than the flat approach.

4.1.3 *Challenges*

One thing that LCN, LCPN and LCL all have in common is that the final model is a combination of smaller classification models. These components can be fine-tuned and adapted as necessary. However, these approaches also suffer from drawbacks. For example, an error made by a top-level classifier is propagated to lower levels in the tree and cannot be corrected anymore.

Furthermore, most of these models assume mandatory leaf node prediction. However, if a document is more general, for example a survey or review paper, it does not fit in a leaf node, but should be classified into an intermediary node. A solution to achieve this is to start classification at the root node, set thresholds at intermediary nodes and only continue classification if the threshold value is reached. This value is usually a minimal classifier confidence, but can also be an entropy measure of confidence scores over child nodes [58]. A method to automatically compute thresholds was proposed by Ceci and Malerba [14].

Using thresholds however also introduces another problem, namely the blocking problem [79]. Blocking occurs during the testing phase if a data sample is rejected by an intermediary classifier and thus not reaches it associated expert classifier. For example, in Figure 4.2, if the classifier at node B does not reach the threshold confidence, it will not be passed down to the classifiers at leaf nodes B.1 and B.2. To address this problem, three methods are suggested. The first method is threshold reduction, which consists of lowering the threshold value of the subtree classifier at a node. This results in passing more samples to child nodes. The second method is to use restricted voting, which consists of using the subtree classifier to link a node to its grand-parent node. The idea is to allow more specialized classifiers access to a data sample before it is rejected. The third and final method is to use extended multiplicative thresholds, by recursively using the multiplicative thresholds which only worked for hierarchies of two levels [25]. To illustrate how these multiplicative thresholds function, we give a small example. Given the tree in Figure 4.2 and a subset of confidence scores of the classifiers at nodes $c_A = 0.7$, $c_{A.1} = .9$, $c_{A.2} = 0.3$, $c_{A.3} = 0.5$ for a data sample and a static threshold of 0.5. The data sample is classified as $A.1$, because $0.7 \cdot 0.9 > 0.5$. If the multiplicative confidence would have been lower than 0.5 at A.1, A.2 and A.3, the sample would be classified as A.

Another challenge these models have difficulty coping with is the class inconsistency. In Figure 4.4 for example, the top classifier can predict B, while the bottom classifier predicts A.1. Because A.1 is not a child node of B, the final prediction is inconsistent with the tree structure. To cope with this problem, a bottom-up strategy is proposed [83]. Every leaf node classifier is evaluated. If the prediction at a leaf node is *True*, the prediction at its ancestors is also set to *True* if it was *False* to make the prediction consistent. This approach is more about correcting mistakes, but in another approach the tree is pruned based on document similarity to all documents and their classes, reducing class inconsistency in the testing phase [89]. The authors report a 77.7% increase in classification performance at depth 5 of the Open Directory Project.

4.2 METHODS

The methods of the experiment intended to answer research question 3 are described in this section. The LCPN approach was chosen and implemented for the following reasons:

- The LCPN approach does not struggle as much with class hierarchy inconsistency problems as LCN. The dataset was not multi-labeled, so the natural multi-label strength of the LCN approach was not required.
- LCL was not chosen, because it does not take any class hierarchy into account.
- A global model is not modular. Modularity is an important aspect of the final system, because every classifier can be evaluated individually and can be trained using the most distinctive features of the corresponding subset of the data. This process can be automated as well [85]. As shown in Section 2.2, the data is very diverse and imbalanced, which requires different approaches to achieve optimal classification performance.

The use of various feature inputs and classification methods were explored in order to find the most suitable feature types and classifiers for the classification of economic activity based on text in Chapter 3. The use of tfidf vectors as features in combination with a linear classifier trained using SGD of the MH loss function resulted in the highest performance. Therefore, the classification pipelines of both the traditional "flat" classifier and the LCPN classification system consisted of a tfidf feature extractor and a linear classifier trained using SGD of the MH loss function. Both components are described in Chapter 3.

The flat classifier consist of a linear classifier trained using SGD of the MH loss function which is trained to infer division labels from tfidf vectors.

The LCPN classification model consists of multiple components:

- The section linear classifier trained using SGD of the MH loss function based on tfidf vectors. The corpus of this tfidf vector extractor contained the 50000 most occurring words in all of the data.
- For every section, a linear classifier trained using SGD of the MH loss function based on tfidf vectors. These classifiers were trained using only data from the corresponding section, i.e. a classifier for section A was trained using only data in division A.1, A.2 and A.3. The corpus of the tfidf vector extractor contained only the 50000 most occurring words found in that section as well in order to avoid non-representative texts for that section.

In order to answer the research question, namely if hierarchical classification can achieve higher performance in this specific problem, the "flat" classifier and the LCPN classification model are both trained and evaluated on the same training and testing data. The results are documented in Section 4.3. These results will yield insight in how useful an LCPN classification model can be in economic activity classification when compared to a regular "flat" classifier.

4.3 RESULTS

The results of the experiment regarding "flat" (non-hierarchical) versus hierarchical classification are described in this section. In Table 4.1, the accuracy and the macro weighted F_1 score [84] of both the "flat" and the hierarchical approaches to division classification are listed. The accuracy indicates the percentage of correct classifications. The macro weighted F_1 score shows the performance in terms of precision and recall for every class. The F_1 score will be lower if the classifier is biased toward a class, while the accuracy is not necessarily lower. The hierarchical approach results in the highest performance with an accuracy of 0.64, almost equal to 0.63, which is achieved when using the "flat" approach. Full classification reports and confusion matrices can be found in Appendix E. Confusion matrices for every component of the LCPN classifier are included to be able to further investigate the components of the LCPN classifier.

4.4 DISCUSSION

The difference in classification performance when using the "flat" or the hierarchical approach is investigated in this chapter, which is

Approach	Accuracy	F_1
"Flat"	0.63	0.63
Hierarchical	0.64	0.64

Table 4.1: "Flat" and hierarchical classifier accuracy and macro weighted F_1 scores.

research question 3. The hypothesis for this research question was that the hierarchical approach would achieve a higher classification performance in terms of accuracy than the "flat" approach. In order to test this hypothesis experimentally, a single division classifier was trained and tested to find the performance for the flat approach. Next, a system of classifiers according to the LCPN approach was trained and tested for division classification. All classifiers were linear classifier trained using SGD of the MH loss function and were based on tfidf vectors, because these features and classifier resulted in the highest performance in terms of both accuracy and macro weighted F_1 score. The experiments related to these findings are described in Chapter 3. The performances were compared and the use of the LCPN approach resulted in slightly higher performance than when using the "flat" approach. This proves that the concept of hierarchical classification generalizes from benchmark datasets to a real-life problem such as economic activity classification for web pages. This result is in line with most research done in hierarchical classification (further reading in [74]). Most of the experiments described in this survey conclude that hierarchical classification at least slightly improves classification performance.

This project solely focussed on constructing a basic LCPN classification model. This model used a toplevel classifier to select a sublevel classifier. Therefore, the LCPN system is highly dependent on the toplevel classifier. Possible improvements such as pruning of the tree based on document similarity [89] and a bottom-up approach to correct mistakes made by the toplevel classifier [83] are described in the background section of this chapter. The toplevel classifier has achieved an accuracy of 0.70 for sections (see Chapter 3 and the LCPN division classifier reached an accuracy of 0.64, which is only slightly lower.

Multiplicative classifier confidence based classification was explored as well for the LCPN classification model. Every classifier prediction includes a confidence, which is a probability. By multiplying the confidence for a section with the confidence of a divisions in that section, a multiplicative confidence is found. Once this action is performed for all divisions, a final prediction, corresponding to the highest multiplicative confidence, can be extracted. Unfortunately, this approach resulted in decreased performance when compared to only selecting one division classifier with the section classifier. A possible reason is that the classifier was always relatively certain about a prediction (e.g. most class probabilities were zero), effectively nullifying the multiplicative confidence for those classes. Further investigation of this phenomenon is required in order to confirm this explanation.

5 DEPLOYMENT

The deployment of the resulting hierarchical classifier is discussed in this chapter. The first method is the implementation in DMAP, which is monthly used to generate a prediction for all business and e-commerce related domains in the `.nl` zone. These predictions are subsequently used at SIDN to gain more insight in the zone. The second method is exposure in the form of a web page. The main goal of this web page is to utilize the knowledge of everyone interacting with the web page to obtain additional labeled data.

5.1 ZONE PREDICTION

This research project resulted, in addition to valuable insight, in a hierarchical classification system. SIDN decided to take this classification system into production due to its high performance compared to the previous model and its ability to perform division classification. In classification problems, it is also possible to predict the most probable n labels. However, generating a full probability distribution for both sections and divisions per domain is computationally expensive. Therefore, it was decided that classifier output would consist of two components:

1. A probability distribution over the sections.
2. A probability distribution over the divisions in the most probable section.

Only two classifier predictions are necessary to generate these two outputs. Therefore, it is relatively cheap from a computational perspective, especially when compared to using all 16 classifiers in the LCPN model. Still, this approach allows division classification.

Economic activity predictions were generated for all business and e-commerce related domains in the `.nl` DNS zone. A comparison between the predicted and actual distribution can be found for both section and divisions in Figure 5.1 and Figure 5.2, respectively. In both these figures, blue is the data distribution as provided by the CBS, the actual data distribution. Orange is the data distribution as provided by the classifier. Green is the difference between those distributions. For example, section M in Figure 5.1 is the most underrepresented in the `.nl` zone.

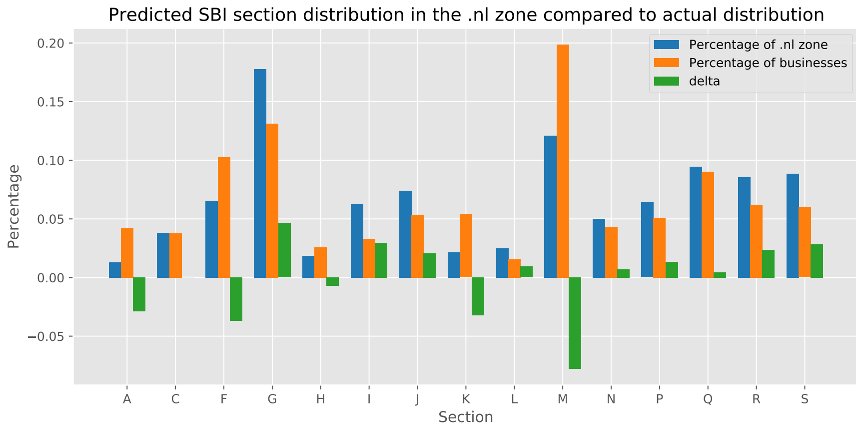

Figure 5.1: The data distribution as provided by the CBS (blue), the classifier (orange) and the difference between those distributions (green). For every section (x-axis) is indicated what percentage of the data it occupies (y-axis).

5.2 FEEDBACK WEBSITE

The classifier was exposed as a web application[1] as well. This website was designed to obtain new labeled data. Figure 5.3 shows a typical usecase for this website. A screenshot of the functional part of the web application is shown in Figure 5.4. First, the user is asked to enter an URL. Once the user clicks the predict button, the text found on the homepage of the corresponding domain is classified and the output is shown to the user. Now, the user is asked to provide feedback. The prediction can be either right or wrong. In the case of a faulty prediction, the user is also asked to provide both the correct section and the correct division. After completing the ReCaptcha challenge, the user can click the send feedback button. Then the feedback is registered and the user thanked for providing feedback. The user can choose to exit after every action. Therefore, it is also possible to use the web application without providing feedback.

1 https://webcola.sidnlabs.nl/

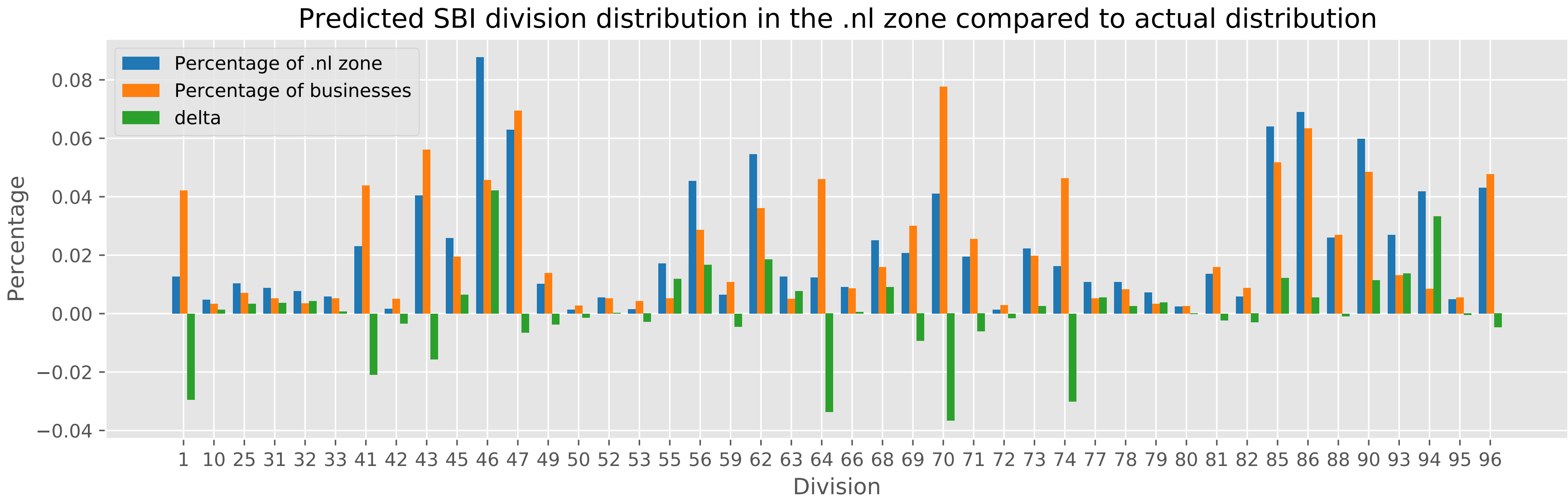

Figure 5.2: The data distribution as provided by the CBS (blue), the classifier (orange) and the difference between those distributions (green). For every division (x-axis) is indicated what percentage of the data it occupies (y-axis).

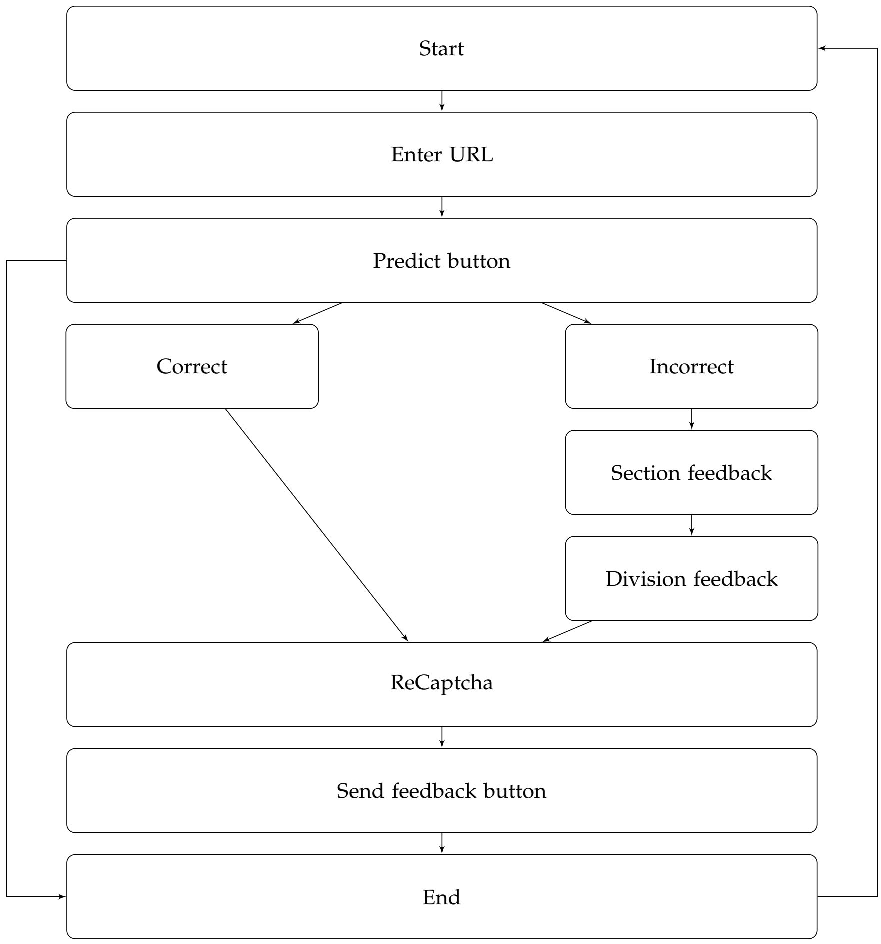

Figure 5.3: This figure illustrates the possible usecases of the web application.

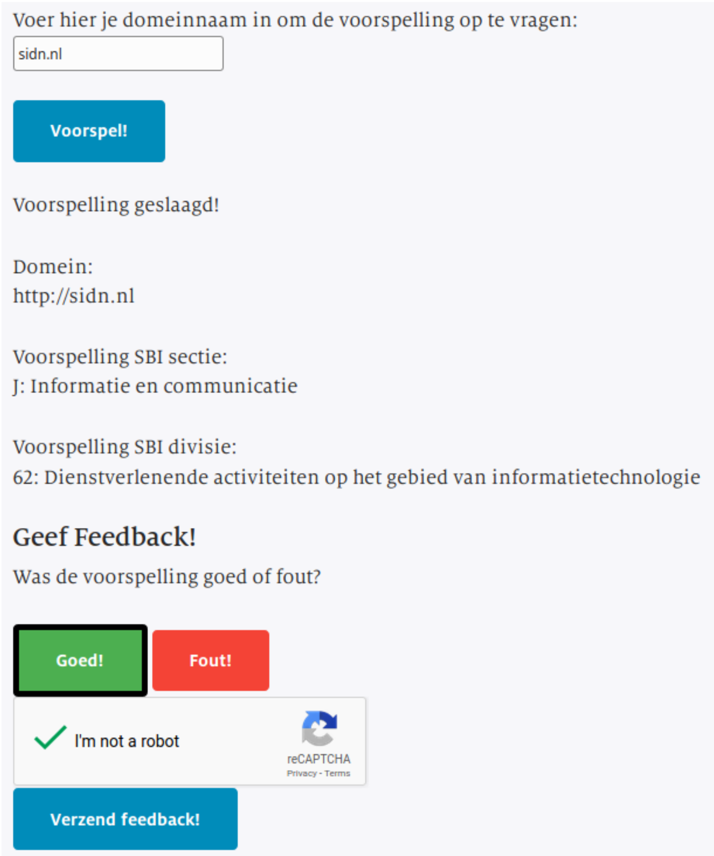

Figure 5.4: Screenshow of the functional part of the web application.

Part III

THE DISCUSSION

6 CONCLUSION

Most more complex concepts for feature extraction, classification and hierarchical classification result in increased performance in the literature. However, those concepts are usually only tested on a small selection of benchmark datasets. This research project was aimed to validate the use of those concepts using a real-life and ever-changing dataset, namely the text found on domains in the `.nl` zone.

Three aspects of the economic activity classification for web pages were investigated in this research project. The three main research questions were answered:

1. The use of different features influences classification performance. Using tfidf vectors as features resulted in the highest classification performance.

2. The use of different classifiers affect performance. Using a linear classifier trained using SGD of the MH loss function based on tfidf vectors as features resulted in the highest classification performance.

3. Performance of a classification system can be slightly improved by using a basic hierarchical classification model.

To conclude, more complicated features and classifiers do not guarantee increased classification performance. Furthermore, the performance increase with hierarchical classification when tested using benchmark datasets generalizes to a non-benchmark problem.

7 DISCUSSION

This research project included a set of pilot experiments in order to allow for an early identification of possible future problems and to make an optimal selection of the data.

The first task was to get familiarized with the data itself in order to identify possible problems such as imbalanced data early and make a final data selection. The label distribution itself was found to be heavily imbalanced, which was to be expected. The number of companies per economic activity differs. As a countermeasure, class weights were introduced in order to avoid classifier bias toward majority classes. The label distribution was however found to be statistically similar to the actual label distribution on both section and division level as provided by the CBS. This means that any results found based on the labeled domains in the `.nl` zone are expected to generalize to all domains in the `.nl` zone.

The second task was to obtain additional labeled data. A name and address based match between the KvK database and the domain registration data was conducted. This match resulted in additional labeled data, but after testing, this new data proved to contain too much noise and only decreased classification performance. Therefore, this data was disregarded.

The third task was to select the labels which were going to be considered. This is an optimization problem, because the classifier performance was to be maximized while still being informative, which means keeping as many labels as possible. In order to maximize performance, divisions with less than 4000 members (according to the CBS) were removed under the pretense that those domains could be manually labelled if required.

A limitation of the dataset itself is that it was (partly) labeled by starting entrepreneurs. These persons do not know the SBI and can therefore make mistakes when registering their business. It is also possible that they decide to participate in another economic activity after registering their business. This problem can be approached by manually checking whether or not the listed economic activity is correct by professionals. In this research project, several but far from all domains and the corresponding economic activity were checked. This check was passed successfully.

Three main research questions were investigated in this project:

1. How important is the usage of different features with regard to classification performance in the economic activity classification problem? The hypothesis was that the more complicated idf

weighted word embeddings would result in the highest performance.

2. How does the use of different classifiers affect classification performance? The hypothesis was that the RCNN [48] would achieve the highest performance.

3. Can classification performance of the classification system be improved by using hierarchical classification? The hypothesis was that hierarchical classification would result in a higher performance than default "flat" classification.

In the first experiment, the use of tfidf vectors and (idf weighted) word embeddings is examined. Contrary to the hypothesis, the use of tfidf vectors resulted in the highest classification performance. A limitation emerged in this experiment: non NN classifiers could not handle multi-dimensional data samples. Therefore, the two-dimensional (idf weighted) word embeddings needed to be transformed to a single dimension. It is possible information was lost in this process. To account for this loss, NN classifiers which could handle the two-dimensional word embedding were trained in the second experiment.

In this experiment, tfidf vectors were used as input for various non-NN classifiers. Word embeddings were used as input for NN classifiers. Contrary to the hypothesis, the linear classifier trained using SGD of the MH loss function outperformed all other classifiers in terms of both accuracy and macro weighted F_1 score. These two experiments show that the use of more complicated methods do not guarantee increased performance.

In the third experiment, these findings were combined to create a LCPN (as identified in [74]) hierarchical classifier which was compared to a single classifier, which is a "flat" classifier. The hypothesis was correct: the hierarchical LCPN classifier slightly outperformed the "flat" classifier, which was in line with most of the work summarized by Silla and Freitas [74]. The LCPN classifier implemented was basic: the section classifier selected a division classifier to provide a final classification. Several improvements are possible to enhance the LCPN classifier. Possible improvements include pruning of the tree [89] and correcting the first classifier in a bottom-up fashion [83]. Further work is required to analyze the effects of those improvements.

Several combinations of features and classifiers were not considered. For example, the use of tfidf vectors combined with a RCNN is not considered. The use of such a NN in hierarchical context is also not considered, because it was already outperformed by the linear classifier trained using SGD of the MH loss function in section classification and hierarchical classification as in a LCPN heavily depends on the highest level classifier. One thing all excluded experiments have in common is that one component resulted in inferior performance.

In order to counteract data imbalance issues, class weights were introduced. The use of data augmentation to circumvent this problem was not investigated. Consider for example the generation of documents with a label according to the word distribution found in the existing documents carrying that label. Such documents could help to make a classifier more robust and less prone to error. SMOTE [17] uses this concept to generate additional labeled data for minority classes. Another possibility is to use balanced training batches. Further work is required to analyze the value of such methods for this specific problem.

This research project proved that the concept of hierarchical classification generalizes from benchmark datasets to a non-benchmark dataset such as the text found on domains in the `.nl` zone in the task of economic activity classification.Another valuable lesson can be learned from both the feature related experiment and the classifier experiment: more complicated methods do not guarantee increased classification performance. Both these lessons can be considered for any future experiments in both text and hierarchical classification.

The project itself was deemed a success: SIDN uses the LCPN classifier in their monthly crawl of the `.nl` zone. It is also possible to interact with the classifier to obtain live predictions for a domain of the user's choice[1].

1 https://webcola.sidnlabs.nl

Part IV

APPENDIX

A TEXT CLASSIFICATION APPLICATIONS

News article classification is the task of determining the already existing category a news article belongs to. Because vast amounts of news articles are written every day by many different organizations, it is problematic to manually assign all news articles to the correct category. Reuters-21578 is a popular benchmark dataset to test news article classification methods (e.g. [24, 66]). An advantage of using news collections to test text classification approaches is that news articles will always appear in a category. This category can be used as a label, eliminating the need to hand-label data.

Determining the sentiment or emotion expressed in a text is another application of text classification. For example, human readers can easily extract sentiment from movie review. A deep NN with recurrent layers can achieve over 90% accuracy when predicting if a review is positive or negative [90]. This classifier can be used to automatically determine a star rating for a movie instead of being dependent on individuals rating a movie without any other context. Sentiment analysis can also be performed in Twitter messages. For example, previous work describes a method to predict a positive or a negative sentiment given a query for Twitter messages [31]. The authors achieve a prediction accuracy of 82.7% when using a Naive Bayes or Maximum Entropy classifier and training on bigrams of words. The novelty in this work lies in not using actual labeled data, but the use of emoticons as noisy labels. The authors call this approach distant supervised learning. Determining this sentiment by query term can be used by e.g. companies who wish to monitor customer satisfaction. Another popular sentiment analysis dataset is the SSTb dataset [75], which contains movie reviews as well. These reviews are represented as fully labeled parse trees annotated by three human judges. In the same publication, the authors also use several (deep learning) algorithms to analyse the sentiment of these reviews. This data is used by e.g. Dos Santos and Gatti [23] to perform sentiment analysis. The authors test their approach with both the SSTb and the STS datasets.

Document organization, similar to news article classification, is the task of assigning a given document to the correct class. The 20 Newsgroups dataset [49] is a popular document collection for this task. It consists, as the name suggests, of text messages of twenty newsgroups. Each class contains about one thousand messages. The dataset can be found and downloaded freely from the internet[1]. This

1 http://qwone.com/~jason/20Newsgroups/

dataset is widely used as a benchmark dataset to test new or verify older text classification approaches (e.g. [6, 29, 48]).

Document organization can also be performed in a more formal domain, namely patent classification. The international patent database is growing continuously and the granting of a patent depends on its similarity with other patents. Furthermore, patents are lengthy and contain a high amount of professional language, which further increases the effort required to manually classify the patent. These reasons imply the need for an automated classification system, because such a system will improve the workflow. In 2003, Fall et al. [27] introduced a new labelled dataset of patents according to the International Patent Classification (IPC) taxonomy, which consists of 451 subclasses divided over 114 topclasses. The collection of patents is used by several other researchers: in [82], several text mining techniques are explored with relation to patent analysis.

As email arose as an efficient and economic method of communication, it unfortunately also became a more popular tool to send unwanted texts. These texts are known as spam and generally range from attempting to extract information to advertising a certain product. Because separating those emails from regular emails can be a cumbersome task, the need for an automatic classification system became clear. For example, previous work describes a comparison of classification methods, all with their own advantages and disadvantages [92]. This research compares NN, SVM, naive Bayesian and J48 classifiers when employed for the email classification task based on tfidf features. The authors found that the J48 classifier, which generates a binary decision tree, performs best with an accuracy of 95.8 percent.

B

PILOT EXPERIMENTS

B.1 STOPWORDS

Set of removed words: {'some', 'couldn', 'for', 'wouldn', 'or', 'zijn', 'between', 'en', 'm', 'zal', 'waren', 're', 'such', "you'll", 'at', "that'll", 'heb', 'through', 'ook', 'we', 'niet', 'uit', 'itself', 'wie', 'off', 'more', 'herself', "aren't", 'voor', 'who', 'but', 'no', 'by', 'myself', 'whom', 'je', 'that', "shan't", 'all', "hasn't", 'zo', 'zonder', 'altijd', 'being', 'other', 'zij', 'meer', 'deze', 'about', 's', "doesn't", 'haven', 'she', 'with', "won't", 'any', 'hun', 'you', 'and', 'het', 'those', "mightn't", 'each', 'u', 'werd', 'doen', 'tot', 'door', 'ze', 'yourself', 'wil', 'after', 'not', 'what', 'the', 'should', 'alles', 'during', 'tegen', 'against', 'down', 'their', 'an', 'omdat', 'mustn', 'our', 'be', 'too', 'had', 'when', 'am', 'weren', 'it', 'na', 'into', 'dat', 'heeft', 'haar', 'wezen', 'wordt', 'doesn', 'dit', 'zelf', "you'd", 'he', 've', 'been', 'om', 'are', 'were', 'why', 'out', 'hasn', 'toen', 'zou', 'above', "wouldn't", 'of', 'own', 'have', "mustn't", 'zich', 'onder', 'themselves', 'very', 'hers', 'als', 'so', 'can', 'where', 'nor', 'wasn', 'der', 'toch', 'ourselves', 'hij', 'kon', 'niets', 'was', 'hadn', "don't", 'daar', 'ge', 'nog', "weren't", 'a', 'once', 'up', 'doch', 'as', 'geweest', 'nu', "wasn't", 'o', 'then', 'maar', 'under', 'there', 'ain', "hadn't", "it's", 'hoe', 'which', 'than', "haven't", 'hebben', 'dan', 'y', 'een', 'its', 't', 'want', 'now', 'wat', 'here', 'his', "you've", 'moet', 'men', 'geen', 'yourselves', 'both', 'aren', 'him', 'just', 'needn', 'iemand', 'they', 'to', 'al', 'will', 'on', 'do', 'te', 'll', 'from', 'andere', 'only', "should've", 'kunnen', "she's", 'if', 'did', 'having', 'ben', "couldn't", 'mijn', 'my', 'aan', 'ons', 'himself', 'theirs', 'these', 'below', 'reeds', 'while', 'isn', 'ja', 'won', 'op', 'yours', 'is', 'don', 'de', 'before', 'same', 'ours', 'in', 'eens', "shouldn't", 'hem', 'them', 'how', "isn't", 'shan', 'again', 'van', 'has', 'die', 'few', 'mightn', 'most', 'this', "you're", 'mij', 'does', 'bij', 'because', 'her', "needn't", 'worden', 'until', 'i', 'shouldn', 'kan', 'over', 'er', 'hier', 'ma', 'veel', "didn't", 'dus', 'naar', 'doing', 'iets', 'd', 'me', 'ik', 'further', 'didn', 'uw', 'your', 'met'}.

B.2 DATA DISTRIBUTIONS

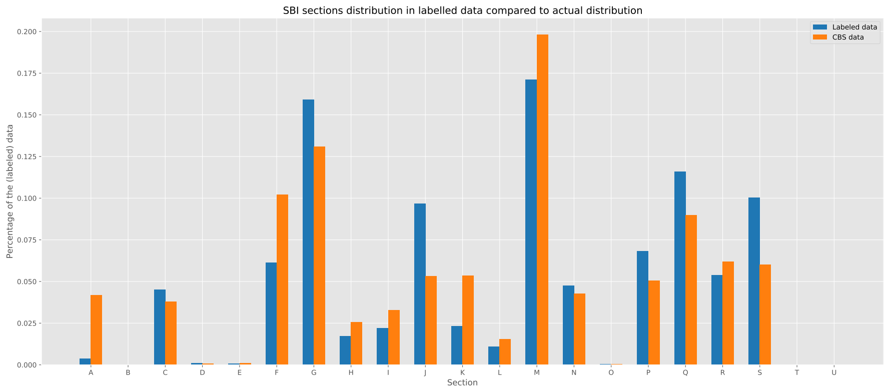

Figure B.1: Similarity between the labeled data distribution and the actual distribution on section level. On the x-axis, the sections are listed. On the y-axis is shown what portion of the data carries that section label.

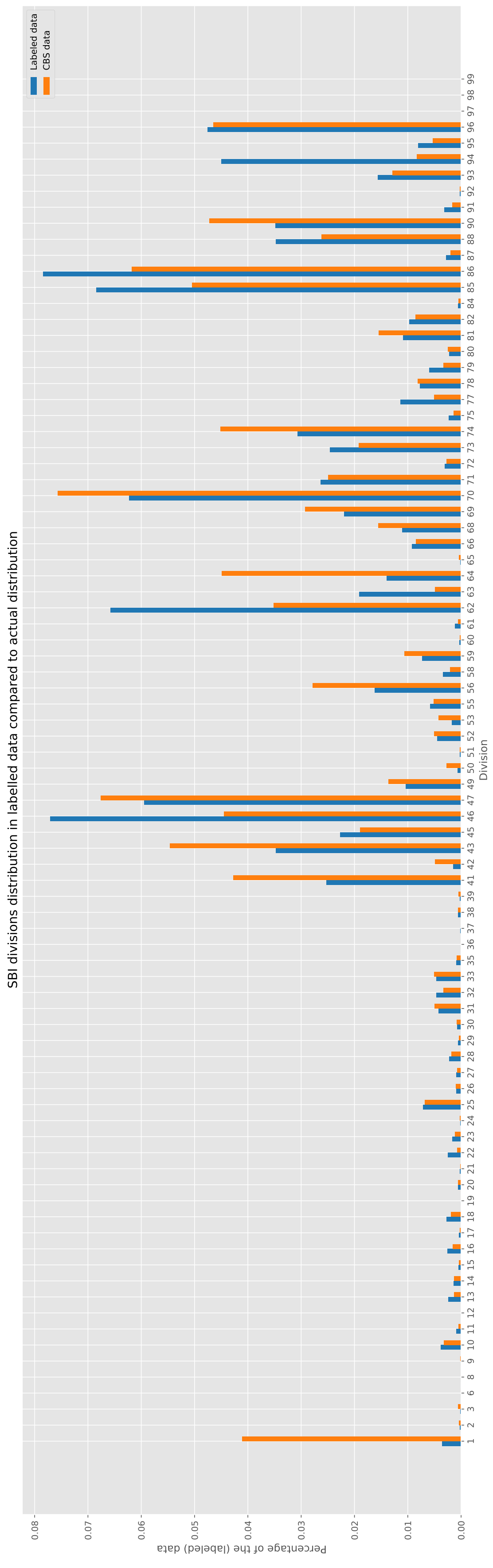

Figure B.2: Similarity between the labeled data distribution and the actual distribution on division level. On the x-axis, the divisions are listed. On the y-axis is shown what portion of the data carries that division label.

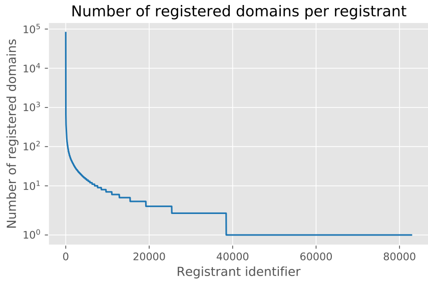

Figure B.3: Illustration of how many domains are typically registered per registrant.

B.3 MATCH DATA EVALUATION

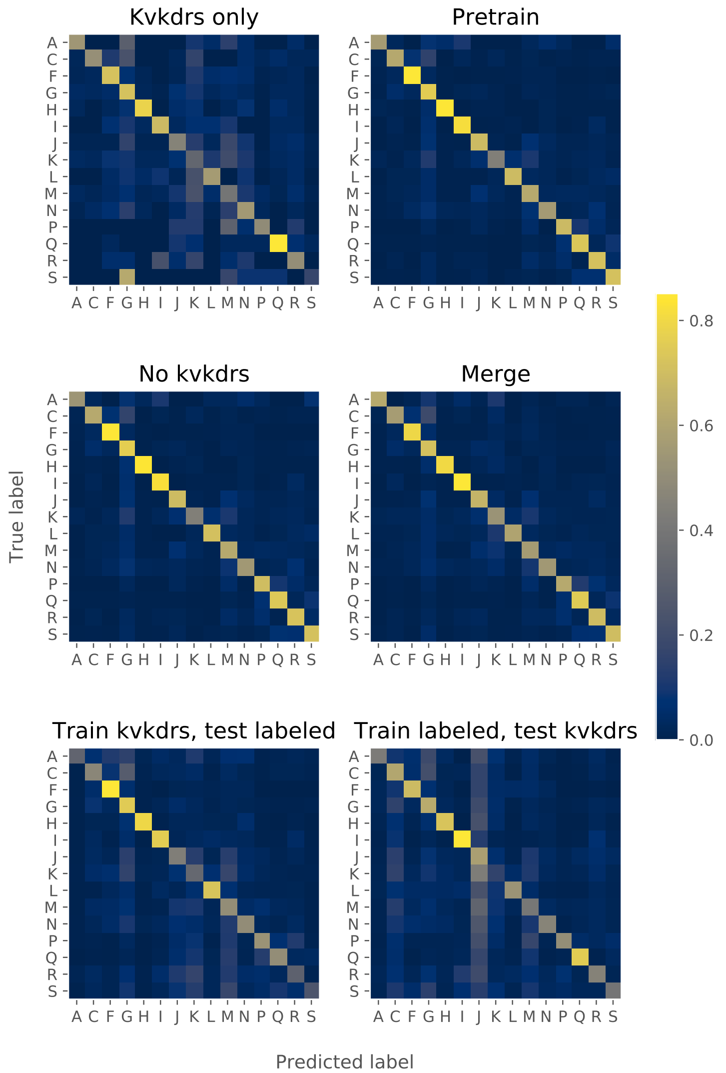

Figure B.4: Normalized confusion matrices for the classifiers which are used to evaluate the new match data. The confusion matrices are normalized on the true labels. On the x-axis, the predicted labels are listed and on every y-axis, the true labels are listed, both for every setting. A more yellow square at (x,y) indicates that the classifier often predicts label x when a true label is y.

	precision	recall	f1-score	support
A	0.229167	0.440000	0.301370	25
C	0.423077	0.415094	0.419048	53
F	0.507576	0.577586	0.540323	116
G	0.606498	0.579310	0.592593	290
H	0.508772	0.630435	0.563107	46
I	0.408163	0.555556	0.470588	36
J	0.290598	0.369565	0.325359	92
K	0.375000	0.262391	0.308748	343
L	0.277778	0.454545	0.344828	55
M	0.497778	0.320000	0.389565	350
N	0.288889	0.448276	0.351351	87
P	0.266667	0.400000	0.320000	20
Q	0.433333	0.684211	0.530612	38
R	0.142857	0.409091	0.211765	22
S	0.166667	0.142857	0.153846	14
accuracy			0.417139	1587
macro avg	0.361521	0.445928	0.388207	1587
weighted avg	0.440002	0.417139	0.417270	1587

Table B.1: New match data only.

	precision	recall	f1-score	support
A	0.276316	0.567568	0.371681	37
C	0.442254	0.620553	0.516447	253
F	0.765778	0.853125	0.807095	640
G	0.749703	0.762364	0.755981	1658
H	0.702326	0.848315	0.768448	178
I	0.646048	0.820961	0.723077	229
J	0.721081	0.695516	0.708068	959
K	0.465217	0.443983	0.454352	241
L	0.529801	0.695652	0.601504	115
M	0.790230	0.626067	0.698634	1757
N	0.610022	0.563380	0.585774	497
P	0.673973	0.691011	0.682386	712
Q	0.769366	0.742566	0.755728	1177
R	0.561848	0.719466	0.630962	524
S	0.756294	0.717973	0.736636	1046
accuracy			0.703881	10023
macro avg	0.630684	0.691233	0.653118	10023
weighted avg	0.713884	0.703881	0.705335	10023

Table B.2: Pretrain with new match data.

	precision	recall	f1-score	support
A	0.298507	0.540541	0.384615	37
C	0.440111	0.624506	0.516340	253
F	0.774148	0.851562	0.811012	640
G	0.747066	0.767793	0.757287	1658
H	0.704225	0.842697	0.767263	178
I	0.660839	0.825328	0.733981	229
J	0.717811	0.697602	0.707562	959
K	0.467249	0.443983	0.455319	241
L	0.535948	0.713043	0.611940	115
M	0.790043	0.623221	0.696787	1757
N	0.617450	0.555332	0.584746	497
P	0.666223	0.703652	0.684426	712
Q	0.767951	0.745115	0.756361	1177
R	0.574018	0.725191	0.640809	524
S	0.758865	0.716061	0.736842	1046
accuracy			0.705477	10023
macro avg	0.634697	0.691708	0.656353	10023
weighted avg	0.714712	0.705477	0.706550	10023

Table B.3: No new match data.

	precision	recall	f1-score	support
A	0.342105	0.629032	0.443182	62
C	0.427873	0.571895	0.489510	306
F	0.767857	0.797351	0.782326	755
G	0.735928	0.717804	0.726753	1949
H	0.651079	0.804444	0.719682	225
I	0.611413	0.852273	0.712025	264
J	0.668577	0.666667	0.667620	1050
K	0.417450	0.532534	0.468021	584
L	0.467890	0.600000	0.525773	170
M	0.769529	0.566002	0.652257	2106
N	0.629630	0.554031	0.589416	583
P	0.626886	0.624317	0.625599	732
Q	0.747755	0.753289	0.750512	1216
R	0.545455	0.702011	0.613909	547
S	0.728780	0.704717	0.716547	1060
accuracy			0.667844	11609
macro avg	0.609214	0.671758	0.632209	11609
weighted avg	0.682760	0.667844	0.670270	11609

Table B.4: Merge of new match data with the labeled data.

	precision	recall	f1-score	support
A	0.139618	0.289604	0.188406	404
C	0.353197	0.431269	0.388348	4714
F	0.644141	0.767511	0.700435	6396
G	0.615192	0.680417	0.646163	16581
H	0.608655	0.719288	0.659363	1799
I	0.451317	0.689414	0.545517	2286
J	0.416511	0.398255	0.407178	10084
K	0.084084	0.304797	0.131807	2418
L	0.280059	0.657689	0.392839	1151
M	0.512418	0.456198	0.482677	17819
N	0.445919	0.453282	0.449571	4966
P	0.463880	0.481669	0.472607	7119
Q	0.674421	0.458779	0.546082	12069
R	0.339899	0.277669	0.305648	5593
S	0.666857	0.222881	0.334097	10463
accuracy			0.480946	103862
macro avg	0.446411	0.485915	0.443383	103862
weighted avg	0.525310	0.480946	0.485303	103862

Table B.5: Train with new match data, test with labeled data.

	precision	recall	f1-score	support
A	0.259939	0.330739	0.291096	257
C	0.247981	0.470694	0.324829	1109
F	0.581784	0.535959	0.557932	1168
G	0.607959	0.489768	0.542501	2932
H	0.390421	0.563941	0.461407	477
I	0.448077	0.656338	0.532571	355
J	0.121386	0.449373	0.191140	1037
K	0.392793	0.124893	0.189524	3491
L	0.417122	0.411131	0.414105	557
M	0.586569	0.314752	0.409674	3552
N	0.471642	0.359909	0.408269	878
P	0.298893	0.395122	0.340336	205
Q	0.482474	0.585000	0.528814	400
R	0.154374	0.352941	0.214797	255
S	0.131661	0.304348	0.183807	138
accuracy			0.367795	16811
macro avg	0.372872	0.422994	0.372720	16811
weighted avg	0.457321	0.367795	0.377850	16811

Table B.6: Train with labeled data, test with new match data.

B.4 SELECTION OF SECTIONS AND DIVISIONS

Set of removed sections: {'O', 'T', 'D', 'U', 'B', 'E'}.
Set of removed divisions:{2, 3, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 51, 58, 60, 61, 65, 75, 84, 87, 91, 92, 97, 98, 99}.

Code	Description	Number of labeled domains
A	Landbouw, bosbouw en visserij	370
C	Industrie	2531
F	Bouwnijverheid	6396
G	Groot- en detailhandel; reparatie van auto's	16581
H	Vervoer en opslag	1778
I	Logies-, maaltijd- en drankverstrekking	2286
J	Informatie en communicatie	9582
K	Financiële instellingen	2409
L	Verhuur van en handel in onroerend goed	1151
M	Advisering, onderzoek en overige specialistisc...	17574
N	Verhuur van roerende goederen en overige zakel...	4964
P	Onderwijs	7118
Q	Gezondheids- en welzijnszorg	11777
R	Cultuur, sport en recreatie	5244
S	Overige dienstverlening	10461

Table B.7: Overview of section representation in the dataset.

Code	Description	Number of labeled domains
1	Landbouw, jacht en dienstverlening voor de lan...	370
10	Vervaardiging van voedingsmiddelen	394
25	Vervaardiging van producten van metaal (geen m...	738
31	Vervaardiging van meubels	438
32	Vervaardiging van overige goederen	482
33	Reparatie en installatie van machines en appar...	479

Continued on next page

Table B.8 – continued from previous page

Code	Description	Number of labeled domains
41	Algemene burgerlijke en utiliteitsbouw en proj...	2631
42	Grond-, water- en wegenbouw (geen grondverzet)	148
43	Gespecialiseerde werkzaamheden in de bouw	3617
45	Handel in en reparatie van auto's, motorfietse...	2363
46	Groothandel en handelsbemiddeling (niet in aut...	8027
47	Detailhandel (niet in auto's)	6191
49	Vervoer over land	1079
50	Vervoer over water	65
52	Opslag en dienstverlening voor vervoer	460
53	Post en koeriers	174
55	Logiesverstrekking	605
56	Eet- en drinkgelegenheden	1681
59	Productie en distributie van films en televisi...	753
62	Dienstverlenende activiteiten op het gebied va...	6844
63	Dienstverlenende activiteiten op het gebied va...	1985
64	Financiële instellingen (geen verzekeringen en...	1451
66	Overige financiële dienstverlening	958
68	Verhuur van en handel in onroerend goed	1151
69	Rechtskundige dienstverlening, accountancy, be...	2284
70	Holdings (geen financiële), concerndiensten bi...	6484
71	Architecten, ingenieurs en technisch ontwerp e...	2739
72	Speur- en ontwikkelingswerk	313

Continued on next page

Table B.8 – continued from previous page

Code	Description	Number of labeled domains
73	Reclame en marktonderzoek	2561
74	Industrieel ontwerp en vormgeving, fotografie,...	3193
77	Verhuur en lease van auto's, consumentenartike...	1178
78	Arbeidsbemiddeling, uitzendbureaus en personee...	797
79	Reisbemiddeling, reisorganisatie, toeristische...	622
80	Beveiliging en opsporing	232
81	Facility management, reiniging en landschapsve...	1129
82	Overige zakelijke dienstverlening	1006
85	Onderwijs	7118
86	Gezondheidszorg	8163
88	Maatschappelijke dienstverlening zonder overna...	3614
90	Kunst	3624
93	Sport en recreatie	1620
94	Levensbeschouwelijke en politieke organisaties...	4682
95	Reparatie van computers en consumentenartikelen	832
96	Wellness en overige dienstverlening; uitvaartb...	4947

Table B.8: Overview of division representation in the dataset.

C FEATURE TYPES AND CLASSIFIER PERFORMANCE

C.1 CLASSIFICATION REPORTS

Section	precision	recall	f1-score	support
A	0.322034	0.513514	0.395833	37
C	0.456647	0.624506	0.527546	253
F	0.778094	0.854688	0.814594	640
G	0.752688	0.759952	0.756303	1658
H	0.678571	0.853933	0.756219	178
I	0.648276	0.820961	0.724470	229
J	0.705699	0.710115	0.707900	959
K	0.463519	0.448133	0.455696	241
L	0.529801	0.695652	0.601504	115
M	0.795339	0.621514	0.697764	1757
N	0.610989	0.559356	0.584034	497
P	0.666220	0.698034	0.681756	712
Q	0.770318	0.740867	0.755305	1177
R	0.568452	0.729008	0.638796	524
S	0.753000	0.719885	0.736070	1046
accuracy			0.705078	10023
macro avg	0.633310	0.690008	0.655586	10023
weighted avg	0.714319	0.705078	0.706085	10023

Table C.1: tfidf vectors.

Section	precision	recall	f1-score	support
A	0.072519	0.513514	0.127090	37
C	0.403727	0.513834	0.452174	253
F	0.663473	0.865625	0.751186	640
G	0.727734	0.606152	0.661402	1658
H	0.608108	0.758427	0.675000	178
I	0.553571	0.812227	0.658407	229
J	0.665868	0.579771	0.619844	959
K	0.446809	0.348548	0.391608	241
L	0.294964	0.713043	0.417303	115
M	0.751048	0.509960	0.607458	1757
N	0.310757	0.627767	0.415723	497
P	0.763466	0.457865	0.572432	712
Q	0.689811	0.776551	0.730616	1177
R	0.506173	0.547710	0.526123	524
S	0.750000	0.608031	0.671595	1046
accuracy			0.610795	10023
macro avg	0.547202	0.615935	0.551864	10023
weighted avg	0.661526	0.610795	0.620951	10023

Table C.2: Average word embeddings.

	precision	recall	f1-score	support
A	0.081395	0.567568	0.142373	37
C	0.473684	0.284585	0.355556	253
F	0.736446	0.764062	0.750000	640
G	0.693520	0.238842	0.355316	1658
H	0.642857	0.707865	0.673797	178
I	0.556034	0.563319	0.559653	229
J	0.550586	0.539103	0.544784	959
K	0.271186	0.398340	0.322689	241
L	0.445545	0.391304	0.416667	115
M	0.550733	0.512806	0.531093	1757
N	0.350427	0.494970	0.410342	497
P	0.748538	0.359551	0.485769	712
Q	0.493441	0.830926	0.619183	1177
R	0.358787	0.654580	0.463514	524
S	0.556503	0.499044	0.526210	1046
accuracy			0.512521	10023
macro avg	0.500646	0.520458	0.477130	10023
weighted avg	0.564307	0.512521	0.505396	10023

Table C.3: Average word embeddings with tfidf weighting.

C.2 CONFUSION MATRICES

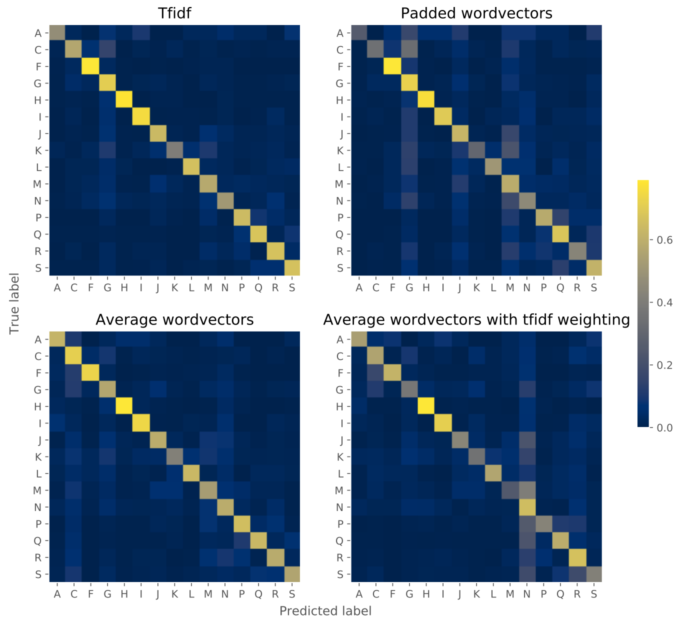

Figure C.1: Confusion matrices for the use of different features. The confusion matrices are normalized on the true labels. On the x-axis, the predicted labels are listed and on every y-axis, the true labels are listed, both for every setting. A more yellow square at (x,y) indicates that the classifier often predicts label x when a true label is y.

D PERFORMANCE OF THE CLASSIFIERS

D.1 CLASSIFICATION REPORTS

Section	precision	recall	f1-score	support
A	0.238636	0.567568	0.336000	37
C	0.460606	0.600791	0.521441	253
F	0.772404	0.848437	0.808637	640
G	0.737959	0.776236	0.756614	1658
H	0.652582	0.780899	0.710997	178
I	0.645161	0.873362	0.742115	229
J	0.699248	0.678832	0.688889	959
K	0.585106	0.456432	0.512821	241
L	0.546763	0.660870	0.598425	115
M	0.774608	0.618099	0.687559	1757
N	0.643026	0.547284	0.591304	497
P	0.636243	0.675562	0.655313	712
Q	0.755725	0.757009	0.756367	1177
R	0.563877	0.732824	0.637344	524
S	0.771368	0.690249	0.728557	1046
accuracy			0.699890	10023
macro avg	0.632221	0.684297	0.648826	10023
weighted avg	0.709068	0.699890	0.700458	10023

Table D.1: SGD (MH).

Section	precision	recall	f1-score	support
A	0.125628	0.675676	0.211864	37
C	0.398496	0.628458	0.487730	253
F	0.717105	0.851562	0.778571	640
G	0.766408	0.655006	0.706341	1658
H	0.547893	0.803371	0.651481	178
I	0.559459	0.903930	0.691152	229
J	0.671306	0.653806	0.662441	959
K	0.506977	0.452282	0.478070	241
L	0.396313	0.747826	0.518072	115
M	0.778400	0.553785	0.647157	1757
N	0.600877	0.551308	0.575026	497
P	0.626807	0.669944	0.647658	712
Q	0.729575	0.758709	0.743857	1177
R	0.545588	0.708015	0.616279	524
S	0.751136	0.631931	0.686397	1046
accuracy			0.662077	10023
macro avg	0.581465	0.683041	0.606806	10023
weighted avg	0.689929	0.662077	0.667000	10023

Table D.2: SGD (SVM).

Section	precision	recall	f1-score	support
A	0.273973	0.540541	0.363636	37
C	0.435013	0.648221	0.520635	253
F	0.778102	0.832812	0.804528	640
G	0.742443	0.740651	0.741546	1658
H	0.713542	0.769663	0.740541	178
I	0.670103	0.851528	0.750000	229
J	0.690096	0.675704	0.682824	959
K	0.469636	0.481328	0.475410	241
L	0.581395	0.652174	0.614754	115
M	0.777361	0.590211	0.670980	1757
N	0.581028	0.591549	0.586241	497
P	0.624843	0.699438	0.660040	712
Q	0.767296	0.725573	0.745852	1177
R	0.543018	0.734733	0.624493	524
S	0.752303	0.702677	0.726644	1046
accuracy			0.690312	10023
macro avg	0.626677	0.682454	0.647208	10023
weighted avg	0.702933	0.690312	0.692269	10023

Table D.3: LR.

Section	precision	recall	f1-score	support
A	0.220000	0.594595	0.321168	37
C	0.465190	0.581028	0.516696	253
F	0.767988	0.817187	0.791824	640
G	0.685959	0.772014	0.726447	1658
H	0.735955	0.735955	0.735955	178
I	0.624606	0.864629	0.725275	229
J	0.670011	0.626694	0.647629	959
K	0.638710	0.410788	0.500000	241
L	0.525547	0.626087	0.571429	115
M	0.735719	0.608423	0.666044	1757
N	0.621687	0.519115	0.565789	497
P	0.646091	0.661517	0.653713	712
Q	0.746924	0.722175	0.734341	1177
R	0.488312	0.717557	0.581144	524
S	0.748565	0.623327	0.680230	1046
accuracy			0.673351	10023
macro avg	0.621418	0.658739	0.627846	10023
weighted avg	0.684951	0.673351	0.673994	10023

Table D.4: SGD (LR).

Section	precision	recall	f1-score	support
A	0.122449	0.162162	0.139535	37
C	0.254613	0.272727	0.263359	253
F	0.582931	0.565625	0.574148	640
G	0.545779	0.553679	0.549701	1658
H	0.588235	0.561798	0.574713	178
I	0.558559	0.541485	0.549889	229
J	0.475584	0.467153	0.471331	959
K	0.314410	0.298755	0.306383	241
L	0.433628	0.426087	0.429825	115
M	0.456573	0.442800	0.449581	1757
N	0.334783	0.309859	0.321839	497
P	0.485753	0.502809	0.494134	712
Q	0.527592	0.536109	0.531816	1177
R	0.371025	0.400763	0.385321	524
S	0.484449	0.491396	0.487897	1046
accuracy			0.478200	10023
macro avg	0.435757	0.435547	0.435298	10023
weighted avg	0.478682	0.478200	0.478304	10023

Table D.5: DT.

Section	precision	recall	f1-score	support
A	0.500000	0.324324	0.393443	37
C	0.468421	0.351779	0.401806	253
F	0.641809	0.820312	0.720165	640
G	0.646455	0.797346	0.714016	1658
H	0.671795	0.735955	0.702413	178
I	0.596386	0.864629	0.705882	229
J	0.669834	0.588113	0.626319	959
K	0.766129	0.394191	0.520548	241
L	0.615385	0.626087	0.620690	115
M	0.672120	0.561184	0.611663	1757
N	0.672727	0.372233	0.479275	497
P	0.621551	0.664326	0.642227	712
Q	0.677009	0.723025	0.699260	1177
R	0.434119	0.597328	0.502811	524
S	0.723977	0.591778	0.651236	1046
accuracy			0.642023	10023
macro avg	0.625181	0.600841	0.599450	10023
weighted avg	0.649798	0.642023	0.636831	10023

Table D.6: RF.

Section	precision	recall	f1-score	support
A	0.186441	0.297297	0.229167	37
C	0.295597	0.557312	0.386301	253
F	0.786585	0.806250	0.796296	640
G	0.715951	0.703860	0.709854	1658
H	0.623318	0.780899	0.693267	178
I	0.559748	0.777293	0.650823	229
J	0.644906	0.679875	0.661929	959
K	0.402490	0.402490	0.402490	241
L	0.416667	0.608696	0.494700	115
M	0.759653	0.548662	0.637145	1757
N	0.510549	0.486922	0.498455	497
P	0.625538	0.612360	0.618879	712
Q	0.741176	0.695837	0.717791	1177
R	0.456311	0.627863	0.528514	524
S	0.688912	0.641491	0.664356	1046
accuracy			0.641724	10023
macro avg	0.560923	0.615140	0.579331	10023
weighted avg	0.662533	0.641724	0.646357	10023

Table D.7: RNN.

Section	precision	recall	f1-score	support
A	0.260870	0.486486	0.339623	37
C	0.381546	0.604743	0.467890	253
F	0.763869	0.839063	0.799702	640
G	0.745383	0.681544	0.712035	1658
H	0.627193	0.803371	0.704433	178
I	0.649123	0.807860	0.719844	229
J	0.673695	0.699687	0.686445	959
K	0.465517	0.448133	0.456660	241
L	0.483221	0.626087	0.545455	115
M	0.758696	0.595902	0.667517	1757
N	0.584906	0.561368	0.572895	497
P	0.561983	0.764045	0.647619	712
Q	0.786008	0.649108	0.711028	1177
R	0.547078	0.643130	0.591228	524
S	0.714840	0.704589	0.709677	1046
accuracy			0.670957	10023
macro avg	0.600262	0.661008	0.622137	10023
weighted avg	0.687260	0.670957	0.673680	10023

Table D.8: RCNN.

D.2 CONFUSION MATRICES

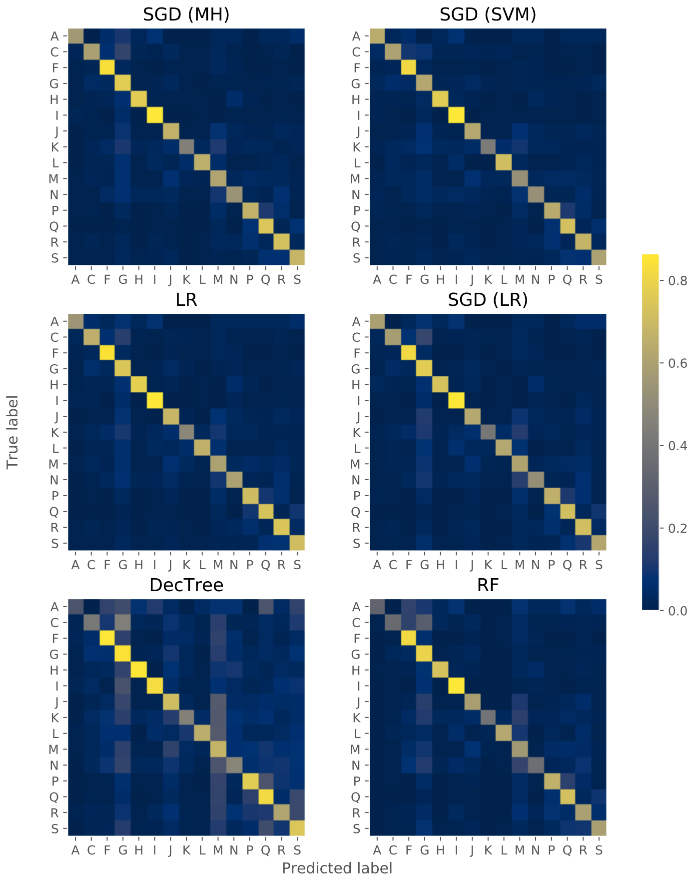

Figure D.1: Confusion matrices for different classifiers.

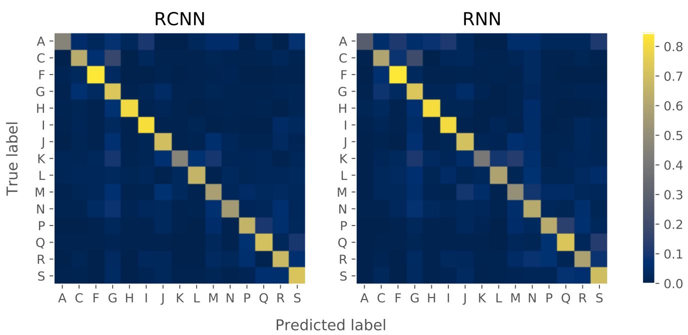

Figure D.2: Confusion matrices of the RCNN and RNN.

D.3 NEURAL NETWORK CONFIGURATION

Layer	Output shape	Activation (see table D.10)	Remarks
Document input	(None, 300)	Identity	Document as integer sequence. These integers correspond to entries in the embedding matrix.
Left context input	(None, 300)	Identity	Left context as integer sequence.
Right context input	(None, 300)	Identity	Right context as integer sequence.
Embedding	(None, None, 300)	None	Not trainable. Embedding matrix of shape (50000, 300). Used to obtain word embeddings by integer indices.
Document LSTM	(None, None, hidden1)	Tanh	
Left context LSTM	(None, None, hidden1)	Tanh	
Right context LSTM	(None, None, hidden1)	Tanh	
Concatenate	(None, None, 3· hidden1)	Identity	Concatenates Document, Left context and Right context LSTM layers
Convolution (1D)	(None, None, hidden2)	Tanh	Semantic layer
Max pooling	(None, hidden2)	Identity	Semantic pooling
Dense	(None, 20)	Softmax	Output layer

Table D.9: RCNN layout.

Type	Function	Remarks
Identity	$a(y) = y$	No activation function.
Tanh	$a(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$	
Softmax	$a(y) = \frac{e^{y_i}}{\sum_j e^{y_j}}$	Converts outputs to probability distribution.

Table D.10: Activation functions.

E "FLAT" VERSUS HIERARCHICAL CLASSIFICATION

E.1 CLASSIFICATION REPORTS

Division	precision	recall	f1-score	support
1	0.229885	0.540541	0.322581	37
10	0.319149	0.769231	0.451128	39
25	0.421488	0.689189	0.523077	74
31	0.260870	0.681818	0.377358	44
32	0.314815	0.708333	0.435897	48
33	0.241379	0.583333	0.341463	48
41	0.686508	0.657795	0.671845	263
42	0.300000	0.800000	0.436364	15
43	0.692308	0.696133	0.694215	362
45	0.756554	0.855932	0.803181	236
46	0.683241	0.462017	0.551263	803
47	0.640351	0.589661	0.613961	619
49	0.707317	0.805556	0.753247	108
50	0.500000	0.428571	0.461538	7
52	0.462687	0.673913	0.548673	46
53	0.619048	0.764706	0.684211	17
55	0.526882	0.803279	0.636364	61
56	0.640394	0.773810	0.700809	168
59	0.381944	0.733333	0.502283	75
62	0.723958	0.608759	0.661380	685
63	0.587097	0.457286	0.514124	199
64	0.308642	0.172414	0.221239	145
66	0.610169	0.750000	0.672897	96
68	0.584507	0.721739	0.645914	115
69	0.694545	0.837719	0.759443	228
70	0.687773	0.485362	0.569106	649
71	0.570423	0.591241	0.580645	274
72	0.160494	0.419355	0.232143	31

Continued on next page

Table E.1 – continued from previous page

Division	precision	recall	f1-score	support
73	0.541176	0.539062	0.540117	256
74	0.795349	0.536050	0.640449	319
77	0.424051	0.567797	0.485507	118
78	0.476562	0.762500	0.586538	80
79	0.456311	0.758065	0.569697	62
80	0.363636	0.869565	0.512821	23
81	0.692308	0.796460	0.740741	113
82	0.329114	0.257426	0.288889	101
85	0.704180	0.615169	0.656672	712
86	0.781167	0.721814	0.750318	816
88	0.604863	0.551247	0.576812	361
90	0.619718	0.607735	0.613668	362
93	0.543210	0.814815	0.651852	162
94	0.619145	0.649573	0.633994	468
95	0.466667	0.590361	0.521277	83
96	0.845070	0.848485	0.846774	495
accuracy	0.626160	0.626160	0.626160	0.62616
macro avg	0.535794	0.648799	0.567784	10023
weighted avg	0.650879	0.626160	0.628698	10023

Table E.1: “Flat” classification approach.

Division	precision	recall	f1-score	support
1	0.289855	0.540541	0.377358	37
10	0.479167	0.589744	0.528736	39
25	0.459184	0.608108	0.523256	74
31	0.375000	0.613636	0.465517	44
32	0.432432	0.666667	0.524590	48
33	0.328125	0.437500	0.375000	48
41	0.645985	0.673004	0.659218	263
42	0.526316	0.666667	0.588235	15
43	0.618483	0.720994	0.665816	362

Continued on next page

Table E.2 – continued from previous page

Division	precision	recall	f1-score	support
45	0.774704	0.830508	0.801636	236
46	0.622739	0.600249	0.611287	803
47	0.615964	0.660743	0.637568	619
49	0.664122	0.805556	0.728033	108
50	0.545455	0.857143	0.666667	7
52	0.475410	0.630435	0.542056	46
53	0.578947	0.647059	0.611111	17
55	0.638889	0.754098	0.691729	61
56	0.658291	0.779762	0.713896	168
59	0.602740	0.586667	0.594595	75
62	0.653203	0.684672	0.668567	685
63	0.696721	0.427136	0.529595	199
64	0.266055	0.200000	0.228346	145
66	0.614035	0.729167	0.666667	96
68	0.539474	0.713043	0.614232	115
69	0.745968	0.811404	0.777311	228
70	0.646602	0.513097	0.572165	649
71	0.640553	0.507299	0.566191	274
72	0.360000	0.290323	0.321429	31
73	0.597765	0.417969	0.491954	256
74	0.771689	0.529781	0.628253	319
77	0.504854	0.440678	0.470588	118
78	0.590361	0.612500	0.601227	80
79	0.589041	0.693548	0.637037	62
80	0.571429	0.695652	0.627451	23
81	0.726496	0.752212	0.739130	113
82	0.375000	0.207921	0.267516	101
85	0.659686	0.707865	0.682927	712
86	0.758186	0.737745	0.747826	816
88	0.603774	0.531856	0.565538	361
90	0.537118	0.679558	0.600000	362
93	0.580645	0.777778	0.664908	162
94	0.644252	0.634615	0.639397	468
95	0.744186	0.385542	0.507937	83

Continued on next page

Table E.2 – continued from previous page

Division	precision	recall	f1-score	support
96	0.843813	0.840404	0.842105	495
accuracy			0.640028	10023
macro avg	0.581653	0.617974	0.590105	10023
weighted avg	0.644294	0.640028	0.636859	10023

Table E.2: Hierarchical classification approach.

E.2 CONFUSION MATRICES

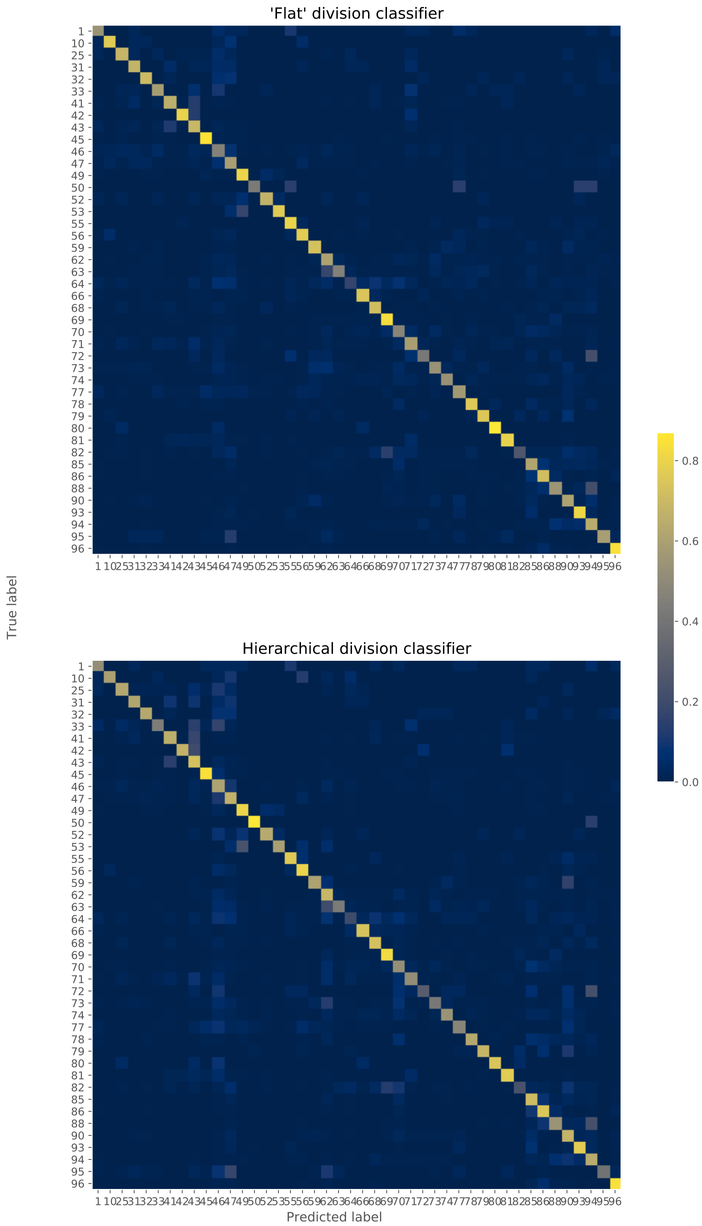

Figure E.1: Confusion matrices for flat and hierarchical classification. On the x-axis lists the predicted labels. The y-axis lists the true labels. A high intensity diagonal (yellow) indicates a high correct classification rate. Lower intensity (grey or blue) at the diagonal indicate a low correct classification rate for class corresponding to the y-coordinate. High intensity squares at other locations than the diagonal indicate that the classifier mistakes the class corresponding to the y-coordinate for the class corresponding to the x-coordinate. Therefore, these figures can be used to visualize classifier performance per class.

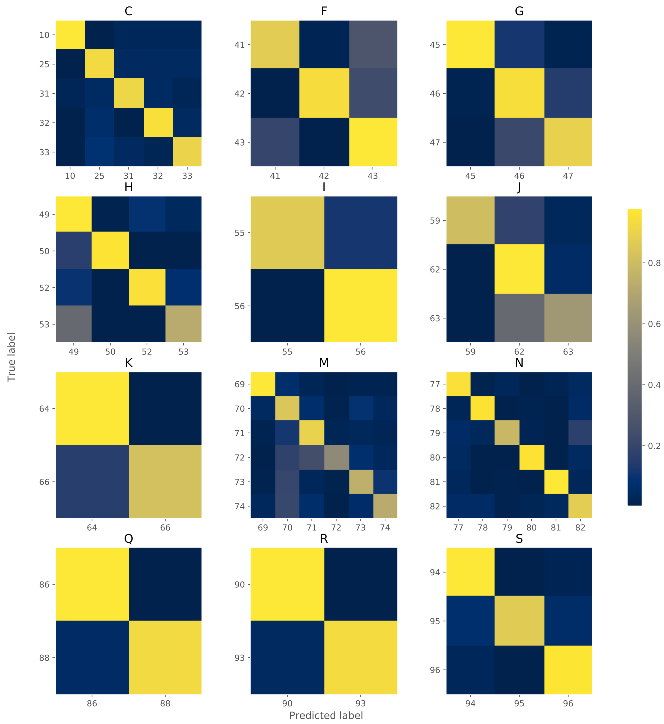

Figure E.2: Confusion matrices for every division classifier of the hierarchical classifier.

BIBLIOGRAPHY

[1] Charu C. Aggarwal and ChengXiang Zhai. "A Survey of Text Classification Algorithms." In: *Mining Text Data*. Springer, 2012, pp. 163–222.

[2] Chid Apté, Fred Damerau, and Sholom Weiss. *Text mining with decision rules and decision trees*. Citeseer, 1998.

[3] Chidanand Apté, Fred Damerau, and Sholom Weiss. "Automated Learning of Decision Rules for Text Categorization." In: *Transactions on Information Systems* 12.3 (1994), pp. 233–251.

[4] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A Simple but Tough-to-Beat Baseline for Sentence Embeddings." In: (2016). 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

[5] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, et al. "The Gene Ontology Consortium Gene Ontology: Tool for the Unification of Biology." In: *Nat Genet* 25.1 (2000), pp. 25–29.

[6] L. Douglas Baker and Andrew Kachites McCallum. "Distributional Clustering of Words for Text Classification." In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1998, pp. 96–103.

[7] A. J. Barrett. "Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB). Enzyme Nomenclature. Recommendations 1992. Supplement 4: corrections and additions (1997)." In: *European Journal of Biochemistry* 250.1 (1997), p. 1.

[8] Zafer Barutcuoglu and Christopher DeCoro. "Hierarchical Shape Classification using Bayesian Aggregation." In: *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. IEEE. 2006, pp. 44–44.

[9] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. " O'Reilly Media, Inc.", 2009.

[10] Léon Bottou. "Online Algorithms and Stochastic Approximations." In: *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[11] Léon Bottou and Olivier Bousquet. "The Tradeoffs of Large Scale Learning." In: *Optimization for Machine Learning*. MIT Press, 2012, pp. 351–368.

[12] Leo Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32.

[13] CBS, Centraal Bureau voor de Statistiek. "Standaard Bedrijfsindeling 2008." In: *na* (2019).

[14] Michelangelo Ceci and Donato Malerba. "Classifying Web Documents in a Hierarchy of Categories: a Comprehensive Study." In: *Journal of Intelligent Information Systems* 28.1 (2007), pp. 37–78.

[15] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. "Hierarchical Classification: combining Bayes with SVM." In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM. 2006, pp. 177–184.

[16] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. "Scalable Feature Selection, Classification and Signature Generation for organizing Large Text Databases into Hierarchical Topic Taxonomies." In: *The VLDB Journal* 7.3 (1998), pp. 163–178.

[17] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: Synthetic Minority Over-Sampling Technique." In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[18] Jan Salomon Cramer. *The Origins of Logistic Regression*. Tech. rep. Amsterdam: Tinbergen Institute, 2002.

[19] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. "Representation Learning for very Short Texts using Weighted Word Embedding Aggregation." In: *Pattern Recognition Letters* 80 (2016), pp. 150–156.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A Large-Scale Hierarchical Image Database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255.

[21] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. "Hierarchical Annotation of Medical Images." In: *Pattern Recognition* 44.10-11 (2011), pp. 2436–2449.

[22] Pedro Domingos. "A few useful Things to Know about Machine Learning." In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[23] Cicero Dos Santos and Maira Gatti. "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014, pp. 69–78.

[24] Susan Dumais. "Using SVMs for Text Categorization." In: *IEEE Intelligent Systems* 13.4 (1998), pp. 21–23.

[25] Susan Dumais and Hao Chen. "Hierarchical Classification of Web Content." In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2000, pp. 256–263.

[26] Eurostat and NACE. "Rev. Statistical Classification of Economic Activities in the European Community." In: *Office for Official Publications of the European Communities, Luxemburg* (2008).

[27] Caspar J. Fall, Atilla Törcsvári, Karim Benzineb, and Gabor Karetka. "Automated Categorization in the International Patent Classification." In: *SIGIR Forum* 37.1 (2003), pp. 10–25.

[28] B. W. A. C. Farley and W. Clark. "Simulation of Self-Organizing Systems by Digital Computer." In: *Transactions of the IRE Professional Group on Information Theory* 4.4 (1954), pp. 76–84.

[29] Eibe Frank and Remco R. Bouckaert. "Naive Bayes for Text Classification with Unbalanced Classes." In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2006, pp. 503–510.

[30] Alexander Genkin, David D. Lewis, and David Madigan. "Large-Scale Bayesian Logistic Regression for Text Categorization." In: *Technometrics* 49.3 (2007), pp. 291–304.

[31] Alec Go, Richa Bhayani, and Lei Huang. "Twitter Sentiment Classification using Distant Supervision." In: *CS224N Project Report, Stanford* 1.12 (2009).

[32] Hervé Goëau, Hervé Glotin, Willem-Pier Vellinga, Robert Planqué, Andreas Rauber, and Alexis Joly. "LifeCLEF Bird Identification Task 2014." In: *CLEF: Conference and Labs of the Evaluation Forum*. Vol. CEUR Workshop Proceedings. 1180. Sheffield, 2014, pp. 585–597.

[33] Michael Granitzer and Peter Auer. "Experiments with hierarchical text classification." In: *Proceedings of 9th International Conference on Artifical Intelligence, ACTA Press, Benidorm, Spain, IASTED*. 2005.

[34] Jun Han and Claudio Moraga. "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning." In: *International Workshop on Artificial Neural Networks*. Springer. 1995, pp. 195–201.

[35] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 1949. ISBN: 9781135631901.

[36] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. "OHSUMED: an Interactive Retrieval Evaluation and New Large Test Collection for Research." In: *SIGIR'94*. Springer. 1994, pp. 192–201.

[37] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[38] Phoenix X Huang, Bastiaan J Boom, and Robert B Fisher. "Hierarchical Classification with Reject Option for Live Fish Recognition." In: *Machine Vision and Applications* 26.1 (2015), pp. 89–102.

[39] Thorsten Joachims. *Making Large-Scale SVM Learning Practical*. Tech. rep. 28. Dortmund: Universität Dortmund, Sonderforschungsbereich 475 - Komplexitätsreduktion in Multivariaten Datenstrukturen, 1998.

[40] Thorsten Joachims. "Text Categorization with Support Vector Machines: Learning with many Relevant Features." In: *European Conference on Machine Learning*. Springer. 1998, pp. 137–142.

[41] Byung-soo Kim, Jae Young Park, Anna C. Gilbert, and Silvio Savarese. "Hierarchical Classification of Images by Sparse Approximation." In: *Image and Vision Computing* 31.12 (2013), pp. 982–991.

[42] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." In: *arXiv preprint arXiv:1408.5882* (2014).

[43] Jyrki Kivinen, Manfred K. Warmuth, and Peter Auer. "The Perceptron Algorithm versus Winnow: Linear versus Logarithmic Mistake Bounds when few Input Variables are Relevant." In: *Artificial Intelligence* 97.1-2 (1997), pp. 325–343.

[44] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. "Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network." In: *Neurocomputing: Algorithms, Architectures and Applications*. Vol. F68. Springer, 1990, pp. 41–50.

[45] Daphne Koller and Mehran Sahami. *Hierarchically Classifying Documents using very few Words*. Tech. rep. Stanford: InfoLab, 1997.

[46] A. N. Kolmogorov. "Sulla Determinazione Empírica di uma Legge di Distribuzione." In: *G. Ist. Ital. Attuari*. Vol. 4. 1933, pp. 83–91.

[47] Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Schubert. "Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects." In: *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM. 2004, pp. 102–113.

[48] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. "Recurrent Convolutional Neural Networks for Text Classification." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI '15. AAAI Press, 2015, pp. 2267–2273.

[49] Ken Lang. "Newsweeder: Learning to Filter Netnews." In: *Proceedings of the Twelfth International Conference on Machine Learning*. 1995, pp. 331–339.

[50] Leah Larkey. "Some Issues in the Automatic Classification of US Patents." In: *Working Notes for the AAAI-98 Workshop on Learning for Text Categorization*. 1998, pp. 87–90.

[51] Wee Sun Lee and Bing Liu. "Learning with Positive and Unlabeled Examples using Weighted Logistic Regression." In: *ICML*. Vol. 3. 2003, pp. 448–455.

[52] Thomas Martin Lehmann, Henning Schubert, Daniel Keysers, Michael Kohnen, and Berthold B. Wein. "The IRMA Code for Unique Classification of Medical Images." In: *Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation*. Vol. 5033. International Society for Optics and Photonics. 2003, pp. 440–451.

[53] Tao Li and Mitsunori Ogihara. "Music Genre Classification with Taxonomy." In: *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005*. Vol. 5. IEEE. 2005, pp. v–197.

[54] Hans Peter Luhn. "A Statistical Approach to Mechanized Encoding and Searching of Literary Information." In: *IBM Journal of Research and Development* 1.4 (1957), pp. 309–317.

[55] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning Word Vectors for Sentiment Analysis." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. Association for Computational Linguistics. 2011, pp. 142–150.

[56] Charles Mathis and Thomas Breuel. "Classification using a Hierarchical Bayesian Approach." In: *Object Recognition supported by User Interaction for Service Robots*. Vol. 4. IEEE. 2002, pp. 103–106.

[57] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.

[58] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. "Weakly-Supervised Hierarchical Text Classification." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 6826–6833.

[59] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *arXiv preprint arXiv:1301.3781* (2013).

[60] Jelena Mirkovic and Peter Reiher. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms." In: *ACM SIGCOMM Computer Communication Review* 34.2 (2004), pp. 39–53.

[61] F. Pedregosa et al. “Scikit-Learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[62] José Peña, Pedro Gutiérrez, César Hervás-Martínez, Johan Six, Richard Plant, and Francisca López-Granados. “Object-Based Image Classification of Summer Crops with Machine Learning Methods.” In: *Remote Sensing* 6.6 (2014), pp. 5019–5041.

[63] John W. Ratcliff and David E. Metzener. “Pattern Matching: The Gestalt Approach.” In: *Dr. Dobb's Journal* 13.7 (1988), p. 46.

[64] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora.” English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[65] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method.” In: vol. 22. 3. 1951, pp. 400–407.

[66] Monica Rogati and Yiming Yang. “High-Performing Feature Selection for Text Classification.” In: *Proceedings of the Eleventh International Conference on Information and Knowledge Management*. 2002, pp. 659–661.

[67] Frank Rosenblatt. “The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain.” In: *Psychological Review* 65.6 (1958), p. 386.

[68] Miguel E. Ruiz and Padmini Srinivasan. “Hierarchical Text Categorization using Neural Networks.” In: *Information Retrieval* 5.1 (2002), pp. 87–118.

[69] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors.” In: *Cognitive Modeling* 5.3 (1988), p. 1.

[70] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2002.

[71] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers.” In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229.

[72] Or Sharir, Barak Peleg, and Yoav Shoham. “The Cost of Training NLP Models: A Concise Overview.” In: *arXiv preprint arXiv:2004.08900* (2020).

[73] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. “The Princeton Shape Benchmark.” In: *Proceedings Shape Modeling Applications, 2004.* IEEE. 2004, pp. 167–178.

[74] Carlos N. Silla and Alex A. Freitas. “A Survey of Hierarchical Classification across different Application Domains.” In: *Data Mining and Knowledge Discovery* 22.1-2 (2011), pp. 31–72.

[75] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle: Association for Computational Linguistics, 2013, pp. 1631–1642.

[76] Karen Sparck Jones. "A Statistical Interpretation of Term Specificity and its Application in Retrieval." In: *Journal of Documentation* 28.1 (1972), pp. 11–21.

[77] V. Srividhya and R. Anitha. "Evaluating Preprocessing Techniques in Text Categorization." In: *International Journal of Computer Science and Application* 47.11 (2010), pp. 49–51.

[78] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP." In: *arXiv preprint arXiv:1906.02243* (2019).

[79] Aixin Sun, E. P. Lim, W. K. Ng, and Jaideep Srivastava. "Blocking Reduction Strategies in Hierarchical Text Classification." In: *IEEE Transactions on Knowledge and Data Engineering* 16.10 (2004), pp. 1305–1308.

[80] Xingping Sun, Yibing Li, Hongwei Kang, and Yong Shen. "Automatic Document Classification Using Convolutional Neural Network." In: *Journal of Physics: Conference Series*. Vol. 1176. 3. IOP Publishing. 2019, p. 032029.

[81] Tatiana Tommasi, Barbara Caputo, Petra Welter, Mark Oliver Güld, and Thomas M. Deserno. "Overview of the CLEF 2009 Medical Image Annotation Track." In: *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer. 2009, pp. 85–93.

[82] Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. "Text Mining Techniques for Patent Analysis." In: *Information Processing & Management* 43.5 (2007), pp. 1216–1247.

[83] Giorgio Valentini. "True Path Rule Hierarchical Ensembles for Genome-Wide Gene Function Prediction." In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 8.3 (2011), pp. 832–847.

[84] Cornelis Joost Van Rijsbergen. "Information Retrieval (2nd ed.)" In: *Journal of the American Society for Information Science* (1979).

[85] Damir Vandic, Flavius Frasincar, and Uzay Kaymak. "A Framework for Product Description Classification in E-commerce." In: *J. Web Eng.* 17.1&2 (2018), pp. 1–27.

[86] S. Vijayarani, Ms. J. Ilamathi, and Ms. Nithya. "Preprocessing Techniques for Text Mining - An Overview." In: *International Journal of Computer Science & Communication Networks* 5.1 (2015), pp. 7–16.

[87] Maarten Wullink, Giovane C. M. Moura, and Cristian Hesselman. "DMAP: Automating Domain Name Ecosystem Measurements and Applications." In: *IFIP/IEEE Network Traffic Measurement and Analysis Conference (TMA 2018)*. Vienna, Austria, June 2018.

[88] Zhongzhe Xiao, Emmanuel Dellandrea, Weibei Dou, and Liming Chen. "Hierarchical Classification of Emotional Speech." In: *IEEE Transactions on Multimedia* 37 (2007).

[89] Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. "Deep Classification in Large-Scale Text Hierarchies." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Association for Computing Machinery, 2008, pp. 619–626.

[90] Alec Yenter and Abhishek Verma. "Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment Analysis." In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE. 2017, pp. 540–546.

[91] Yongwook Yoon, Changki Lee, and Gary Geunbae Lee. "An Effective Procedure for constructing a Hierarchical Text Classification System." In: *Journal of the American Society for Information Science and Technology* 57.3 (2006), pp. 431–442.

[92] Seongwook Youn and Dennis McLeod. "A Comparative Study for Email Classification." In: *Advances and Innovations in Systems, Computing Sciences and Software Engineering*. Springer, 2007, pp. 387–391.

[93] ChengXiang Zhai and Sean Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Morgan & Claypool, 2016.

[94] G. K. Zipf. *The Psycho-Biology of Language: An Introduction to Dynamic Philology*. Cognitive Psychology v. 21. Written in 1935. Routledge, 1999. ISBN: 9780415209762.